UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E MÉTODOS QUANTITATIVOS

ALAN DA SILVA ASSUNÇÃO

PROCESSOS T-STUDENT EM CLASSIFICAÇÃO

FORTALEZA
2021

ALAN DA SILVA ASSUNÇÃO

PROCESSOS T-STUDENT EM CLASSIFICAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Modelagem e Métodos Quantitativos.

Orientador: Prof. Dr. José Ailton Alencar Andrade.

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A873p Assunção, Alan da Silva.
Processos t-Student em Classificação / Alan da Silva Assunção. – 2021.
117 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Modelagem e Métodos Quantitativos, Fortaleza, 2021.
Orientação: Prof. Dr. José Ailton Alencar Andrade.

1. Modelagem bayesiana não paramétrica. 2. Processo t-Student. 3. Processo Gaussiano. 4. Modelagem robusta. 5. Classificadores. I. Título.

CDD 510

ALAN DA SILVA ASSUNÇÃO

PROCESSOS T-STUDENT EM CLASSIFICAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Modelagem e Métodos Quantitativos.

Aprovada em: 17/02/2021

BANCA EXAMINADORA

Prof. Dr. José Ailton Alencar Andrade. (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Gualberto Segundo Agamez Montalvo
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Ferraz do Nascimento
Universidade Federal do Piauí (UFPI)

À minha família. Sem vocês, jamais teria conseguido chegar até este ponto crucial da minha caminhada. Louvo a Deus pela vida de cada um. Vocês são o meu alicerce.

AGRADECIMENTOS

A Deus, por seu amor inigualável, ao Senhor Jesus Cristo, por sua Graça salvadora, e ao Espírito Santo, por seu companherismo, e pela certeza de que nunca estarei só. A Eles sejam a Honra, a Glória e o Louvor para todo sempre.

À minha família, por sempre ter me apoiado e me ensinado desde cedo a necessidade de lutar pelos meus sonhos, e o valor de buscar viver de forma íntegra nessa terra.

Aos meus demais parentes, por sempre me ajudarem em tudo que precisei.

Ao meu orientador, professor José Ailton, por todo apoio e aprendizagem que adquiri durante o tempo de pesquisa que desenvolvemos juntos. Obrigado pela confiança em mim depositada, e por me motivar a não parar na minha caminhada acadêmica.

Aos amigos, em especial João Marcos e Kleunice Oliveira, por sempre se prontificarem nos momentos difíceis.

Aos irmãos de fé e pastores que se encontram na igreja Nova Aliança em Timon-MA, suas palavras motivadoras foram combustíveis essenciais para chegar até aqui.

À igreja Batista em Antônio Bezerra, por ter me recebido tão bem durante a minha estadia em Fortaleza, e pelos amigos que fiz por lá durante esse tempo.

Aos professores do PPGMMQ, por todo conhecimento que obtive durante o tempo de curso.

Aos funcionários do PPGMMQ, e aos demais funcionários do DEMA, por toda gentileza.

Aos colegas de curso, pela recepção e ajuda que obtive durante o curso. Agradeço por tudo que aprendi na companhia de vocês.

À Fundação Cearense de Apoio ao Desenvolvimento (Funcap), pelo financiamento desta pesquisa de mestrado via bolsa de estudos.

E a todos que colaboraram direta e indiretamente para a conclusão deste trabalho.

Olha sempre para frente, mantém teu olhar fixo no objetivo a ser alcançado. Reflete sobre tuas escolhas e sobre o caminho por onde andas, e todos os teus planos serão bem sucedidos! (Provérbios 4:25-26, versão KJA).

RESUMO

Modelos de regressão baseados em Processo Gaussiano (GPR) são excelentes alternativas não-paramétricas para modelagem de problemas complexos, e apresentam muitas atratividades das quais podemos citar: boa performance preditiva, flexibilidade não-paramétrica, interpretabilidade e relativamente fácil implementação conceitual. Dessa forma, a proposta de modelos de classificação de GP é um caminho bastante útil para lidar com os mais diversos problemas de classificação. Entretanto, modelos de Processo Gaussiano não possuem robustez a *outliers*, devido à natureza de cauda leve da distribuição Gaussiana. Com isso, neste trabalho, propomos um novo classificador com um Processo t-Student (TPC), como distribuição a priori, como forma alternativa aos Processos Gaussianos. O TPC tem por objetivo lidar de forma adequada com problemas de classificação cujos dados de entrada $\mathbf{x}$ estejam contaminados por *outliers*. O classificador proposto teve seu desempenho avaliado junto ao tradicional classificador de Processo Gaussiano (GPC) em conjuntos de dados reais da área biomédica, em que os *outliers* foram gerados artificialmente. Para as aplicações no caso de classificação binária, dados de diagnóstico de coluna vertebral e diagnóstico de câncer de mama foram utilizados. Para as aplicações no caso multiclasse, o conjunto de observações de coluna vertebral em sua versão multiclasse foi considerado. As inferências sobre os modelos abordados nesta pesquisa foram feitas por meio do método NUTS, uma técnica MCMC variante do Monte Carlo Hamiltoniano. Pelos resultados das aplicações realizadas neste trabalho, o classificador TPC alcançou resultados bastante promissores, principalmente na tarefa de classificação multiclasse, em que a proposta de robustez em dados contaminados por *outliers* foi bem atendida.

Palavras-chave: Classificador de Processo Gaussiano; robustez; Classificador de Processo t-Student; modelagem não-paramétrica.

ABSTRACT

Gaussian Process regression models (GPR) are excellent non-parametric alternatives for modeling complex problems, among the advantages, we can mention: good predictive performance, non-parametric flexibility, interpretability and easy computational implementation. Thus, the proposal for GP classification models is useful to deal with most diverse classification problems. However, Gaussian Process models are not robust to outliers, due to the light-tailed nature of the Gaussian distribution. In this work, we propose a new t-Student Process classifier (TPC), as an alternative to Gaussian Processes. The TPC aproach is able to deal most adequately with classification problems which input data $\mathbf{x}$ are contaminated by outliers. The proposed classifier had its performance evaluated with the traditional Gaussian Process classifier (GPC) in real data sets from the biomedical area, where the outliers were generated artificially. For applications in the case of binary classification, spinal diagnostic data and breast cancer diagnosis were used. For applications in the multiclass case, the set of vertebral column observations in its multiclass version was considered. The inferences about the models covered in this work were made using the NUTS method, an MCMC technique variant of Hamiltonian Monte Carlo. Due to the results of the applications carried out in this work, the TPC classifier achieved very promising results, mainly in the task of multiclass classification, in which the proposal of robustness in data contaminated by textit outliers was well attended.

Keywords: Gaussian Process classifier; robustness; t-Student Process Classifier; non-parametric modeling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADVI	*Automatic Differentiation Variational Inference*
ARD	Determinação de Relevância Automática
CV	conjunto de dados de Coluna Vertebral
EP	método *Expectation Propagation*
EQ	função de covariância Exponencial Quadrática
GP	Processo Gaussiano
GPC	Classificação de Processo Gaussiano
GPR	Regressão de Processo Gaussiano
GPRT	Regressão de Processo Gaussiano com ruído t-Student
HMC	Monte Carlo Hamiltoniano
iid	independente e identicamente distribuído
IRLS	Mínimos Quadrados Iterativamente Reponderados
L-BFGS	*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*
MAP	Máximo a posteriori
MCMC	*Markov Chain Monte Carlo*
NUTS	*No-U-Turn Sampler*
RQ	função de covariância Radial Quadrática
RV	Teoria da Variação Regular
SMC	Monte Carlo Sequencial
TP	Processo t-Student
TPC	Classificador de Processo t-Student
TPRD	Regressão de Processo t-Student com ruído t-Student dependente
TPRK	Regressão de Processo t-Student com ruído dentro do *kernel*
TPRT	Regressão de Processo t-Student com ruído t-Student
VB	Bayes variacional
WDBC	conjunto de dados *Wisconsin Diagnostic Breast Cancer*

LISTA DE SÍMBOLOS

C_k	classe k dentre $k = 1, \ldots, K$ classes
$\mathbf{x}$	vetor de entradas de treino ou vetor de características
$\mathbf{X}$	Matriz de entradas de treino: matriz de delineamento
$p(\mathbf{x}\|C_k)$	distribuição de classe-condicional
$\boldsymbol{\Sigma}$	matriz de covariância de ordem $n \times n$; e matriz Hessiana na aproximação de Laplace
D	dimensão do vetor de entradas
R_k	região de decisão (ou superfície de decisão) k
K	quantidade de classes em um problema de classificação
$\mathbb{R}$ e $\mathbb{R}_+$	conjunto dos reais e reais positivos respectivamente
$f(\mathbf{x})$	função subjacente
n e n_*	tamanho das amostras de treino e de teste respectivamente
$\boldsymbol{\theta}$	conjunto de parâmetros desconhecidos presentes na função de verossimilhança de $\mathbf{y}$=f($\mathbf{x}$)+$\boldsymbol{\varepsilon}$;
$\boldsymbol{\eta}$	conjunto de hiperparâmetros a serem estimados nos modelos GPC e TPC
$\propto$	proporcional
$\in$	pertencimento a um dado conjunto. Exemplo: $\mathbf{x} \in \mathbb{R}$
$\boldsymbol{\Theta}$	espaço paramétrico
$\mathbf{w}$	vetor de parâmetros
$\boldsymbol{\varepsilon}$	ruído do modelo; $\boldsymbol{\varepsilon}$ é o tamanho do passo leapfrog no HMC.
$\mathbf{y}$	vetor de saídas observadas do modelo
$N(.,.)$ e $N_n(.,.)$	distribuição Gaussiana univariada e distribuição Gaussiana Multivariada de ordem n respectivamente
σ^2	variância do modelo $\mathbf{y}$=f($\mathbf{x}$)+$\boldsymbol{\varepsilon}$, onde $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$
$\|\|.\|\|$	distância euclideana
$\|\mathbf{X}\|$ e $\|\mathbf{x} - \mathbf{x}'\|$	determinante da matriz $\mathbf{X}$ e valor absoluto entre os vetores $\mathbf{x}$ e $\mathbf{x}'$

$\sim$	distribui-se como. Exemplo: $\varepsilon \sim N(0, \sigma^2)$
exp	função exponencial e^x
$\mathbf{X}^T$	matriz transposta
$\mathbf{X}^{-1}$	matriz inversa
$\mathbf{x}_*$	entrada de teste
$\mathbf{I}$	matriz identidade de ordem $n \times n$.
$\mathbf{0}$	vetor de zeros
$\boldsymbol{\mu}$	vetor de médias $n \times 1$
$MVT_n(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$	distribuição t-Student Multivariada de ordem n, onde ν são os graus de liberdade, $\boldsymbol{\mu}$ o vetor de médias, e $\boldsymbol{\Sigma}$ a matriz de covariância
$(\Omega, \mathscr{A}, \mathscr{P})$	espaço de probabilidade
$\boldsymbol{\chi}$	conjunto de índices do Processo Estocástico e também o espaço de entradas dos modelos de GP e TP
$\mathbf{K}$	matriz de covariância dos modelos GP e TP de ordem $n \times n$
$\mathbf{f}$	vetor de valores latentes
$k(\mathbf{x}, \mathbf{x}')$	função de covariância
$\mathscr{GP}(.,.)$	Processo Gaussiano
$\mathscr{D}$	conjunto de dados de treinamento
$\mathscr{D}_*$	conjunto de dados de teste
$\boldsymbol{\psi}$	conjunto de hiperparâmetros da função de covariância
$\mathbf{K}_*$	matriz de covariância entre dados de treino e de teste $K(\mathbf{X}, \mathbf{X}_*)$
$\mathbf{K}_{**}$	matriz de covariância entre os dados de teste somente $K(\mathbf{X}_*, \mathbf{X}_*)$
$\mathbf{f}_*$	predição de Processo Gaussiano (variável aleatória)
$\hat{f}_*$ e $v(\hat{f}_*)$	média e variância da distribuição preditiva de teste
$\boldsymbol{\phi}$	vetor de dimensão M de funções base $\phi_1, \ldots, \phi_M$
$\boldsymbol{\Phi}$	matriz de dimensão $n \times M$ composta das funções base $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M$
$\boldsymbol{\Pi}(n)$	conjunto de todas as matrizes reais, positivas definidas, simétricas $n \times n$
$IW_n(\nu, \mathbf{K})$	distribuição Wishart Inversa de ordem n

$\mathscr{IWP}(.,.)$	Processo Wishart Inverso
$\mathscr{TP}(.,.,.)$	Processo t-Student
$O(.)$	ordem de complexidade computacional
$t-Student(v,\mu,\sigma)$	distribuição t-Student univariada
Gamma$(.,.)$	distribuição Gamma
Tr()	Traço de uma matriz
$\mathrm{E}_q()$	Esperança sob aproximação de Laplace
$\hat{\mathbf{f}}$	moda a posteriori
$K_v(.)$	função Bessel modificada.
$\mathbf{t}$	vetor de rótulos de classe.
$p(C_k\|\mathbf{x})$	distribuição condicional da classe C_k.
$L(k,k')$	função de perda. Refere-se à perda incorrida ao tomar a decisão k', quando a a classe verdadeira é C_k.
$R(k,k')$	função risco. Equivalente à perda esperada, ou seja: $\sum_k L(k,k')p(C_k\|\mathbf{x})$.
$\sigma(.)$	função sigmóide. Podendo ser a função logística, ou a função de distribuição Gaussiana.
log(.)	função logarítmica na base e.
∇ e $\nabla\nabla$	gradiente de uma dada função (vetor de derivadas de primeira ordem) e hessiana (matriz de segundas derivadas) respectivamente
$\pi(\mathbf{x})$ e $\bar{\pi}_*(\mathbf{x})$	probabilidade de classe e probabilidade de classe de teste: $p(y=1\|\mathbf{x})$ e $p(y=1\|\mathbf{x}_*)$.
$H(\mathbf{q},\mathbf{p})$	função de energia Hamiltoniana apresentada no algoritmo HMC, em que $\mathbf{q}$ são variáveis de posição e $\mathbf{p}$ são variáveis de momento.
L	A quantidade de passos leapfrog ε na discretização da simulação do Hamiltoniano
$\mathbf{L}$	matriz triangular inferior $n \times n$ referente à decomposição de Cholesky tal que: uma matriz $\mathbf{K}$ $n \times n$ pode ser escrita como $\mathbf{K} = \mathbf{L}^T\mathbf{L}$.

SUMÁRIO

1 INTRODUÇÃO

A área de Reconhecimento de Padrões tem uma longa história, que vem evoluindo com grande êxito no decorrer do tempo, principalmente com o avanço computacional, que proporcionou um crescimento avassalador de algoritmos para as mais diversas situações. Podemos perceber que descobertas que foram fundamentais para a ciência se utilizaram do princípio básico de observação de padrões. Por exemplo, as observações astronômicas de Tycho Brahe no século 16 permitiram as descobertas das leis empíricas do movimento planetário por Johannes Kepler (BISHOP, 2006). Segundo Pao (1989), atividades comuns da rotina de um ser humano, como falar, escrever, interpretação de imagens entre outros, envolvem de forma implícita reconhecer padrões. Morais (2010), cita uma lista de aplicações atuais nas quais se utilizam técnicas sofisticadas de Reconhecimento de Padrões, dentre elas: reconhecimento biométrico (incluindo o reconhecimento da face, da íris e das impressões digitais), diagnóstico médico, bioinformática etc.

O campo de Reconhecimento de Padrões está preocupado com a descoberta de regularidades nos dados através do uso de algoritmos de computador (BISHOP, 2006). Pal Sankar K e Bezdek (1992), por exemplo, definem Reconhecimento de Padrões como a busca por estruturas em dados. De posse da identificação dessas estruturas, desejamos tomar decisões tais como classificar os dados em uma categoria dentre algumas pré-definidas, ou fazer predições para novos dados com base em algum modelo paramétrico estabelecido para descrever o mecanismo gerador dos valores observados.

Há uma gama variada de formas de abordagem que podemos utilizar para o problema de Reconhecimento de Padrões. Com base nos trabalhos de Jain *et al.* (2000) e Campos (2001), podemos apresentar um breve resumo dado a seguir

1. Casamento (*Tamplate Matching*) (THEODORIDIS; KOUTROUMBAS, 2009);
2. Abordagem Sintática (THEODORIDIS; KOUTROUMBAS, 2009; MORIMOTO *et al.*, 1996);
3. Redes Neurais (THEODORIDIS; KOUTROUMBAS, 2009);
4. Lógica Nebulosa (BLOCH, 1999);
5. Morfologia matemática com aprendizado computacional (BARRERA *et al.*, 2000);
6. Estatística (THEODORIDIS; KOUTROUMBAS, 2009; BISHOP, 2006).

Essa forma de separação é apresentada apenas para fins didáticos, como argumentam Campos (2001) e Jain *et al.* (2000), pois certos modelos de uma determinada

abordagem na lista acima podem ser equivalentes a algum outro modelo que utiliza abordagem teoricamente diferente. Por exemplo, alguns modelos de Redes Neurais podem ser equivalentes ou similares a certos modelos estatísticos. A Tabela 3 do artigo de Jain *et al.* (2000) apresenta alguns desses modelos.

A área de RP é vasta, mas possui como uma de suas principais atividades a tarefa de classificação. Em linhas gerais, o objetivo do problema de classificação é atribuir um padrão desconhecido a uma classe dentre um conjunto de classes finitas. Dentre as abordagens mais comuns para tratar tais problemas, temos a abordagem estatística, que segundo Jain *et al.* (2000) pode ser caracterizada como a atribuição de um dado vetor $\mathbf{x}=[x_1,\ldots,x_D]^T$ de D-características, conhecido como padrão, a uma de K categorias (classes) $C_1,\ldots,C_K$ pré-definidas. O conjunto de todos os vetores de características $\mathbf{x}$ compõem uma matriz $\mathbf{X}$, conhecida na prática por Matriz de delineamento[1]. As características são assumidas como sendo provenientes de uma função densidade (ou de massa, no caso discreto) de probabilidade condicionada nos padrões de classe. Dessa forma, um padrão $\mathbf{x}$ pertencente à classe C_k, é visto como uma observação proveniente da distribuição de classe-condicional $p(\mathbf{x}|C_k)$.

Portanto, dependendo do tipo de informação que dispomos das distribuições de classe-condicionais, podemos propor distintos classificadores estatísticos. Por exemplo, se temos todas as distribuições de classe-condicionais conhecidas, exceto os seus respectivos parâmetros, então estamos lidando com um problema de classificação estatístico paramétrico. Nesta linha de classificadores, o classificador Discriminante Quadrático pode ser citado como um exemplo no qual às distribuições de classe-condicionais $p(\mathbf{x}|C_k)$ atribuímos distribuições Gaussianas que possuem distintas matrizes de covariâncias $\mathbf{\Sigma}_k$. Mas, se a forma das distribuições de classe-condicionais forem completamente desconhecidas, então estamos tratando com um problema não-paramétrico (JAIN *et al.*, 2000), e como um exemplo dessa classe de modelos, temos os classificadores de Processos Gaussianos.

O modelo de classificação que será utilizado no presente trabalho utiliza abordagem estatística, fornecendo estimativas probabilísticas de um determinado padrão $\mathbf{x}$ pertencer a uma determinada classe C_k, $k=1,\ldots,K$, cuja representação de padrões é dada por um conjunto de D características (JAIN *et al.*, 2000), $\mathbf{x}=[x_1,x_2,\ldots,x_D]^T$, sendo cada

[1] Esta matriz é composta por todos os vetores de entrada de treino do modelo.

padrão visto como um ponto do espaço D-dimensional.

De posse de uma amostra de dados, o classificador a utilizará para construir o que chamaremos de fronteiras de decisão (ou superfícies de decisão) do espaço de entradas[2], que por sua vez, dão origem às chamadas regiões de decisão. Essas superfícies são multidimensionais, particionando o espaço de entradas em K regiões, $R_1, \ldots, R_K$, para um problema com K classes, cada região correspondendo a uma classe (MORAIS, 2010). Para obtenção de mais detalhes sobre a área de Reconhecimento de Padrões, pode-se consultar Duda e Hart (1973), Jain *et al.* (2000) e Bishop (2006).

1.2 Proposta do Trabalho

Modelos de Processos Gaussianos são modelos probabilísticos não-paramétricos que proporcionam uma conveniente estrutura de inferência para modelos complexos. Estes modelos, a grosso modo, não assumem uma forma paramétrica pré-estabelecida e deixam que os próprios dados manifestem a relação existente entre as variáveis em estudo. Comumente, os modelos de Processo Gaussiano (GP) podem ser classificados como modelos de regressão ou classificação, dependendo se o valor da variável resposta $\mathbf{y}$ é contínua ou discreta.

Os modelos de Classificação de Processo Gaussiano (GPC) são excelentes propostas à modelagem probabilística de classificação, pois os mesmos apresentam boa flexibilidade não-paramétrica, proporcionando dessa forma boa adequação a diferentes tarefas de classificação. Williams e Barber (1998), por exemplo, apresentam resultados do desempenho de um classificador de GP com outros classificadores apresentados em Ripley (1994) e Ripley (1996), utilizando três bancos de dados, dos quais, dois são casos binários (ou seja K=2) e o terceiro é um problema de classificação multiclasse. Em linhas gerais, para os três bancos de dados, o classificador de GP apresenta um desempenho igual ao melhor classificador apresentado em Ripley (1994) e Ripley (1996), a saber, um classificador de Rede Neural de duas unidades ocultas com conexões diretas de entrada para saída, uma unidade de saída logística e treinada com a técnica de estimação de máxima verossimilhança.

[2] O espaço de entradas aqui refere-se ao conjunto de todos os possíveis valores que uma dada entrada $\mathbf{x}$ pode assumir. Em aplicações práticas, esse espaço de entradas pode ser equivalente ao $\mathbb{R}^D$.

O Processo t-Student (TP) é uma forma de distribuição a priori alternativo ao Processo Gaussiano (SHAH *et al.*, 2014), e proporcionam uma modelagem mais robusta a *outliers*, por serem estes processos de cauda pesada. Trabalhos como os de Tang *et al.* (2016) e Tang *et al.* (2017), mostram empiricamente que modelos de regressão com Processo t-Student têm desempenho muito melhor em dados contaminados por *outliers* quando comparados aos modelos de GP.

Dados reais, ao contrário do que poderia se pensar, comumente apresentam comportamento de cauda pesada (NAIR *et al.*, 2013), e são contaminados por *outliers* (BENDRE *et al.*, 1994; NIU *et al.*, 2015; NIU *et al.*, 2016). Para problemas de classificação, em termos práticos, um *outlier* pode se originar de duas formas: através de atributos ruidosos (variáveis de entrada) e por meio de rótulos[3] ruidosos (BARRETO; BARROS, 2016). Atributos ruidosos afetam os valores observados das variáveis de entrada, e quanto aos rótulos ruidosos, estes surgem quando se atribui a um dado vetor de características $\mathbf{x}$ um rótulo de classe que não é o verdadeiro, por exemplo, quando se muda um rótulo observado de 1 para 0 em classificação binária. Vale ressaltar que este último tipo de *outlier* independe das variáveis de entrada. Bootkrajang (2016) argumenta que o surgimento de *outliers* pode ser comum em tarefas de classificação complexas como em dados Biomédicos e dados de classificação textual, e que atualmente, certas práticas para obtenção de dados de treinamento[4] rotulados de forma rápida e barata, introduzem ruídos nos dados. Barreto e Barros (2016) também argumentam que devido às técnicas de classificação atuais serem muito mais automáticas, executadas por computadores de forma programada, sem a avaliação minuciosa adequada, os *outliers* podem permanecer sem serem notados. Dessa forma, construir modelos de classificação que possam manipular adequadamente estes dados tornam-se necessários, pois a presença de *outliers* no banco de dados afeta o desempenho dos classificadores, levando os mesmos a apresentarem conclusões imprecisas (BOOTKRAJANG, 2016).

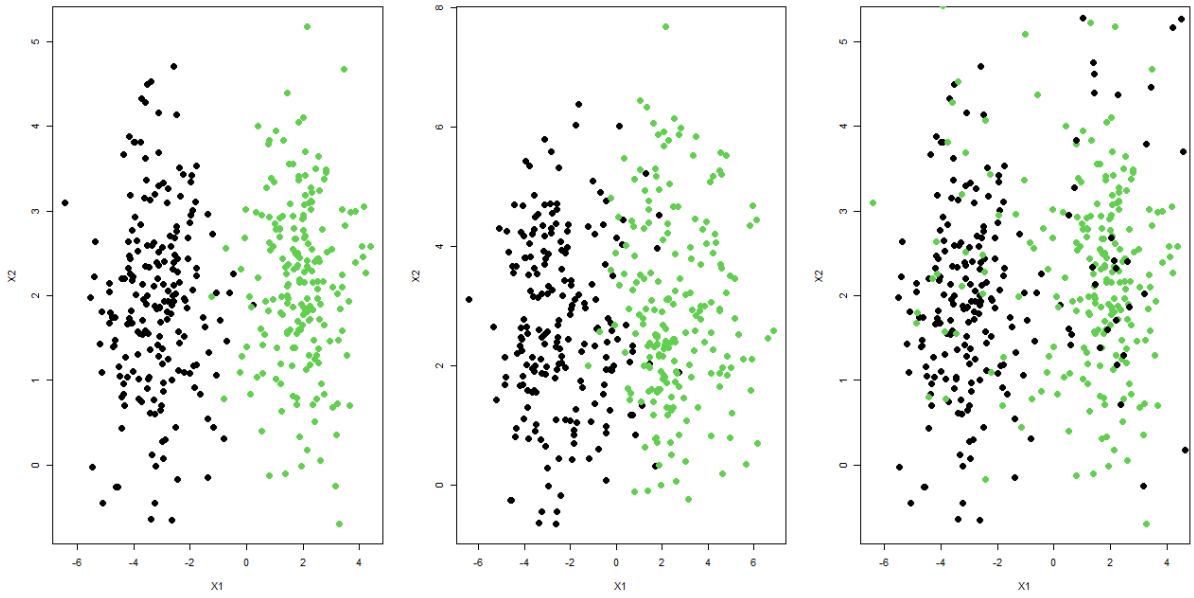
A nossa proposta tem por objetivo utilizar a estrutura dos classificadores de

[3] Um rótulo de classe em termos práticos é a forma de codificação numérica das classes de um problema de classificação.

[4] Dados de treino, em poucas palavras, são amostras de dados utilizadas para otimizar/estimar os (hiper-)parâmetros do classificador, que posteriormente terá seu desempenho avaliado por uma outra amostra, chamada de amostra de teste.

Processo Gaussiano, substituindo a distribuição a priori de Processo Gaussiano por um Processo t-Student. Classificadores com este tipo de processo como distribuição a priori, até o presente momento, ainda não foram propostos na literatura. O nosso objetivo em desenvolver tal classificador consiste em apresentar uma nova alternativa para tentar lidar com bancos de dados cujas observações são contaminadas por *outliers*, especificamente as variáveis de entrada, que são os vetores de características $\mathbf{x} \in \mathbb{R}^D$. Vetores de características $\mathbf{x}$ altamente ruidosos influenciam a rotulação de classes dos pontos observados. Este cenário, intuitivamente, é o que pode ser pensado como sendo o mais comum na prática, e portanto é mais geral, podendo ser encontrado com muito mais frequência em problemas do mundo real se comparados ao *outlier* provocado apenas pela mudança de rótulo (BOOTKRAJANG, 2016). A Figura 1 traz uma ilustração para dados de classificação binária, onde $\mathbf{x} \in \mathbb{R}^2$, dos tipos de ruídos que podem ocorrer em dados de classificação. Os gráficos representam respectivamente: i) os dados originais sem alterações; ii) dados com pertubações nas variáveis de entrada; iii) dados, cujos vetores de entrada tiveram seus rótulos invertidos.

Figura 1 – Ilustração de tipos de ruído em dados de classificação. Gráfico à esquerda: dados originais. Gráfico central: ruído provocado por perturbações nas variáveis de entrada. Gráfico à direita: ruído provocado pela alteração dos rótulos originais.



Fonte: Autoria própria.

Este trabalho será desenvolvido sob o enfoque bayesiano, abordagem comumente utilizada na literatura para modelagem de GP. Mesmo sob a ótica bayesiana, há diversas

técnicas existentes para tratar problemas de classificação, dentre elas podemos citar as aproximações Gaussianas feitas pelo método de Laplace (WILLIAMS; BARBER, 1998) ou pelo método *Expectation Propagation* (EP) (MINKA, 2001), aproximação sob Algoritmos de Campo Médio (OPPER; WINTHER, 2000), e abordagem variacional (GIBBS; MACKAY, 1997).

Apesar da variedade de métodos, optamos por utilizar técnicas de inferência baseada em amostragem *Markov Chain Monte Carlo* (MCMC) que possuem a propriedade de serem assintoticamente exatas no limite das simulações computacionais. Precisamente, utilizaremos o esquema de amostragem *No-U-Turn Sampler* (NUTS) que vem a ser uma extensão do Monte Carlo Hamiltoniano (HMC) (DUANE *et al.*, 1987), seguindo os trabalhos de Williams e Barber (1998) e Neal (1997).

1.3 Objetivos

Os objetivos do presente trabalho consistem em avaliar a eficiência e o desempenho do Classificador de Processo t-Student (TPC) com relação ao modelo *baseline* de GP, em dados contaminados por *outliers* ou que possam apresentar comportamento de cauda pesada. Para alcançar estes objetivos, faremos aplicações em dados reais, cujas variáveis de entrada serão perturbadas[5] sob situações específicas para gerar os *outliers*. Os *outliers* serão gerados sob certas quantidades para verificar o comportamento do desempenho dos classificadores TPC e GPC à medida que aumentamos a quantidade de ruídos nos dados.

1.4 Dados para as aplicações

Para as aplicações realizadas neste trabalho, bancos de dados pertencentes a área biomédica foram utilizados, dos quais, dois bancos de dados foram destinados para as aplicações de classificação binária, e para o caso multiclasse, apenas um conjunto de dados.

Para as aplicações de classificação binária, o primeiro conjunto de observações consiste em classificar o diagnóstico da coluna vertebral de pacientes em "Normal" e "Anormal", em que, dentro da categoria "Anormal" constitui-se os pacientes diagnosticados com "Espondilolistese" e "Hérnia de disco". Cabe ressaltar que este banco de dados foi

[5] O termo "perturbação" neste contexto, refere-se à geração de valores nas variáveis bem distoante das demais observações.

organizado de duas formas ligeiramente diferente, mas possuindo tarefa de classificação relacionada. Essa construção foi idealizada para que o referido banco de dados pudesse ser utilizado tanto na tarefa de classificação binária, quanto na tarefa de classificação multiclasse. Por fim, cada paciente neste conjunto de dados é representado por seis variáveis biomecânicas derivadas da forma e orientação da pelve e coluna lombar.

O segundo conjunto de dados destinado à aplicação binária, refere-se ao diagnóstico de câncer de mama de pacientes. O diagnóstico de cada paciente é classificado, após avaliação do tecido de massa mamária, em "Benigno" ou "Maligno". Cada paciente é representado por um vetor de trinta variáveis, extraídas da análise da imagem digitalizada obtida da massa mamária através de punção aspirativa por agulha fina[6].

Para a aplicação no caso multiclasse, o banco de dados de diagnóstico de coluna vertebral em sua versão multiclasse foi considerado. As observações para esta versão do conjunto de dados de coluna vertebral são as mesmas que para a versão do caso binário, com uma leve diferença no número de categorias de diagnóstico, que agora são três: " Espondilolistese", "hérnia de disco" e "Normal".

1.5 Organização do trabalho

No Capítulo 2 é apresentada uma introdução aos Processos Gaussianos e aos Processos t-Student, destacando características essenciais dos mesmos para a modelagem de dados. O Capítulo 3 apresenta uma introdução sobre o problema de classificação, e introduz os classificadores de Processo Gaussiano e t-Student. No Capítulo 4, temos as aplicações em dados reais, cujo desempenho do classificador proposto será avaliado em relação ao modelo tradicional de GP. No Capítulo 5, finalizamos este trabalho com uma discussão dos resultados e propostas de desenvolvimentos para trabalhos futuros.

[6] A punção aspirativa por agulha fina, ou biópsia aspirativa por agulha fina, pode ser definida como o exame de aspiração de células de lesões de órgãos superficiais ou profundos usando agulha fina, com o propósito de um diagnóstico para predizer a natureza da lesão.

2 PROCESSOS GAUSSIANOS E PROCESSOS T-STUDENT

Os modelos de Processo Gaussiano obtiveram grande relevância nos últimos anos, nos quais a recorrente utilização de tais modelos nos mais diversos campos científicos comprovam o bom resultado dos mesmos. Engenharias, Inteligência Artificial, Probabilidade e Estatística, Telecomunicações, Geoestatística, Física Aplicada, Automação de Controle de Sistemas e Sistemas Dinâmicos são algumas das áreas que podem ser citadas.

Processos Gaussianos são modelos probabilísticos que procuram identificar o relacionamento entre os dados medidos. Esse tipo de estrutura caracteriza os chamados modelos não-paramétricos. Nos casos de regressão, por exemplo, o modelo ser chamado de não-paramétrico justifica-se por não se assumir uma forma paramétrica para a função de regressão (KUSS, 2006), abordagem comum nos modelos de regressão tradicionais. Modelos não-paramétricos são muito atraentes, principalmente aliados à abordagem bayesiana, pois permitem que sejam definidos um número arbitrário de parâmetros sem exigir cálculos de alteração de dimensionalidade dispendiosos para realizar inferências.

Vale também ressaltar que Processos Gaussianos também possuem suas limitações. Bernardo *et al.* (1998) apresentam pelo menos duas delas: (i) Processos Gaussianos não são distribuições a priori apropriadas para funções que possuam pontos de descontinuidade; (ii) funções, cujos graus de suavidade ou escala de variação variam como uma função das entradas de maneira desconhecida, são muito problemáticas. Para estas situações, o autor cita outras abordagens de modelagem mas, propõe caminhos alternativos de modelagem usando ainda Processos Gaussianos.

Os modelos de Processo Gaussiano possuem complexidade computacional básica da ordem de $O(n^3)$[1] devido ao cálculo da inversão da matriz de covariância $n \times n$. Por isso, dentro do contexto da literatura de GP, tem-se proposto métodos de inferência aproximada para poder manusear de forma satisfatória as questões de complexidade computacional, assim como, poder fazer aplicações para bancos de dados com um grande número de observações.

Normalmente em situações práticas com modelos de Processo Gaussiano, quando

[1] A complexidade de algoritmos computacionais é medida em termos de comportamento assintótico de funções dos programas, ou seja, o comportamento dos algoritmos para n bem grande. Diz-se que uma função $f(n)$ domina assintoticamente outra função $g(n)$, e escrevemos $g(n) = O(f(n))$, se existirem duas constantes positivas c e m tais que, para $n \geq m$, temos $|g(n)| \leq c|f(n)|$.

os dados de saída são variáveis contínuas, categoriza-se este modelo como sendo um caso de regressão. Caso as variáveis de saída sejam um conjunto finito de valores discretos, trata-se este problema como sendo um caso de classificação. Esta forma de definir estas duas situações é bem comum na área de *Machine Learning* quando se trabalha com o caso de aprendizado supervisionado (WILLIAMS; RASMUSSEN, 2006).

O modelo é composto por um conjunto de dados de entrada-saída que caracterizam o comportamento do modelo, e uma função de covariância que descreve a relação dos dados de saída em função das respectivas entradas. A predição do modelo de GP é dada em termos de uma distribuição Gaussiana Multivariada, identificada por seu vetor de médias e matriz de covariância, em que o valor médio representa o valor mais provável da saída do modelo e a variância, a confiança dessa predição (KOCIJAN, 2016).

Trabalhos com Processos Gaussianos não são tão recentes, pelo menos para modelos com séries temporais onde se utilizaram abordagens de predições com GP, em que indícios da teoria básica estavam presentes nos trabalhos de Wiener (1949) e Kolmogorov (1941). Após algum tempo, os modelos de GP passaram a ser usados para resolver problemas de regressão (O'HAGAN, 1978), e depois disso disseminou-se pela comunidade *Machine Learning* no final dos anos 1990, onde (NEAL, 1996) desenvolveu o trabalho pioneiro na área, relacionando Redes Neurais artificiais com modelo de GP. Para ter mais informações sobre a história da evolução dos trabalhos com Processos Gaussianos, pode-se consultar Kocijan (2016) e Williams e Rasmussen (2006).

2.1 Abordagem não-paramétrica

Como mencionado no início deste Capítulo, os modelos de Processo Gaussiano são não-paramétricos. Em problemas de regressão, caso que trataremos neste Capítulo, esta abordagem toma como aleatória a parte funcional do modelo, não atribuindo a ela uma forma paramétrica conhecida. Esta abordagem se torna extremamente útil em problemas nos quais os modelos resultantes são complexos, e suposição de relação linear entre as entradas $\mathbf{x}$ e saídas $\mathbf{y}$ do modelo não é adequada.

Aos modelos bayesianos paramétricos tradicionais, atribuímos distribuições de probabilidades a priori para expressar o conhecimento disponível que se tem sobre o experimento. Em contraste, para modelos bayesianos não-paramétricos, por assumir que a função subjacente $f(\mathbf{x})$ é aleatória, atribuímos Processos Estocásticos como distribuição a

priori. Dessa forma, Processo Gaussiano é um Processo Estocástico, e é interpretado como sendo uma distribuição a priori sobre um espaço de funções.

A atribuição de Processos Estocásticos como sendo uma distribuição a priori, leva as variáveis a valores em um espaço infinito-dimensional (KUSS, 2006). Assim, modelos não-paramétricos bayesianos crescem à medida que a quantidade de dados aumentam. Williams e Rasmussen (2006), intuitivamente, retratam o funcionamento de uma função como sendo um vetor muito longo, onde cada entrada representa o valor da função $f(\mathbf{x})$ em $\mathbf{x}$. Assim, as inferências com Processo Gaussiano conseguem fornecer propriedades importantes da função de interesse com base apenas em um subconjunto de pontos, como se todos os pontos tivessem sido levados em consideração. Dessa forma, para qualquer conjunto de dados, apenas um número finito de parâmetros (valores da função) devem ser representados explicitamente (KUSS, 2006).

O fundamento para o surgimento dos modelos não-paramétricos bayesianos veio através dos trabalhos de Ferguson (1973) e Doksum (1974). Estes trabalhos proporcionaram o surgimento de vários outros artigos que foram pioneiros no trato de problemas inferenciais não-paramétricos, e contribuiram de forma significativa para o desenvolvimento da abordagem não-paramétrica que até aquele momento, tinha sido pouca explorada (PHADIA, 2015). Processo Gama (KALBFLEISCH, 1978), Processo Dirichlet (BASU; TIWARI, 2011), Processo Beta (HJORT *et al.*, 1990) e Processo Beta-Stacy (MULIERE; WALKER, 1997) são alguns dos Processos a priori que podemos citar da vasta literatura existente. Para obtenção de mais informações sobre modelagem bayesiana não-paramétrica, pode-se consultar: Moala e O'Hagan (2010) e Phadia (2015).

2.2 Abordagem bayesiana

O modelo proposto neste trabalho utiliza enfoque bayesiano. Com isso, faremos uma breve introdução sobre a abordagem bayesiana tomando como base o modelo de regressão bayesiano, apontando conceitos principais que serão aproveitados no decorrer deste trabalho.

Os modelos bayesianos possuem uma vasta literatura. Sua interpretabilidade intuitiva e boa adequabilidade a diversos tipos de modelo, são algumas das muitas atratividades que tais modelos podem oferecer. A abordagem bayesiana permite que certo conhecimento a priori que se tem de determinado experimento, incorporado

através de uma distribuição de probabilidade chamada distribuição a priori[2], possa ser aumentado ao observar uma amostra de tamanho n retirada do experimento de interesse $\mathbf{X} = [\mathbf{X}_1 = x_1, \mathbf{X}_2 = x_2, \ldots, \mathbf{X}_n = x_n]^T$ (de forma compacta, podemos escrever $\mathbf{X} = \mathbf{x}$, com dimensão $D \times n$), resultando em uma distribuição de probabilidades atualizada, chamada distribuição a posteriori. A regra de atualização do conhecimento a priori é dada pelo teorema de Bayes. Suponha que certo parâmetro θ seja a quantidade que temos interesse em estimar. Se θ for contínuo, a formula de Bayes é dada pela seguinte expressão

$$p(\theta|\mathbf{X} = \mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\theta)p(\theta)}{\int p(\theta, \mathbf{x}) d\theta}, \tag{2.1}$$

em que $p(\theta|\mathbf{X} = \mathbf{x})$ é a distribuição a posteriori, $p(\mathbf{x}|\theta)$ é a função de verossimilhança, $p(\theta)$ é a distribuição a priori, e $p(\mathbf{x})$, a distribuição marginal de $\mathbf{x}$.

Se θ for discreto, a fórmula se resume na expressão a seguir

$$p(\theta|\mathbf{X} = \mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{\sum_\theta p(\theta)p(\mathbf{x}|\theta)}, \tag{2.2}$$

em que o somatório se estende sobre o intervalo de todos os possíveis valores de θ.

Na expressão (2.1), $p(\mathbf{x})$ é independente de θ, e neste caso a mesma é tratada como uma constante de normalização. Assim, em muitas situações, principalmente em processos de estimação , é necessário apenas a obtenção da distribuição a posteriori não normalizada, que é a distribuição a posteriori a menos de uma constante de normalização. Dessa forma, chegamos ao seguinte resultado

$$p(\theta|\mathbf{X} = \mathbf{x}) \propto p(\mathbf{x}|\theta)p(\theta). \tag{2.3}$$

Um dos principais objetivos que se tem em problemas de modelagem é fazer previsões para uma nova observação. A modelagem bayesiana permite um método geral para determinar a distribuição de probabilidade de uma nova observação. Esta distribuição é chamada de distribuição preditiva, e é dada na definição a seguir:

Definição 1 *(Distribuição preditiva): Seja* $\mathbf{X}_1 = x_1, \mathbf{X}_2 = x_2, \ldots, \mathbf{X}_n = x_n$ *(* $\mathbf{X} = \mathbf{x}$*) os dados observados,* x_{n+1} *uma nova observação, e* $p(\theta|\mathbf{x})$ *a distribuição a posteriori de* θ *após*

[2] Se o conhecimento a priori refere-se a uma quantidade univariada, por exemplo um parâmetro escalar desconhecido, então a distribuição a priori é univariada. Caso a quantidade aleatória seja multivariada, exemplo: um vetor de parâmetros; então a distribuição a priori é multivariada.

observar os dados. Tem-se que a distribuição preditiva para x_{n+1} é dada pela seguinte forma

$$p(x_{n+1}|x_1,x_2,\ldots,x_n) = \int_{\theta\in\Theta} p(x_{n+1}|\mathbf{X},\theta)p(\theta|x_1,x_2,\ldots,x_n)d\theta. \tag{2.4}$$

Logo, a distribuição preditiva de x_{n+1}, torna-se uma média da distribuição de $p(x_{n+1}|\mathbf{X},\theta)$ em relação a distribuição a posteriori. Para obtenção de mais detalhes, pode-se consultar Groot e Schervish (2002), Gelman A. *et al.* (2004) e Gamerman e Lopes (2006).

2.2.1 O modelo de regressão linear bayesiano

Como exemplo, vamos apresentar o modelo de regressão linear bayesiano padrão

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}, \quad \mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon}, \tag{2.5}$$

em que $\mathbf{x}$ é o vetor de entradas, pertencente aos $\mathbb{R}^D$, em que D é a dimensão do vetor $\mathbf{x}$, $\mathbf{w}$ é o vetor de parâmetros, $f = f(\mathbf{x})$ é o valor da função subjacente (parte sistemática do modelo), $\boldsymbol{\varepsilon}$ é o vetor de ruídos do modelo e $\mathbf{y}$ são os valores de saída observados.

Em certas situações práticas, a matriz de delineamento $\mathbf{X}$ é aumentada adicionando uma coluna de valores unitários para incluir o parâmetro de víes, possuindo dimensão $(D+1)\times n$. Dito isso, o modelo linear padrão é obtido da expressão (2.5) supondo que o ruído ε_i, para $i = 1,\ldots,n$, é independente e identicamente distribuído (iid) segundo uma distribuição Gaussiana

$$\varepsilon_i \sim N(0,\sigma^2) \quad \text{para} \quad i = 1,\ldots,n. \tag{2.6}$$

A função de verossimilhança é dada pela distribuição conjunta do vetor $\mathbf{y}$

$$p(\mathbf{y}|f(\mathbf{x}),\sigma^2) = \prod_{i=1}^{n} N(y_i|f(\mathbf{x}_i),\sigma^2). \tag{2.7}$$

Devido à suposição de normalidade e independência do ruído, a verossimilhança da expressão (2.7) pode ser fatorada sobre todos os casos y_i, cujo resultado é dado a seguir

$$\begin{aligned} p(\mathbf{y}|\mathbf{X},\mathbf{w}) = \prod_{i=1}^{n} N(y_i|\mathbf{x}_i,\mathbf{w},\sigma^2) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\mathbf{x}_i^T\mathbf{w})^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{-1}{2\sigma^2}||\mathbf{y}-\mathbf{X}^T\mathbf{w}||^2\right), \end{aligned} \tag{2.8}$$

em que $||.||$ denota a distância euclideana.

Vamos atribuir, por simplicidade, uma distribuição a priori Multivariada Gaussiana ao vetor de parâmetros $\mathbf{w}$

$$\mathbf{w} \sim N_{D\times 1}(\mathbf{0}, \mathbf{\Sigma_w}), \tag{2.9}$$

em que $\mathbf{0}$ é o vetor nulo. Veremos que a opção por essa distribuição para expressar o nosso conhecimento a priori dos parâmetros, facilitará os cálculos algébricos do modelo a posteriori final, resultando em um modelo de forma conhecida (tratável analiticamente[3]), dispensando assim, a necessidade de métodos de amostragem aproximada, situação comum em modelos bayesianos em que a posteriori não é um modelo tratável analiticamente.

Este tipo de modelo é conhecido como modelo conjugado, em que a distribuição do modelo a posteriori pertence a mesma família da distribuição a priori. Porém, vale ressaltar o perigo do uso de certas distribuições a priori conjugadas, pois há situações em que as mesmas podem não representar de forma apropriada as incertezas a priori (GAMERMAN, 1997).

De posse da função de verossimilhança e da distribuição a priori, podemos obter o modelo a posteriori. Pela fórmula de Bayes, temos

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}. \tag{2.10}$$

O modelo da expressão (2.10) representa todo o conhecimento que se tem sobre os parâmetros. Fazendo as manipulações algébricas corretas e tomando na proporcional da posteriori apenas os termos que dependem de $\mathbf{w}$, completando o quadrado, obtemos o seguinte resultado (WILLIAMS; RASMUSSEN, 2006)

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \quad \propto \exp\left(-\tfrac{1}{2}(\mathbf{w}-\bar{\mathbf{w}})^T(-\tfrac{1}{\sigma^2}\mathbf{X}\mathbf{X}^T + \mathbf{\Sigma_w}^{-1})(\mathbf{w}-\bar{\mathbf{w}})\right), \tag{2.11}$$

em que $\bar{\mathbf{w}} = \sigma^{-2}(\sigma^{-2}\mathbf{X}\mathbf{X}^T + \mathbf{\Sigma_w}^{-1})^{-1}\mathbf{X}\mathbf{y}$. A forma da expressão acima é equivalente à de uma Gaussiana, logo, a posteriori é um modelo conjugado, como explanado anteriormente. Dessa forma, a posteriori final é dada por

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \sim N_{D\times 1}(\bar{\mathbf{w}}, \mathbf{A}^{-1}), \tag{2.12}$$

em que $\mathbf{A}^{-1} = \sigma^{-2}\mathbf{X}\mathbf{X}^T + \mathbf{\Sigma_w}^{-1}$.

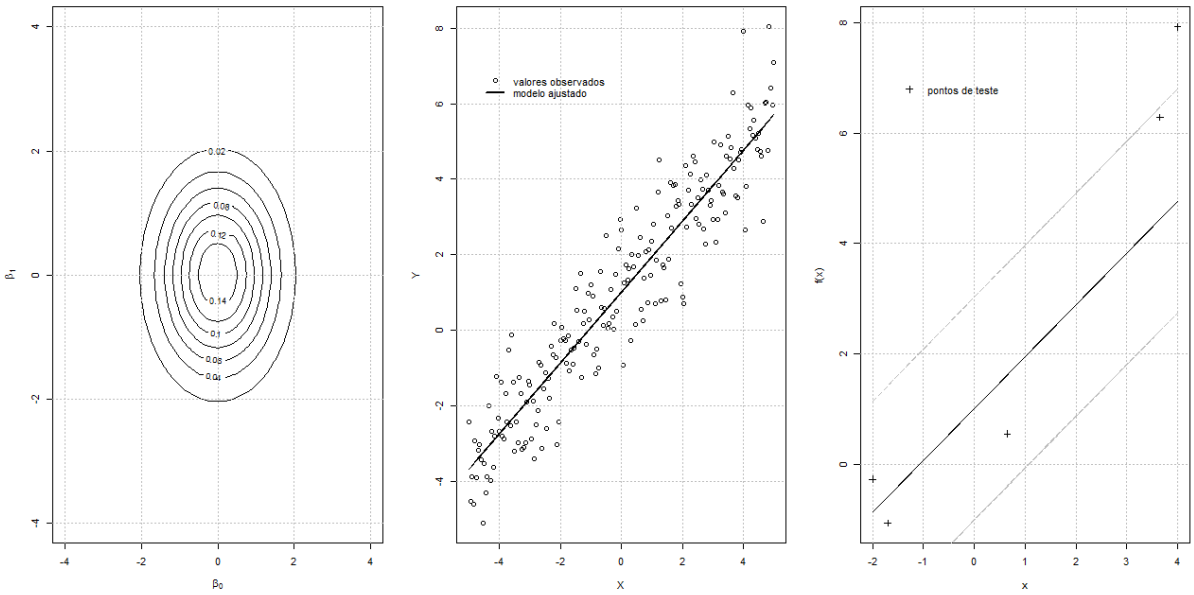
[3] Utilizamos o termo tratável analiticamente para se referir à expressões que podem ser resolvidas de forma fechada.

Após encontrar a distribuição a posteriori, podemos encontrar a distribuição preditiva para uma nova observação $\mathbf{x}_* \in \mathbb{R}^D$ usando a definição 1. Logo, a distribuição preditiva para $f(\mathbf{x}_*)$ no ponto $\mathbf{x}_*$ é dada a seguir

$$\begin{aligned} p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= N(\frac{1}{\sigma^2}\mathbf{x}_*^T \mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}_*^T \mathbf{A}^{-1}\mathbf{x}_*), \end{aligned} \tag{2.13}$$

em que $f_* = f(\mathbf{x}_*) = \mathbf{x}_*^T \mathbf{w}$. A distribuição preditiva novamente resulta em uma distribuição Gaussiana, com parâmetro de localização (média) dado pela média da posteriori na expressão (2.12), multiplicado pela entrada de teste. Se quisermos, por exemplo, fornecer a distribuição preditiva a posteriori para y_*, basta adicionarmos σ^2 à variância da distribuição preditiva de f_*.

Figura 2 – Gráfico à esquerda: gráfico de contornos da distribuição à priori. Gráfico central: pontos observados e modelo ajustado através das estimativas à posteriori de $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$. Gráfico à direita: distribuição preditiva para novas observações y_*; a linha preta representa a média preditiva mais ou menos dois desvios padrões representados pelas linhas diagonais em cinza.



Fonte: Autoria própria.

Na Figura 2 temos o exemplo de um modelo de regressão linear bayesiano, em que $\mathbf{x}$ é unidimensional e podemos facilmente visualizar os valores observados $\mathbf{y}$ em função de $\mathbf{x}$. Os gráficos que constam na Figura 2 são respectivamente: o gráfico de contornos da distribuição a priori (equação (2.9)), modelo ajustado sobre os dados observados e distribuição preditiva sobre novas observações y_*.

2.3 Processos Gaussiano e t-Student

Nesta Seção faremos uma breve introdução aos Processos Gaussianos e Processos t-Student. A compreensão do funcionamento destes modelos será crucial, para posteriormente apresentarmos os modelos de classificação, objetos de estudo deste trabalho, com abordagem de Processos t-Student.

Por serem necessárias propriedades importantes da distribuição Normal Multivariada e distribuição t-Student Multivariada, no desenvolvimento dos modelos que serão trabalhados adiante, as mesmas serão apresentadas a seguir, destacando duas propriedades importantes: marginalização e condicionamento.

Distribuição Normal Multivariada

Seja um vetor aleatório $\mathbf{f} \in \mathbb{R}^n$. Diremos que $\mathbf{f}$ segue uma distribuição Normal Multivariada se sua distribuição conjunta possui a seguinte densidade

$$p(\mathbf{f}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp[-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{f}-\boldsymbol{\mu})], \tag{2.14}$$

em que $\boldsymbol{\mu}$ é um vetor $n \times 1$ e $\boldsymbol{\Sigma}$ é uma matriz de dimensão $n \times n$, positiva semidefinida[4]. A notação normalmente atribuída a esta distribuição é $N_n(\boldsymbol{\mu},\boldsymbol{\Sigma})$.

Sejam $\mathbf{f}_1 \in \mathbb{R}^{n_1}$ e $\mathbf{f}_2 \in \mathbb{R}^{n_2}$ dois vetores aleatórios seguindo conjuntamente uma distribuição Normal Multivariada:

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} \sim N_n\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right), \tag{2.15}$$

em que os vetores $\mathbf{f}_1$ e $\mathbf{f}_2$ possuem n_1 e n_2 entradas respectivamente e $n = n_1 + n_2$. A distribuição marginal de $\mathbf{f}_1$ e sua respectiva distribuição condicional dado $\mathbf{f}_2$ são dados a seguir

$$\mathbf{f}_1 \sim N_{n_1}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_{11}) \quad \text{e} \quad \mathbf{f}_1|\mathbf{f}_2 \sim N_{n_1}(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(x_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}^T). \tag{2.16}$$

Para obtenção de mais detalhes sobre as propriedades da distribuição Normal Multivariada, pode-se consultar Mardia (1979).

[4] A matriz de covariância deve ser positiva definida, mas esta suposição pode ser relaxada para ser simplesmente positiva semidefinida.

Distribuição t-Student Multivariada

Seja um vetor $\mathbf{f} \in \mathbb{R}^n$. Diremos que este vetor segue uma distribuição t-Student Multivariada se sua distribuição conjunta possui a seguinte função densidade

$$p(\mathbf{f}|v,\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{\Gamma((v+n)/2)|\boldsymbol{\Sigma}|^{-1/2}}{v^{1/2}\Gamma(v/2)\pi^{n/2}}\left[1+\frac{1}{v}(\mathbf{f}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{f}-\boldsymbol{\mu})\right]^{-(v+n)/2}, \tag{2.17}$$

em que $\boldsymbol{\mu}$ é um vetor $n \times 1$, $v > 0$ são os graus de liberdade e $\boldsymbol{\Sigma}$ é uma matriz de dimensão $n \times n$, positiva semidefinida. A notação que será utilizada ao longo deste trabalho para denotar esta distribuição é $MVT_n(v,\boldsymbol{\mu},\boldsymbol{\Sigma})$.

Sejam $\mathbf{f}_1 \in \mathbb{R}^{n_1}$ e $\mathbf{f}_2 \in \mathbb{R}^{n_2}$ dois vetores aleatórios seguindo conjuntamente uma distribuição t-Student Multivariada:

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} \sim MVT_n\left(v, \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right). \tag{2.18}$$

A distribuição marginal de $\mathbf{f}_1$ e sua respectiva distribuição condicional dado $\mathbf{f}_2$ são dados a seguir

$$\mathbf{f}_1 \sim MVT_{n_1}(v,\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_{11}) \text{ e} \tag{2.19}$$

$$\mathbf{f}_1|\mathbf{f}_2 \sim MVT_{n_1}(v+n_2, \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{f}_2-\boldsymbol{\mu}_2)+\boldsymbol{\mu}_1, \frac{v+\beta_1-2}{v+n_1-2}\times(\boldsymbol{\Sigma}_{11}-\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{21})), \tag{2.20}$$

em que $\beta_1 = (\mathbf{f}_2-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_2-\boldsymbol{\mu}_2)$. Para mais detalhes e apresentação de outras propriedades da distribuição t-Student Multivariada, pode-se consultar Kibria e Joarder (2006).

2.3.1 Processos Gaussianos

Apresentaremos os Processos Gaussianos (vale igualmente para Processos t-Student) sob a perspectiva da chamada visão espaço de funções, na qual os Processos Gaussianos são vistos como distribuições a priori atribuídas a espaços de funções. Além da visão espaço de funções, há outras formas alternativas de interpretação da qual poderíamos construir os modelos de GP. Há pelo menos duas que poderíamos citar: 1) modelos de GP poderiam ser interpretados como modelos lineares generalizados bayesianos em um conjunto de funções base, a chamada visão espaço de peso (KUSS, 2006); 2) um GP a

priori poderia ser obtido como o limite de uma Rede Neural artificial a priori quando o número de unidades ocultas tende ao infinito (NEAL, 1996). Apesar de haver formas alternativas, a visão espaço de funções é bem mais elegante e possui a vantagem de nos permitir trabalhar diretamente as crenças a priori que temos sobre a função latente f. Ao final desta Seção, apresentaremos de forma sucinta a interpretação de GP através da visão espaço de peso.

Sob a ótica da visão espaço de funções, Processos Gaussianos são Processos Estocásticos. Por isso, a definição sobre Processo Estocástico baseada em Kuss (2006) é dada a seguir

Definição 2 *(Processo Estocástico) Seja* $(\Omega, \mathscr{A}, \mathscr{P})$ *um espaço de probabilidade, e seja* $\boldsymbol{\chi}$ *um conjunto de índices. Um Processo Estocástico é uma função aleatória (ou também pode ser chamado família de variáveis aleatórias)* $f(\mathbf{x}, \boldsymbol{\omega})$ *que pode assumir valores reais ou discretos, e que para todo* $\mathbf{x} \in \boldsymbol{\chi}$ *fixo, torna-se uma função mensurável de* $\boldsymbol{\omega} \in \Omega$.

O conjunto de índices $\boldsymbol{\chi}$ é tomado normalmente para ser $\boldsymbol{\chi} = \mathbb{R}^D$, o espaço de entradas. Da definição acima, temos duas situações possíveis: se $\boldsymbol{\omega} \in \Omega$ for fixado, então $f(\mathbf{x}, \boldsymbol{\omega})$ torna-se uma função não-aleatória de $\mathbf{x}$, chamada de realização ou "*sample path*" do processo. Mas, se em vez disso, fixarmos $\mathbf{x} \in \boldsymbol{\chi}$, $f(\mathbf{x}, \boldsymbol{\omega})$ torna-se uma variável aleatória em Ω (KUSS, 2006). Com base nesse entendimento, um Processo Gaussiano a priori pode ser comprendido como a crença de que a função latente f (desconhecida), assume a forma de uma realização (*sample path*) de um Processo Gaussiano (KUSS, 2006). Procedendo assim, estamos atribuíndo uma distribuição a priori no espaço de funções plausíveis para a modelagem da função subjacente $f(\mathbf{x})$. No decorrer deste trabalho, suprimiremos, sem maiores danos, a indexação em $\boldsymbol{\omega}$.

A função média do processo $f(\mathbf{x})$ é definida por $m(\mathbf{x}) = E[f(\mathbf{x})]$ e a covariância entre duas variáveis aleatórias $f(\mathbf{x})$ e $f(\mathbf{x}')$ correspondente a $\mathbf{x}$ e $\mathbf{x}'$ é dada por

$$cov(f(\mathbf{x}), f(\mathbf{x}')) \quad = \quad E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{2.21}$$

Para definirmos um Processo Estocástico, precisamos caracterizar as distribuições marginais finito-dimensionais dos valores da função $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$, para quaisquer subconjunto de entradas $[\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$. Com isso, temos a seguinte definição de Processo Gaussiano

Definição 3 *(Processo Gaussiano) Um Processo Gaussiano é uma coleção de variáveis aleatórias, em que qualquer conjunto finito delas possui distribuição conjunta Gaussiana Multivariada.*

Da definição, caracterizamos um Processo Gaussiano da seguinte forma

$$\begin{aligned} p(\mathbf{f}|\mathbf{X},\theta) &= N_n(\mathbf{f}|\mathbf{m},\mathbf{K}) \\ &= \frac{1}{(2\pi)^{n/2}|\mathbf{K}|^{1/2}}\exp[-\frac{1}{2}(\mathbf{f}-\mathbf{m})^T\mathbf{K}^{-1}(\mathbf{f}-\mathbf{m})], \end{aligned} \tag{2.22}$$

em que $|.|$ é o determinante da matriz de covariância, $\mathbf{m}: \boldsymbol{\chi} \longrightarrow \mathbb{R}$ é a função média do GP com $\mathbf{m} = [m(\mathbf{x}_1),\ldots,m(\mathbf{x}_n)]^T$ e $\mathbf{K}$ é uma matriz de covariância construída a partir de uma função de covariância $k: \boldsymbol{\chi} \times \boldsymbol{\chi} \longrightarrow \mathbb{R}$, com $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. Usualmente, vamos escrever Processos Gaussianos como

$$f(\mathbf{x}) \sim \mathscr{GP}(m(\mathbf{x}),k(\mathbf{x},\mathbf{x}')). \tag{2.23}$$

Kuss (2006) alerta que a função média pode ser escolhida de forma arbitrária, mas a função de covariância deve atender aos requisitos: ser positiva definida e simétrica. O primeiro requisito é essencial para permitir a existência de todas as distribuições finito-dimensionais, mas ainda assim, esta restrição pode ser relaxada para semi-definida positiva. Uma função de covariância é semidefinida positiva se tal matriz, $n \times n$, para qualquer escolha de $\mathbf{v} \in \mathbb{R}^n$, exceto o vetor nulo, satisfaz a condição

$$\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0. \tag{2.24}$$

A função $m(\mathbf{x})$ especifica o valor médio da função em um ponto fixo $\mathbf{x}$. E no presente trabalho, quando apresentarmos os modelos de regressão de GP, tomaremos como distribuição a priori um Processo Gaussiano no qual a função média é o vetor nulo, apesar disso não ser uma regra. Esta forma de distribuição a priori revela a crença que os valores de $\mathbf{f}$ podem ser tanto positivos quanto negativos em qualquer uma de suas entradas. A função de covariância especifica o grau de dependência entre dois pontos de dados no espaço de entradas, e também é conhecida por função *kernel*. É uma medida de similaridade, e é usada como uma função de mapeamento para obtenção de uma relação direta existente entre os dados (KOCIJAN, 2016). Esta forma de interpretar a função de covariância é bem útil para os problemas atuais de aprendizagem com GP, envolvendo desde problemas de regressão quanto de classificação, e veremos mais adiante que ela controla vários aspectos importantes da função subjacente $f(\mathbf{x})$.

A atribuição de um Processo Gaussiano a priori sobre o espaço de todas as funções candidatas para modelar $f(\mathbf{x})$, carrega crenças iniciais que temos sobre a relação que existe entre os dados. Estas crenças são codificadas pela função de covariância, que determina importantes propriedades dos Processos Gaussianos. A suavidade dos processos, por exemplo, está intimimante relacionada a propriedades de continuidade e diferenciabilidade, que por sua vez, só podem ser testificadas através da função de covariância (ADLER, 2010; ABRAHAMSEN, 1997). Muitas formas de comportamento podem ser capturadas pela função de covariância, sendo uma das características mais atrativas da abordagem por Processos Gaussianos.

Regressão com Processo Gaussiano

Definido o que é um Processo Gaussiano, podemos introduzir os modelos de regressão não-paramétricos utilizando os Processos Gaussiano e t-Student. O objetivo consiste em tentar modelar a distribuição condicional $p(y|\mathbf{x})$. Para tanto, podemos decompor esta distribuição em uma parte chamada de sitemática, e outra, que chamaremos de aleatória (KUSS, 2006). A parte sistemática será definida por uma função latente $f : \boldsymbol{\chi} \longrightarrow \mathbb{R}$, tal que o modelo é dado por

$$p(y|f(\mathbf{x}), \boldsymbol{\theta}), \tag{2.25}$$

em que $\boldsymbol{\theta}$ é conjunto de parâmetros presente na função de verossimilhança.

Os dados para a construção do modelo serão coletados aos pares em um conjunto incial, chamado conjunto de treinamento $\mathscr{D}$ de n observações $\mathscr{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, em que $\mathbf{x}_i \in \mathbb{R}^D$ são as entradas do modelo (covariáveis) e $\mathbf{y}$ as saídas do modelo (variável resposta). E neste caso, $\boldsymbol{\chi} = \mathbb{R}^D$ é o conjunto de todas as possíveis entradas. A matriz de delineamento $\mathbf{X}$ com dimensão $D \times n$ é composta pelos vetores colunas das entradas $\mathbf{x}_i$, $i = 1, \ldots, n$, e as saídas escalares y_i compõem o vetor $\mathbf{y}$, tal que, podemos escrever $\mathscr{D} = (\mathbf{X}, \mathbf{y})$. O conjunto de teste pode ser escrito da seguinte forma: $\mathscr{D}_* = \{(\mathbf{x}_{i*}, y_{i*})|i_* = 1, \ldots, n_*\}$.

O modelo na sua forma decomposta, como é comumente característico dos modelos de regressão, é dado a seguir

$$y_i = f(\mathbf{x}_i) + \varepsilon_i \quad \forall \quad i = 1, \ldots, n, \tag{2.26}$$

em que $\boldsymbol{\varepsilon}$ é a parte ruidosa do modelo. Se considerarmos que os valores ruidosos ε_i são aditivos, independentes e identicamente distribuídos, de média 0 e variância σ^2, ou seja,

$\varepsilon_i \sim N(0,\sigma^2)\ \forall i = 1,\ldots,n$, teremos a seguinte função de verossimilhança

$$p(\mathbf{y}|\mathbf{f},\boldsymbol{\theta}) = N_n(\mathbf{y}|\mathbf{f},\sigma^2\mathbf{I}) = \prod_{i=1}^{n} N(y_i|f(\mathbf{x}_i),\sigma^2), \tag{2.27}$$

em que $\mathbf{I}$ é a mariz identidade $n \times n$, e $\boldsymbol{\theta} = \sigma^2$. Dado o modelo (2.27), para fazer inferências sobre f, precisamos formalizar uma distribuição a priori sobre a função latente. Assim, vamos atribuir um GP como distribuição a priori ao conjunto finito de valores $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)]^T$. Cada valor deste vetor corresponde a uma variável aleatória $f(\mathbf{x})$ referente a uma entrada $\mathbf{x}$. Com isso, um GP a priori significa que a ditribuição conjunta de $\mathbf{f}$ é Normal Multivariada (da definição 3). Prosseguindo, especificando uma função média $\mathbf{m}$ e uma função de covariância $k(\mathbf{x},\mathbf{x}',\boldsymbol{\psi}) : \boldsymbol{\chi} \times \boldsymbol{\chi} \longrightarrow \mathbb{R}$, e $\mathbf{K} = k(\mathbf{x},\mathbf{x}',\boldsymbol{\psi})$, temos

$$p(\mathbf{f}|\mathbf{X},\boldsymbol{\psi}) \sim \mathscr{GP}(\mathbf{m},\mathbf{K}), \tag{2.28}$$

em que $\boldsymbol{\psi}$ são os hiper-parâmetros presentes na função de covariância.

De posse da distribuição a priori e da função de verossimilhança, o teorema de Bayes é a regra de atualização do conhecimento a priori para obtermos a distribuição a posteriori sobre $\mathbf{f}$. Dado o conjunto de dados de treinamento $\mathscr{D}$, a distribuição a posteriori é da seguinte forma

$$\begin{aligned} p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi}) &= \frac{p(\mathbf{y}|\mathbf{f},\boldsymbol{\theta})p(\mathbf{f}|\mathbf{X},\boldsymbol{\psi})}{p(\mathscr{D}|\boldsymbol{\theta},\boldsymbol{\psi})} \\ &\propto N_n(\mathbf{y}|\mathbf{f},\sigma^2)N_n(\mathbf{f}|\mathbf{0},\mathbf{K}) \\ &= N_n(\mathbf{f}|\mathbf{K}(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y},(\mathbf{K}^{-1}+\sigma^2\mathbf{I})^{-1}). \end{aligned} \tag{2.29}$$

O termo do meio foi escrito na forma de uma proporcional pelo fato de $p(\mathscr{D}|\boldsymbol{\theta},\boldsymbol{\psi})$, a verossimilhança marginal, não depender de $\mathbf{f}$, e nesse caso ela faz apenas o papel de uma constante de normalização. A verossimilhança marginal (também chamada de evidência), no contexto de estimação dos hiperparâmetros da função de covariância (maximização da evidência (MACKAY, 1992)), desempenha papel fundamental na seleção de modelos, e a mesma pode ser obtida analiticamente devido às propriedades de marginalização da distribuição Gaussiana

$$\begin{aligned} p(\mathscr{D}|\boldsymbol{\theta},\boldsymbol{\psi}) &= \int_{\mathbf{f}} p(\mathbf{y}|\mathbf{f},\boldsymbol{\theta})p(\mathbf{f}|\mathbf{X},\boldsymbol{\psi})d\mathbf{f} \\ &= \int_{\mathbf{f}} N_n(\mathbf{y}|\mathbf{f},\sigma^2\mathbf{I})N_n(\mathbf{f}|\mathbf{0},\mathbf{K})d\mathbf{f} \\ &= N_n(\mathbf{y}|\mathbf{0},\mathbf{K}+\sigma^2\mathbf{I}). \end{aligned} \tag{2.30}$$

Dada uma nova entrada $\mathbf{x}_* \in \mathbb{R}^D$, a distribuição preditiva relacionada a $f_* = f(\mathbf{x}_*) \in \mathbb{R}$ pode ser calculada de maneira conveniente através das propriedades tanto de condicionamento quanto de marginalização da distribuição Gaussiana. A distribuição conjunta das saídas observadas $\mathbf{y}$ e do valor da função f_* de teste é dada a seguir

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim N_{n+1}\left(\mathbf{0}\ , \begin{bmatrix} \mathbf{K}+\sigma^2 I & \mathbf{K}_* \\ \mathbf{K}_*^T & K_{**} \end{bmatrix} \right), \tag{2.31}$$

em que $\mathbf{K}_* = [k(\mathbf{x}_*,\mathbf{x}_1),\ldots,k(\mathbf{x}_*,\mathbf{x}_n)]^T$ é a função de covariância de dimensão $(n \times 1)$, calculada para a entradas de treinamento e de teste, e $K_{**} = k(\mathbf{x}_*,\mathbf{x}_*)$ é o valor da função de covariância referente somente à entrada de teste.

Para obter a distribuição preditiva a posteriori $p(f_*|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})$, precisamos restringir a distribuição conjunta em (2.31) a conter apenas as funções que estão de acordo com os dados observados. Podemos alcançar isso de maneira simples, em termos probabilísticos, condicionando a distribuição de f_* ao dados observados $\mathbf{y}$, utilizando a propriedade de condicionamento da distribuição Normal Multivariada apresentada anteriormente. Este condicionamento dispensa a necessidade de usar mais uma vez o teorema de Bayes, e pela propriedade de condicionamento temos o seguinte resultado

$$\begin{aligned} p(f_*|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi}) &\sim N(\bar{f}_*, V(f_*)), \qquad (2.32) \\ \text{em que} & \\ \bar{f}_* &= \mathbf{K}_*^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} \\ V(f_*) &= K_{**} - \mathbf{K}_*^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{K}_*. \end{aligned}$$

Para obtermos a distribuição preditiva para $y_* \in \mathbb{R}$, basta adicionarmos σ^2 a $V(f_*)$ no modelo (2.32). O modelo (2.32) tem a vantagem de possibilitar predições dadas completamente por uma distribuição de probabilidade definida, e não apenas por uma simples estimação pontual (MATTOS, 2017). Percebemos que a média da distribuição preditiva é uma combinação linear das saídas $\mathbf{y}$, e a variância preditiva não possui dependência em $\mathbf{y}$. Uma estimativa plausível para f_*, considerando a teoria das decisões (veja Williams e Rasmussen (2006)), seria dada por $\bar{f}_*$ no modelo (2.32).

O resulatdo obtido em (2.32) poderia também ser alcançado de forma alternativa, procedendo com marginalização da variável latente f_* na distribuição conjunta a posteriori $p(f_*,\mathbf{f}|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})$. Pelo fato da função de verossimilhança e da posteriori

$p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})$ serem de natureza Gaussiana, a integral abaixo é tratável analiticamente e $p(f_*|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})$ pode ser obtida de forma direta

$$\begin{aligned} p(f_*|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi}) &= \int_{\mathbf{f}} p(f_*,\mathbf{f}|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})d\mathbf{f} \\ &= \int_{\mathbf{f}} p(f_*|\mathbf{f},\mathbf{X},\mathbf{x}_*,\boldsymbol{\psi})p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta},\boldsymbol{\psi})d\mathbf{f} \\ &= N(f_*|\mathbf{K}_*^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y}, K_{**}-\mathbf{K}_*^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{K}_*). \end{aligned} \quad (2.33)$$

Algumas observações interessantes podem ser feitas sobre a modelagem com Processo Gaussiano. Na Figura 3 encontra-se a ilustração de uma modelagem de regressão com GP. A primeira observação que podemos fazer através desta Figura, é que os y_i são condicionalmente independentes entre si (representados pelos nós do gráfico), dada a função latente $f_i = f(\mathbf{x}_i)$ e o ruído $\boldsymbol{\varepsilon}_i$ para $i = 1,\ldots,n$, justificando a função de verossimilhança ser dada por $\prod_{i=1}^n N(y_i|f_i,\sigma^2)$ (MATTOS, 2017). A segunda observação, é que a barra grossa que conecta os valores da função f_i reflete a dependência que há entre os valores da função, como era de se esperar, como consequência da definição da matriz de covariância $\mathbf{K}$.

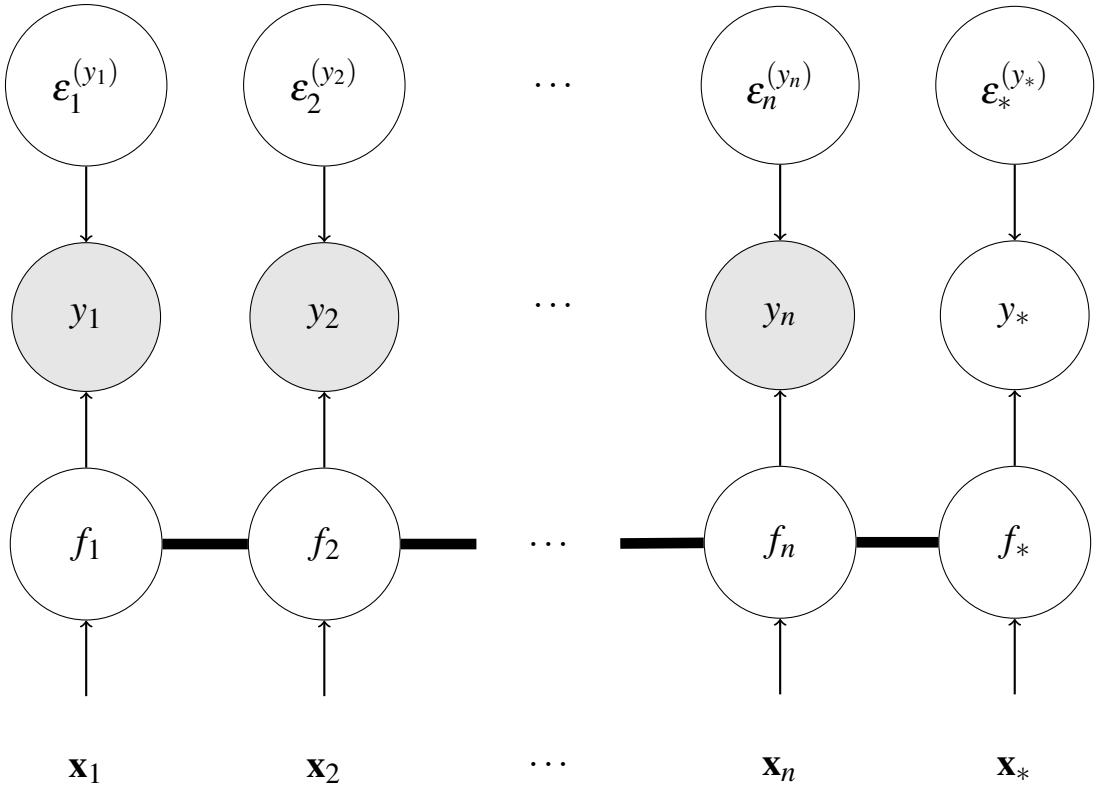
Vale notar que o modelo de Regressão de Processo Gaussiano (GPR) tem a vantagem de possuir formas fechadas em todos os seus resultados (distribuição a posteriori, distribuição preditiva, e verossimilhança marginal), o que permite que todas as inferências do modelo sejam feitas de forma exata, sem recorrer a necessidade de métodos de aproximação para executar tal tarefa. Na Figura 4 temos o exemplo de um modelo de regressão com GP, $y = f(\mathbf{x}) + \boldsymbol{\varepsilon}$, em que a função latente verdadeira é $f(x) = \sin(x)/x$ e $\boldsymbol{\varepsilon} \sim N_n(0, 0.1)$.

Interpretação alternativa de priori de processo Gaussiano

Como falamos no ínicio desta Seção, há formas alternativas de interpretação de um GP a priori. A forma que descreveremos a seguir é a conhecida visão espaço de peso, e surge naturalmente como uma generalização do modelo linear bayesiano apresentado na Seção 2.2. A apresentação dessa visão segue a forma abordada em Kuss (2006) e Williams e Rasmussen (2006).

O modelo linear bayesiano dado por (2.5), possui expressividade limitada, se adequando apenas à situações onde os dados apresentam um comportamento com tendência linear. Uma forma de contornar isso seria primeiro projetar as entradas em algum espaço alto-dimensional, também chamado de espaço de características usando um conjunto de

Figura 3 – Ilustração de modelagem com Processo Gaussiano mostrando a relação existente entre as variáveis. Os nós preenchidos representam as observações de saída do modelo. Os nós brancos representam as variáveis latentes (não observadas). Ilustração baseada em MATTOS (2017).



Fonte: Autoria própria.

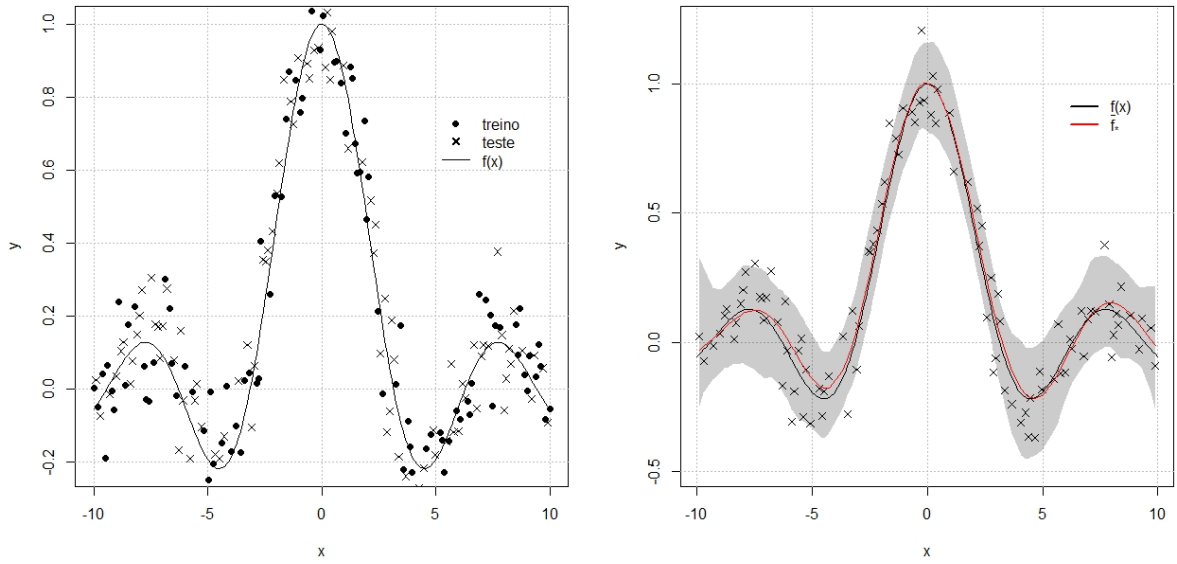
funções bases, e então aplicar o modelo linear neste novo espaço. O adjetivo linear ainda é válido, pois o modelo continuará sendo linear nos parâmetros. Com isso, a forma de $f(\mathbf{x})$ é dada por

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}, \tag{2.34}$$

em que $\boldsymbol{\phi} = [\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x})]^T$ são os valores de N funções base. Um exemplo desse tipo de modelo são os modelos com ajuste polinomial, em que $\boldsymbol{\phi} = [1, x^1, x^2, x^3, \ldots]^T$ são funções potências de x. Atribuíndo uma distribuição Normal Multivariada $N_N(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{w})$ aos parâmetros $\mathbf{w}$ (pesos) do modelo, estamos implicitamente especificando uma distribuição a priori sobre funções, uma vez que para cada amostra $\mathbf{w}$ retirada da priori, teremos uma função f correspondente.

Dessa forma $f(\mathbf{x})$, por ser uma combinação linear dos parâmetros $\mathbf{w}$ (expressão 2.34) para cada entrada $\mathbf{x}$, é uma variável aleatória com distribuição Normal Multivariada. Vamos agora determinar o vetor média de $f(\mathbf{x})$ e sua respectiva matriz de covariância.

Figura 4 – GPR com ruído normal. A função verdadeira é $f(x) = \sin(x)/x$, e $\sigma^2 = 0.1$; pontos preenchidos: dados de treino; pontos em "x": dados de teste. Gráfico à esquerda: valores observados de $\mathbf{y}$; a linha contínua representa a função verdadeira $f(\mathbf{x})$. Gráfico à direita: distribuição preditiva dos valores de teste; linha contínua vermelha representa a média preditiva $\bar{f}_*$ em (2.32); linha contínua preta representa a função latente $f(\mathbf{x})$; área hachurada em cinza representa a área de credibilidade de 95%. A função de covariância utilizada foi a exponencial quadrática $k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp(-0.5(\mathbf{x}-\mathbf{x}')^2/l^2)$



Fonte: Autoria própria.

Para a média de $f(\mathbf{x})$, note que a média da priori em $\mathbf{w}$ é zero, assim temos

$$E[f(\mathbf{x})] = \boldsymbol{\phi}^T E[\mathbf{w}] = \mathbf{0}, \tag{2.35}$$

e para a matriz de covariância temos,

$$\text{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \boldsymbol{\phi}^T E[\mathbf{w}^T \mathbf{w}] \boldsymbol{\phi} = \boldsymbol{\phi}^T \boldsymbol{\Sigma}_\mathbf{w} \boldsymbol{\phi}. \tag{2.36}$$

Portanto, temos que a distribuição de $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$ segue uma distribuição Normal Multivariada com média sendo o vetor nulo e função de covariância dada por (2.36). Assim, usando a perspectiva espaço de peso, os GP são equivalentes a assumir um modelo linear em um conjunto de funções base, utilizando uma distribuição Normal multivariada para $\mathbf{w}$.

2.3.2 Processos t-Student

Os Processos t-Student (TP) surgiram como uma alternativa aos Processos Gaussianos na atribuição de distribuição a priori não-paramétrica sobre funções (SHAH *et*

al., 2014). A proposta surgiu principalmente para a construção de modelos que fossem robustos a *outliers*. Os modelos de GP podem ser sensivelmente prejudicados se houver a presença de *outliers* nos dados, uma vez que a distribuição Gaussiana pertence à chamada classe de distribuições de "cauda curta". Em outras palavras, para estas distribuições há uma forte restrição que os pontos variem muito longe da média. No entanto, a distribuição t-Student pode ajudar a superar esta dificuldade, pois esta tem "caudas pesadas", o que consequentemente, permite que os dados possam variar um pouco mais afastados da média.

No referido trabalho de Shah *et al.* (2014), os autores mostram que os processos t-Student conseguem reter propriedades importantes dos GP (algumas destas, melhoradas) sem a necessidade de maiores custos computacionais adicionais, como boa interpretabilidade, flexibilidade não-paramétrica e fácil seleção de modelos através dos hiperparâmetros da função de covariância.

A ideia básica para obtenção de um Processo t-Student, consiste em propor uma abordagem totalmente não-paramétrica para um modelo de Processo Gaussiano. Isto é alcançado construindo um modelo hierárquico, onde atribuímos para a função de covariância do Processo Gaussiano uma distribuição a priori que reflita a incerteza que se tem sobre a forma paramétrica da mesma. Normalmente em aplicações, a função de covariância do Processo Gaussiano é escolhida de antemão, possuindo uma forma paramétrica estabelecida, representando suposições dos comportamentos das funções sob um GP, restando-nos apenas proceder com a estimação dos parâmetros (hiperparâmetros) por algum algorítimo de estimação.

A distribuição Wishart Inversa é uma escolha atrativa de distribuição a priori para uma matriz de covariância sobre $\mathbf{\Pi}(n)$, o conjunto de todas as matrizes reais $n \times n$, definidas positivas e simétricas. Para obtenção de mais informações e propriedades da distribuição Wishart Inversa, consulte Nydick (2012). A seguir, adotamos o mesmo roteiro abordado por Shah *et al.* (2014) para apresentar os Processos t-Student. Vale ressaltar que existem outras formas, das quais, apresentaremos brevemente algumas. Primeiramente, apresentaremos a função densidade da distribuição Wishart Inversa.

Definição 4 *(Distribuição Wishart Inversa) Seja* $\mathbf{\Sigma}$ *uma variável aleatória* $\in \mathbf{\Pi}(n)$ *seguindo uma distribuição Wishart Inversa com parâmetros* $\nu \in \mathbb{R}_+$, $\mathbf{K} \in \mathbf{\Pi}(n)$ *e escreveremos*

$\boldsymbol{\Sigma} \sim IW_n(v, \mathbf{K})$ *se sua densidade for dada por*

$$p(\boldsymbol{\Sigma}) = c_n(v, \mathbf{K})|\boldsymbol{\Sigma}|^{-(v+2n)/2} exp\left(-\frac{1}{2} Tr(\mathbf{K}\boldsymbol{\Sigma}^{-1})\right), \tag{2.37}$$

com $c_n(v, \mathbf{K}) = \frac{|\mathbf{K}|^{(v+n-1)/2}}{2^{(v+n-1)n/2}\Gamma_n((v+n-1)/2)}$.

Em (2.37), $Tr(\mathbf{K}\boldsymbol{\Sigma}^{-1})$ refere-se ao traço da matriz $\mathbf{K}\boldsymbol{\Sigma}^{-1}$. Dawid (1981) mostra que a distribuição Wishart Inversa definida dessa forma é consistente sob marginalização, ou seja, se $\boldsymbol{\Sigma} \sim IW_n(v, \mathbf{K})$, então qualquer submatriz $\boldsymbol{\Sigma}_{11}$ de tamanho $n_1 \times n_1$ terá distribuição $IW_{n1}(v, \mathbf{K}_{11})$. Outra propriedade importante na distribuição Wishart Inversa está no fato do parâmetro v não depender necessariamente do tamanho da matriz. Estas propriedades são fundamentais para podermos definir um processo cujas distribuições marginais possuam distribuição Wishart Inversa de tamanho arbitrário Shah *et al.* (2014). Definimos um processo Wishart Inversa da seguinte forma

Definição 5 *(Processo Wishart Inversa): Seja* $\boldsymbol{\chi}$ *algum espaço de entradas e* $\mathbf{K} : \boldsymbol{\chi} \times \boldsymbol{\chi} \to \mathbb{R}$ *uma matriz de covariância positiva definida.* $\boldsymbol{\Sigma}$ *é um processo Wishart Inversa em* $\boldsymbol{\chi}$ *com parâmetros* $v \in \mathbb{R}_+$ *e matriz de covariância* $\boldsymbol{K}$ *se qualquer coleção finita* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \boldsymbol{\chi}$, $\boldsymbol{\Sigma}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \sim IW_n(v, \mathbf{K})$, *em que* $\mathbf{K} \in \boldsymbol{\Pi}(n)$ *com* $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. *Denotamos este processo por* $\boldsymbol{\Sigma} \sim \mathscr{IWP}(v, \mathbf{K})$.

Definido o processo Wishart Inversa, usaremos este processo como distribuição a priori sobre matrizes de covariância. Dada uma matriz de covariância $\mathbf{K}$, parametrizada por $\boldsymbol{\theta}$, e uma função média m, temos o seguinte modelo de Processo Gaussiano hierárquico

$$\begin{aligned} f|\boldsymbol{\Sigma} &\sim \mathscr{GP}(m, (v-2)\boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &\sim \mathscr{IWP}(v, \mathbf{K}). \end{aligned} \tag{2.38}$$

Podemos obter o Processo t-Student, marginalizando f na integral do modelo (2.38). A distribuição marginal $p(f|v, \mathbf{K})$ pode ser obtida analiticamente devido ao fato de haver conjugação entre a distribuição Wishart Inversa com a matriz de covariância de uma distribuição Gaussiana Multivariada. Assim, dado o vetor de valores $\mathbf{f}=[f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$, e a função média $m = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)]^T$, a distribuição marginal $p(f|v, \mathbf{K})$ é obtida da

seguinte forma

$$\begin{aligned} p(f|v,\mathbf{K}) &= \int p(f|\mathbf{\Sigma})p(\mathbf{\Sigma}|v,\mathbf{K})d\mathbf{\Sigma} \\ &\propto \frac{\exp\left(-\frac{1}{2}Tr\left(\left(\mathbf{K}+\frac{(\mathbf{f}-\mathbf{m})(\mathbf{f}-\mathbf{m})^T}{(v-2)}\right)\mathbf{\Sigma}^{-1}\right)\right)}{|\mathbf{\Sigma}|^{(v+2n+1)/2}}d\mathbf{\Sigma} \\ &\propto \left(1+\frac{1}{v-2}(\mathbf{f}-\mathbf{m})\mathbf{K}^{-1}(\mathbf{f}-\mathbf{m})^T\right)^{-(v+n)/2}. \end{aligned} \tag{2.39}$$

Esta última expressão é proporcional ao núcleo de uma distribuição t-Student Multivariada. Deste modo, como a distribuição marginal das variáveis latentes $\mathbf{f}$ possui uma distribuição t-Student Multivariada, podemos utilizar uma definição similar a que se encontra na definição 3, para definimos um Processo t-Student. Em Shah *et al.* (2014), no Lema 1, os autores mostram que a distribuição t-Student é consistente sob marginalização, logo, podemos dar a seguinte definição para Processos t-Student

Definição 6 *(Processo t-Student):* $\mathbf{f}$ *é um Processo t-Student em* $\boldsymbol{\chi}$*, com parâmetro* $v>2$*, função média* $m:\boldsymbol{\chi}\to\mathbb{R}$ *e matriz de covariância* $\mathbf{K}:\boldsymbol{\chi}\times\boldsymbol{\chi}\to\mathbb{R}$ *se qualquer coleção finita de valores da função possui distribuição conjunta t-Student Multivariada:* $[f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n)]\sim MVT_n(v,m,\mathbf{K})$ *em que* $\mathbf{K}\in\mathbf{\Pi}(n)$ *com* $K_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$ *e* $m\in\mathbb{R}^n$*. Denotaremos este processo por* $f\sim\mathscr{TP}(v,m,\mathbf{K})$*.*

Além desta possibilidade de obtenção do Processo t-Student, há outras formas existentes na literatura. Em Williams e Rasmussen (2006), há a descrição de uma forma alternativa (O'HAGAN, 1991; O'HAGAN *et al.*, 1999), aplicando uma mistura de escala em uma Normal Multivariada, onde divide-se a matriz de covariância $\mathbf{K}$ por um escalar τ, e atribuímos a este escalar uma distribuição a priori Gama. O modelo referente a este método é dado a seguir

$$\tilde{k}(\mathbf{x},\mathbf{x}')=\tau^{-1}k(\mathbf{x},\mathbf{x}'),\quad p(\tau|\alpha,\beta)=\frac{\alpha^\alpha}{\beta^\alpha\Gamma(\alpha)}\tau^{\alpha-1}\exp\left(-\frac{\tau\alpha}{\beta}\right), \tag{2.40}$$

em que a distribuição a priori $p(\tau|\alpha,\beta)$ possui parametrização, tal que, $E(\tau)=\beta$. Sendo $\mathbf{f}$, n valores da função subjacente, e assumindo um Processo Gaussiano, considere que o processo latente seja livre de ruído, ou seja, modelos que são dados pela lei $y=f(\mathbf{x})$. Assim, o Processo t-Student pode ser obtido encontrando a distribuição marginal de $p(\mathbf{f}|\alpha,\beta,\mathbf{K})$,

de forma semelhante ao procedimento executado no modelo (2.38), da seguinte forma

$$\begin{aligned} p(\mathbf{f}|\alpha,\beta,\mathbf{K}) &= \int_{\tau} N_n(\mathbf{f}|\mathbf{0},\tau^{-1}\mathbf{K})p(\tau|\alpha,\beta)d\tau \\ &= \frac{\Gamma(\alpha+n/2)(2\pi\alpha)^{-n/2}}{\Gamma(\alpha)|\beta^{-1}\mathbf{K}|^{-1/2}}\left[1+\frac{\beta\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}}{2\alpha}\right]^{-(\alpha+n/2)}, \end{aligned} \tag{2.41}$$

que resulta em uma distribuição t-Student Multivariada: $f \sim MTV_n(2\alpha,\mathbf{0},\beta^{-1}\mathbf{K})$. O modelo dado em (2.40), como mencionado, não se aplica a funções ruidosas, pois os autores argumentam que inserir uma contribuição ruidosa no modelo o tornaria muito complexo, pois a soma de uma distribuição t-Student com outra t-Student, ou até mesmo com uma distribuição Gaussiana, conduziria o modelo a ser analiticamente intratável. A proposta de Shah *et al.* (2014) no entanto, quando os mesmos apresentam o modelo de regressão de TP (modelo Regressão de Processo t-Student com ruído dentro do *kernel* (TPRK)), evita esta debilidade incorporando o ruído dentro da função de covariância. Esta abordagem não é igual a inserir o ruído de forma independente no modelo, como fizemos no caso dos modelos de regressão com Processo Gaussiano (GPR), onde ε_i era considerado i.i.d, pois o ruído apesar de não ser correlacionado com a função latente, se torna dependente.

Prosseguindo, podemos ainda citar um outro modelo hierárquico para obtenção do Processo t-Student. Neste modelo, obtemos o mesmo Processo t-Student marginal, atribuindo uma distribuição Gama sobre um escalar. Este modelo com a distribuição a priori Gama, acima referido, pode ser obtido da seguinte forma (SHAH *et al.*, 2014)

$$\begin{aligned} \mathbf{f}|r &\sim N(\mathbf{m},r(v-2)\mathbf{K}/\rho) \\ r^{-1} &\sim \Gamma(v/2,\rho/2), \end{aligned} \tag{2.42}$$

em que a marginalização da distribuição de $\mathbf{f}$ no modelo acima, resulta em uma distribuição t-Student Multivariada $\mathbf{f} \sim MVT_n(v,\mathbf{m},\mathbf{K})$. Os autores argumentam que este resultado é muito importante, pois pode-se obter um processo t-Student à semelhança do modelo (2.38), integrando apenas sobre um escalar, em vez de um objeto de dimensão infinita.

Robustez dos Processos t-Student

Falamos no início da Seção que Processos t-Student são uma alternativa robusta a GP. Vanhatalo *et al.* (2009. Tema: Advances in neural information processing systems) apresentam uma definição de robustez em termos de modelo robusto a *outlier* (com relação à distribuição a posterior de $\mathbf{f}$) da forma: um modelo é definido para ser robusto a *outlier*

de ordem n, se $p(f|y_1,\ldots,y_{n+1}) \longrightarrow p(f|y_1,\ldots,y_n)$ quando $y_{n+1} \longrightarrow \infty$ (O'HAGAN, 1978; WEST, 1984). Isto quer dizer que uma observação discrepante torna-se assintoticamente desprezível para a posteriori quando esta observação tende ao infinito. Este é o caso da distribuição t-Student. Por exemplo, a distribuição t-Student univariada pode rejeitar até m *outliers* se há no mínimo $2m$ observações no total (VANHATALO *et al.*, 2009. Tema: Advances in neural information processing systems; TANG *et al.*, 2017). Dessa forma, espera-se que o TP seja mais robusto a *outliers* nas variáveis de entrada.

Andrade () em seu trabalho, fornece provas matemáticas para a robustez dos TP utilizando o contexto de regressão não-linear, por meio da Teoria da Variação Regular (RV). No referido trabalho, o autor mostra como a distribuição a posteriori do TP comporta-se sob vários tipos de *outliers* (*outliers* nas variáveis de entrada, saída, e entrada-saída), fornecendo a distribuição a posteriori limite para cada tipo de *outlier* quando estes tendem ao infinito.

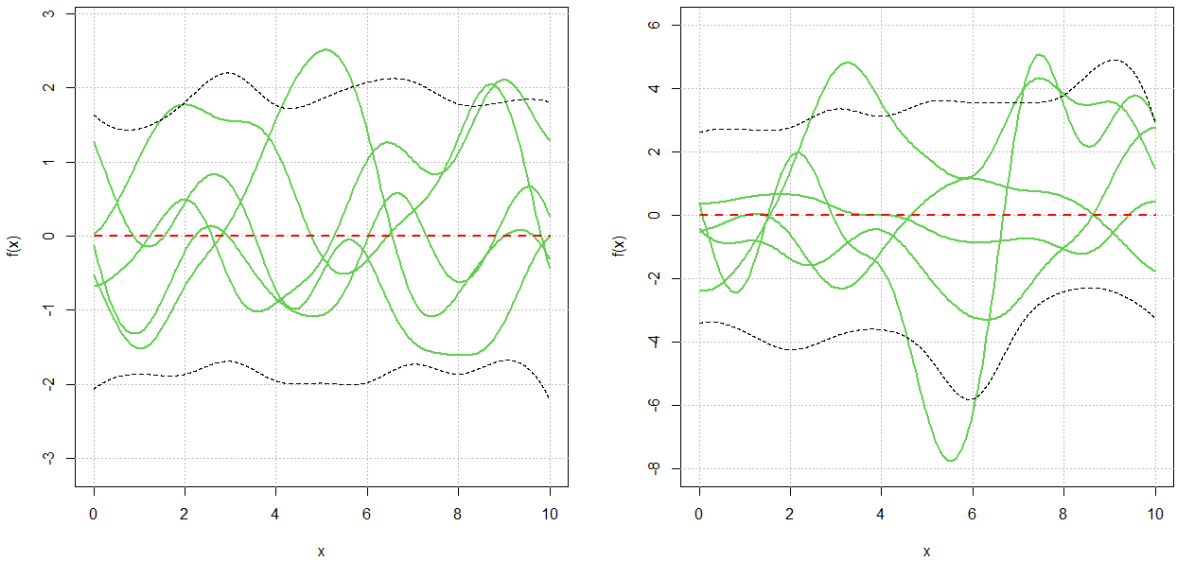
Na Figura 5 temos um exemplo como ilustração da questão da robustez. No gráfico do lado esquerdo da Figura 5, temos as amostras de um GP (linhas verdes), onde percebemos que as mesmas variam muito próximas à média (linha vermelha). No gráfico do lado direito, temos as amostras de um TP cuja variação é bem mais distante do valor médio, como podemos constatar pela abrangência do intervalo de credibilidade, indicando que *outliers* (valores usualmente distantes da média) não têm muito efeito na média de um TP. Tang *et al.* (2017) argumentam que, devido ao fato de normalmente fazermos predições em aplicações práticas com a média preditiva, TPs são esperados para serem mais robustos a *outliers* que GPs.

Para finalizar esta apresentação sobre Processos t-Student, cabe mencionar que os mesmos apresentam-se como uma generalização dos Processos Gaussianos. O parâmetro ν do Processo t-Student controla o peso da cauda, sendo que valores pequenos de ν dão origem a processos com comportamento de cauda pesada, e consequentemente, para valores maiores de ν, os processos resultantes apresentam comportamento de cauda mais leve. Dessa forma, pode-se provar (Shah *et al.* (2014) apresenta a prova deste resultado) que, quando $\nu \to \infty$, o Processo t-Student converge em distribuição para um GP.

Regressão com Processo t-Student

O modelo de regressão apresentado na Seção anterior dispõe de muitas atratitividades, tais como: fácil implementação, expressões a posteriori e preditiva obtidas

Figura 5 – Comparação entre os Processos Gaussianos e t-Student, ambos possuem em comum função média, hiperparâmetros e função de covariância. Gráfico à esquerda: GP. Gráfico à direita: TP. Os graus de liberdade ν do TP é 2. Linha vermelha tracejada representa a função média. Linhas verdes representam as amostras geradas dos processos. Linhas tracejadas representam o intervalo de 95% de credibilidade.



Fonte: Autoria própria.

de forma analítica, boas predições etc. Este modelo tem como suposições que as saídas $\mathbf{y} \in \mathbb{R}^n$ apresentam verossimilhança Gaussiana, e consequentemente, o Processo Gaussiano resultante é dado por expressões analiticamente tratáveis.

Entretanto, há cenários que a modelagem das observações é contaminada com algum tipo de ruído não Gaussiano, caracterizando os chamados *outliers*. Estes, de uma forma bastante resumida, podem ser definidos como pontos de dados que são bem diferentes dos demais (AGGARWAL, 2013). Uma das consequências em se desconsiderar os *outliers* presentes na amostra, seria a má estimação dos hiperparâmetros durante a fase de treinamentos, afetando de forma direta as predições do modelo (MATTOS, 2017).

Os Processos Gaussianos são afetados significativamente pela presença de *outliers* na amostra por serem processos de cauda leve, não permitindo que os valores da função latente $\mathbf{f} \in \mathbb{R}^n$ se afastem muito da média. Em contraste, distribuições de cauda pesada permitem que os valores se afastem muito mais da média, e dessa forma, estas distribuições se tornam robustas a valores extremos. Entre os modelos de cauda pesada mais preferíveis para a construção de modelos robustos, está a distribuição t-Student,

apesar de outras distribuições como modelo de misturas de Gaussianas e modelo Laplace, também serem alternativas atraentes (KUSS, 2006). Vale ressaltar que modelos com verossimilhança não Gaussiana, utilizando GP, não são analiticamente tratáveis e métodos aproximados devem ser usados para fazer inferências, tais como *Markov Chain Monte Carlo*, Bayes variacional (VB), Monte Carlo Sequencial (SMC) entre outros.

Trabalhos como de Neal (1997), Jylanki *et al.* (2011) e Berger e Rauscher (2012) apresentam modelos robustos utilizando ruído t-Student. Tang *et al.* (2017) apresentam um modelo de regressão robusto utilizando Processo t-Student junto com ruído t-Student, em que as inferências para este modelo foram obtidas utilizando aproximação de Laplace. A proposta deste tipo de modelo é lidar com *outliers* tanto na variável resposta quanto nas variáveis regressoras ao mesmo tempo.

Apresentaremos adiante dois tipos de modelos de regressão. O primeiro, apesar de não tratar de modelos de TP diretamente, refere-se a uma Regressão de Processo Gaussiano com ruído t-Student (GPRT), exemplificando uma variante do modelo GPR para regressão robusta. O segundo tipo de modelo atribui à função latente um Processo t-Student, e para o ruído, utiliza-se uma distribuição t-Student[5] (TPRT). Este é o modelo de Tang *et al.* (2017) referido anteriormente.

Modelo GPRT

O modelo GPRT, como é conhecido na literatura, refere-se ao modelo de regressão de Processo Gaussiano com ruído t-Student. Esta modelagem tem a proposta de modelar valores contaminados por *outliers* na variável de saída $\mathbf{y}$. Trabalhos como os de Neal (1997), Vanhatalo *et al.* (2009. Tema: Advances in neural information processing systems) e Jylanki *et al.* (2011) podem ser citados como referências para este modelo.

Prosseguindo, considere um conjunto de observações $\mathscr{D} = (\mathbf{X}, \mathbf{y})$, em que $\mathbf{X} \in \mathbb{R}^{D \times n}$ é o conjunto de covariáveis do modelo, e $\mathbf{y} \in \mathbb{R}^n$ são as variáveis observadas. O modelo GPR básico apresentado na Seção anterior, em (2.26), era dado da seguinte forma

$$y_i = f(\mathbf{x}_i) + \varepsilon_i \quad \forall \quad i = 1, \ldots, n, \tag{2.43}$$

em que ε_i seguia uma distribuição Gaussiana i.i.d. O modelo GPRT é obtido assumindo uma nova suposição ao ruído do modelo, ou seja, $\boldsymbol{\varepsilon} \sim t - Student(\nu, \mathbf{0}, \sigma^2\mathbf{I})$, e mantendo

[5] Regressão de Processo t-Student com ruído t-Student (TPRT).

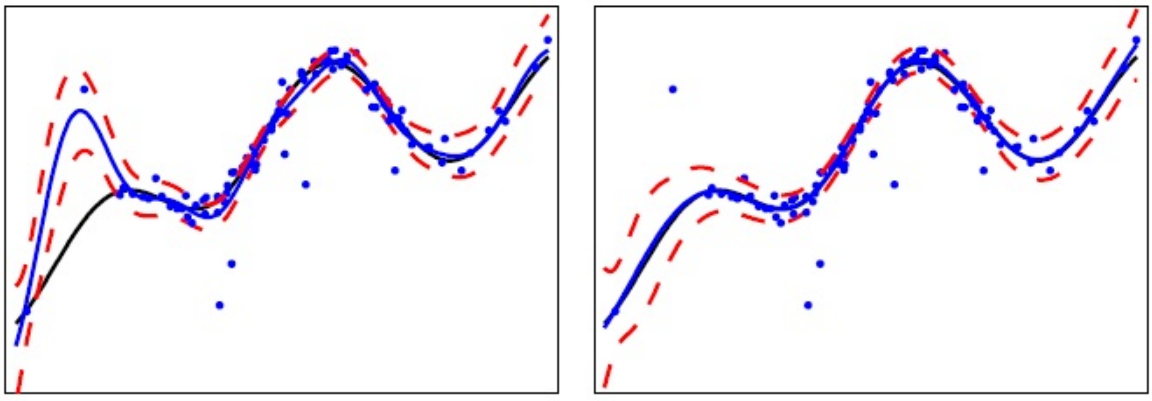
um Processo Gaussiano a priori para f. Dessa forma, o modelo de regressão para y_i com graus de liberdade v é dada por

$$p(y_i|v,f_i,\sigma^2) = \frac{\Gamma((v+1)/2)}{\Gamma(v/2)\sqrt{v\pi}\sigma}\left(1+\frac{(y_i-f_i)^2}{v\sigma^2}\right)^{-(v+1)/2} \quad \forall \quad i=1,\ldots,n, \qquad (2.44)$$

em que $f_i = f(\mathbf{x}_i)$.

Como exemplo, temos na Figura 6 uma ilustração da robustez do modelo com ruído t-Student retirada de Vanhatalo *et al.* (2009. Tema: Advances in neural information processing systems). No gráfico à esquerda na Figura 6, a estimação da média a posteriori de um modelo de regressão de GP com ruído Gaussiano, é significativamente afetada por observações discrepantes. Já o gráfico à direita na Figura 6, mostra a estimação da média a posteriori do modelo de regressão de GP, cujo ruído é dado por uma distribuição t-Student. Vemos que a média a posteriori se mantém no nível das demais observações sem se deixar influenciar de forma significativa pelas observações discrepantes.

Figura 6 – Exemplo de regressão com *outliers*. No gráfico à esquerda, modelo com ruído Gaussiano. No gráfico à direita, modelo com ruído t-Student.



Fonte: Imagem adaptada de Vanhatalo *et al.* (2009. Tema: Advances in neural information processing systems).

Modelo TPRT

O modelo GPRT é extremamente útil para robustez a *outliers* presentes nas variáveis de saída, cujo desempenho tem mostrado ser empiricamente melhor que o GPR tradicional. Mas, no decorrer do tempo vêm sendo despendidos esforços para o desenvolvimento de modelos que não somente combatessem os *outliers* presentes nas

variáveis de saída, mas, ao mesmo tempo, à *outliers* presentes nas variáveis de entrada. Trabalhos que já apresentaram modelos com estrutura robusta nas entradas podem ser citados: Shah *et al.* (2014), Solin e Sarkka (2015), Tang *et al.* (2016) e Tang *et al.* (2017). Todos estes trabalhos utilizam Processo t-Student para lidar com o problema de *outliers* nas variáveis de entrada.

O modelo de Tang *et al.* (2017) propõe-se ser mais abrangente no trato de *outliers*, tanto nas variáveis de entrada quanto na variável de saída. Este modelo vem a ser uma espécie de extenção do trabalho anterior de Tang *et al.* (2016), cujo modelo é uma Regressão de Processo t-Student com ruído t-Student dependente (TPRD).

No modelo TPRT, coloca-se um ruído t-Student independente para lidar com os *outliers* de saída, e uma distribuição a priori de Processo t-Student para a função latente para lidar com os *outliers* de entrada. Os autores argumentam que esta proposta de modelo é mais robusta a *outliers* de entrada e saída que o GPR e GPRT, mostrando através de teoremas que GPR e GPRT são casos especiais do TPRT, e que as vantagens do modelo GPR, já citadas anteriormente, tais como interpretabilidade, flexibilidade não-paramétrica e aprendizagem simples dos hiper-parâmetros, são mantidas.

O modelo TPRT possui inferências analiticamente intratáveis, e por isso é utilizado o método de aproximação de Laplace para calcular a distribuição a posteriori. Dessa forma, temos para f a seguinte priori

$$f \sim \mathscr{TP}(v, \mathbf{0}, \mathbf{K}), \tag{2.45}$$

em que, da mesma forma que em GPR, utilizamos uma priori de vetor média nulo. Com esta priori, $p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ é dada por

$$p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \frac{\Gamma((v+n)/2)}{\Gamma(v/2)(v\pi)^{n/2}|\mathbf{K}|^{1/2}} \left(1 + \frac{1}{v}\mathbf{f}^T\mathbf{K}\mathbf{f}\right)^{-(v+n)/2}, \tag{2.46}$$

e utilizando um ruído t-Student, $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta})$ torna-se,

$$p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{\Gamma((v_2+1)/2)}{\Gamma(v_2/2)\sqrt{v_2\pi}\sigma} \left(1 + \frac{(y_i - f_i)^2}{v_2\sigma^2}\right)^{-(v_2+1)/2}. \tag{2.47}$$

Unindo a distribuição a priori com a função de verossimilhança, temos a proporcional da posteriori $p(\mathbf{f}|\mathscr{D}, \boldsymbol{\theta}) \propto p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$. O método de Laplace é uma aproximação Gaussiana para a moda da posteriori, e parte inicialmente de uma expansão em séries de Taylor de segunda ordem para

a log posteriori $\log p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta})$[6]. O resultado final do método é dado a seguir,

$$p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) = q(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) = N(\mathbf{f}|\hat{\mathbf{f}},\boldsymbol{\Sigma}), \tag{2.48}$$

em que,

$$\begin{aligned}
\hat{\mathbf{f}} &= \arg\max_{\mathbf{f}} p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) = q(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) = N(\mathbf{f}|\hat{\mathbf{f}},\boldsymbol{\Sigma}) \\
&= \arg\min_{\mathbf{f}} \log \mathrm{Q} \\
\log \mathrm{Q} &= \sum_{i=1}^{n} \frac{v_2+1}{2} \log\left[1+\frac{1}{v_2}\left(\frac{y_i-f_i}{\sigma}\right)^2\right] + \frac{v+n}{2}\log\left(1+\frac{1}{v}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right),
\end{aligned} \tag{2.49}$$

e $\boldsymbol{\Sigma}^{-1}$ é a inversa da Hessiana do logaritmo negativo da posteriori condicionada na moda $\hat{\mathbf{f}}$ dada por:

$$\begin{aligned}
\boldsymbol{\Sigma}^{-1} &= -\nabla\nabla \log p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) \\
&= (v+n)\frac{\mathbf{K}^{-1}(v+\hat{\mathbf{f}}^T\mathbf{K}^{-1}\hat{\mathbf{f}}) - 2\mathbf{K}^{-1}\hat{\mathbf{f}}\hat{\mathbf{f}}^T\mathbf{K}^{-1}}{(v+\hat{\mathbf{f}}^T\mathbf{K}^{-1}\hat{\mathbf{f}})^2} + \mathbf{W},
\end{aligned} \tag{2.50}$$

em que $\mathbf{W}$ é uma matriz diagonal com

$$W_{ii} = -(v_2+1)\frac{(y_i-\hat{f}_i)^2 - v_2\sigma^2}{[(y_i-\hat{f}_i)^2 + v_2\sigma^2]^2} \quad i = 1,2,\ldots,n. \tag{2.51}$$

O modelo preditivo para uma observação da variável latente $f(\mathbf{x}_*)$ para uma entrada $\mathbf{x}_*$ pode ser obtido de forma simples. A distribuição preditiva a posteriori sob aproximação de Laplace para f_* é Gaussiana. Com isso, precisamos definir apenas a sua média e variância. Nos limitaremos a apresentar somente a média, a variância pode ser obtida no mesmo caminho, e pode-se consultar Shah *et al.* (2014) e Tang *et al.* (2017) para mais detalhes.

$$\begin{aligned}
E_q(f_*|\mathbf{x}_*,\mathscr{D},\boldsymbol{\theta}) &= \int_{\mathbf{f}} E(f_*|\mathbf{f},\mathbf{X},\mathbf{x}_*) p(\mathbf{f}|\mathscr{D},\boldsymbol{\theta}) d\mathbf{f} \\
&= \mathbf{K}_*^T\mathbf{K}^{-1}\hat{\mathbf{f}},
\end{aligned} \tag{2.52}$$

O modelo TPRT, como mencionado, propõe-se a ter um desempenho melhor no manuseio de *outliers* de entrada e saída, e teoricamente teria um desempenho melhor do que o GPRT. Isso pode ser visto analisando a média a posteriori dos processos nestes

[6] Omitimos daqui em diante a menção aos hiper-parâmetros $\boldsymbol{\psi}$ por simplicidade notacional, retornando a fazê-la em momento oportuno.

dois modelos. A aproximação $\hat{\mathbf{f}}$ do processo a posteriori no modelo GPRT (VANHATALO *et al.*, 2009. Tema: Advances in neural information processing systems), pode ser reescrita da seguinte forma

$$\begin{aligned} \hat{\mathbf{f}} &= \arg\min_{\mathbf{f}} \log \text{Q'} \\ &= \arg\min_{\mathbf{f}} \log \text{Q'} \\ \log \text{Q'} &= \sum_{i=1}^{n} \frac{v_2+1}{2} \log\left[1+\frac{1}{v_2}\left(\frac{y_i-f_i}{\sigma}\right)^2\right] + \mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}. \end{aligned} \tag{2.53}$$

Tomando a aproximação de $\hat{\mathbf{f}}$ no modelo TPRT em (2.49), percebemos que a diferença desta expressão para a expressão em (2.53), é o termo $\frac{v+n}{2}\log\left(1+\frac{1}{v}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right)$ em (2.49) que é o resultado de uma log transformação do termo $\hat{\mathbf{f}}^T\mathbf{K}^{-1}\hat{\mathbf{f}}$ em (2.53). Tang *et al.* (2017) argumentam que se há *outilers* nas variáveis de entrada, o termo $\hat{\mathbf{f}}^T\mathbf{K}^{-1}\hat{\mathbf{f}}$ seria perturbado, e a log transformação poderia diminuir esta perturbação. Portanto, por este caminho, a média da distribuição a posterior aproximada em TPRT seria mais robusta a *outliers* de entrada do que a do modelo GPRT. Seguindo este raciocínio, poderíamos utilizar um caminho similar para mostrar que o modelo GPRT é mais robusto a *outliers* na variável de saída se comparado ao modelo GPR.

Tang *et al.* (2017) também mostram que os modelos GPR e GPRT são casos particulares do modelo TPRT. Estes resultados são por eles apresentados nos Teoremas 1 e 2 respectivamente. Resumidamente, o modelo GPR torna-se um caso especial do TPRT quando $v, v_2 \longrightarrow \infty$, e o GPRT semelhantemente torna-se um caso particular do TPRT quando $v \longrightarrow \infty$. Estes resultados carregam certa intuitividade, pois, como falamos anteriormente, os graus de liberdade de uma distribuição t-Student controlam o peso das caudas da distribuição de tal forma que se $v \longrightarrow \infty$, a distribuição Gaussiana é obtida como resultado limite.

2.4 Função de covariância

A função de covariância codifica as suposições que temos da função que queremos aprender, tais como: suavidade, periodicidade, estacionariedade entre outras. A função de covariância desempenha o papel de uma função de mapeamento, onde a suposição de valores próximos no espaço de entradas $\boldsymbol{\chi}$, possui valores de saída $y \in \mathbb{R}$ próximos. Dessa forma, pontos de treinamento que estão próximos a um ponto de teste podem ser informativos sobre a predição naquele ponto (WILLIAMS; RASMUSSEN, 2006).

Intuitivamente, a função de covariância descreve a força e a direção da dependência linear, a similaridade, ou a informatividade mútua dos valores da função $f(\mathbf{x})$ e $f(\mathbf{x}')$ como uma função das correspondentes entradas (KUSS, 2006). Uma revisão mais abrangente sobre as propriedades das funções de covariância pode ser obtida em Abrahamsen (1997), Gibbs (1998), Kuss (2006), Williams e Rasmussen (2006).

Na literatura, a função de covariância também é conhecida por *kernel*. Os métodos de *kernel* são muito comuns em técnicas de Reconhecimento de Padrões, onde tais técnicas trabalham armazenando informações (aprendizado baseado em memória). As funções de *kernel* geram matrizes semidefinidas positivas da mesma forma que as funções de covariância, desempenhando portanto, papéis semelhantes. Kocijan (2016), através da representação dual, mostra que muitos dos modelos baseados em mapeamento de espaço de características não linear fixo $\boldsymbol{\phi}(\mathbf{x})$, possuem uma equivalência dual com funções *kernel*, sendo dada pela relação de produto interno $k(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{x}')$, em que $\boldsymbol{\phi}(\mathbf{x})$ é o conjunto de funções de base fixa. Williams e Rasmussen (2006) apresentam abordagem similar partindo do modelo de regressão bayesiano linear até chegar na equivalência com funções *kernel*. Semelhantemente, a representação dual mencioanda acima pode ser usada para obter os Processos Gaussianos. Note que quando apresentamos uma interpretação alternativa sobre GP, usando a perspectiva espaço de peso, obtivemos uma matriz de covariância $\mathbf{K}$ que é dada pela relação de produto interno, onde, por (2.36), $\mathbf{K}=\boldsymbol{\phi}^T\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\phi} = k(\mathbf{x},\mathbf{x}')$.

Para trabalharmos com funções de covariância válidas, existem algumas restrições que precisam ser atendidas. Basicamente, a principal é que a função de covariância seja definida positiva. Esta restrição no entanto, pode ser relaxada para uma matriz semidefinida positiva. Em todo caso, essa restrição é necessária para permitir a existência de todas as distribuições finito-dimensionais $\mathbf{f} = [f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n)]$. A característica de simetria da função de covariância, para que esta seja válida, advém da sua própria definição. Mas, na prática é difícil mostrar que uma função de covariância é semidefinida positiva. Kuss (2006) argumenta que a classe de funções de covariância semidefinida positiva é fechada sob certas condições, e deste modo podemos construir novas funções de covariância a partir

das que já são válidas. Algumas dessas condições são dadas a seguir

$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}') \quad (2.54)$$

$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}').k_2(\mathbf{x},\mathbf{x}') \quad (2.55)$$

$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + d \quad (2.56)$$

$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}').d, \quad (2.57)$$

em que k_1 e k_2 são funções de covariância válidas, e $d > 0$ um escalar.

Funções de covariância que dependem apenas de $\mathbf{x}$ e $\mathbf{x}$' através de $\mathbf{x}.\mathbf{x}'$ chamamos de funções *dot product*. Este tipo de função de covariância torna-se invariante a qualquer rotação de suas coordenadas sobre a origem, mas não a translações. Como exemplo dessa classe de funções, temos a função de covariância linear $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$. Funções de covariância que dependem da diferença $\mathbf{x}-\mathbf{x}'$, por outro lado, são invariantes a translações no espaço de entradas $\boldsymbol{\chi}$, e são conhecidas por estacionárias, ou por funções de base radial - RBFs. Assim, podemos escrever uma função de covariância estacionária da seguinte forma

$$k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}+t,\mathbf{x}'+t) \quad \text{para t} \quad \in \boldsymbol{\chi}. \quad (2.58)$$

Processos Gaussianos são estacionários se possuem função média constante e função de covariância estacionária (KUSS, 2006). Mais adiante, se a função de covariância $k(\mathbf{x},\mathbf{x}')$ for função de $|\mathbf{r}| = |\mathbf{x}-\mathbf{x}'|$, então ela é chamada de isotrópica, e é invariante a qualquer movimento rígido (WILLIAMS; RASMUSSEN, 2006); a versão anisotrópica surge, quando $\mathbf{r}$ for dado em função de uma norma $||\mathbf{r}||^2_{\mathbf{W}} = \mathbf{r}^T\mathbf{W}^{-1}\mathbf{r}$ para alguma matriz semidefinida positiva $\mathbf{W}$.

As funções de covariância estacionária são amplamente utilizadas na prática, principalmente em problemas de regressão, onde a suposição de que entradas próximas no espaço de entradas, geram valores da função subjacente similares, independente da posição no espaço de entradas. As funções de covariância podem ser escritas em uma forma normalizada, como a seguir (KUSS, 2006)

$$k(\mathbf{x},\mathbf{x}') = \sigma_s^2 c(\mathbf{x},\mathbf{x}'), \quad (2.59)$$

em que $\sigma_s^2 > 0$ é chamado de sinal da variância, e $|c(\mathbf{x},\mathbf{x}')| \leq 1$ é uma função de correlação semidefinida positiva. Dessa forma, uma função de covariância estacionária escrita na forma da expressão (2.59), pode ser dada como função de $\mathbf{r} = \mathbf{x}-\mathbf{x}'$, tal que $k(\mathbf{x},\mathbf{x}') = \sigma_s^2 c(\mathbf{r})$, onde $|c(\mathbf{r})| \leq 1$.

Apesar de podermos construir funções de covariância que sejam válidas, algumas são inadequadas em aplicações práticas por não levar em conta a estrutura das entradas. Entre elas podemos citar a função de covariância constante $k(\mathbf{x},\mathbf{x}') = \sigma_s^2$, e a função de covariância ruído branco que é constante $k(\mathbf{x},\mathbf{x}') = \sigma_s^2$ se $\mathbf{x} = \mathbf{x}'$ e $k(\mathbf{x},\mathbf{x}') = 0$ caso contrário.

Exemplos de funções de covariância

A seguir, vamos apresentar alguns exemplos de funções de covariância. Aquelas pertencentes à classe das funções de covariância estacionária serão apresentadas na sua forma isotrópica, pois a forma anisotrópica pode ser obtida de forma simples. Para ver mais detalhes sobre os exemplos que apresentaremos aqui, pode-se consultar Kuss (2006), Williams e Rasmussen (2006) e Gibbs (1998). Iniciamos apresentando alguns exemplos de função de covariância estacionária.

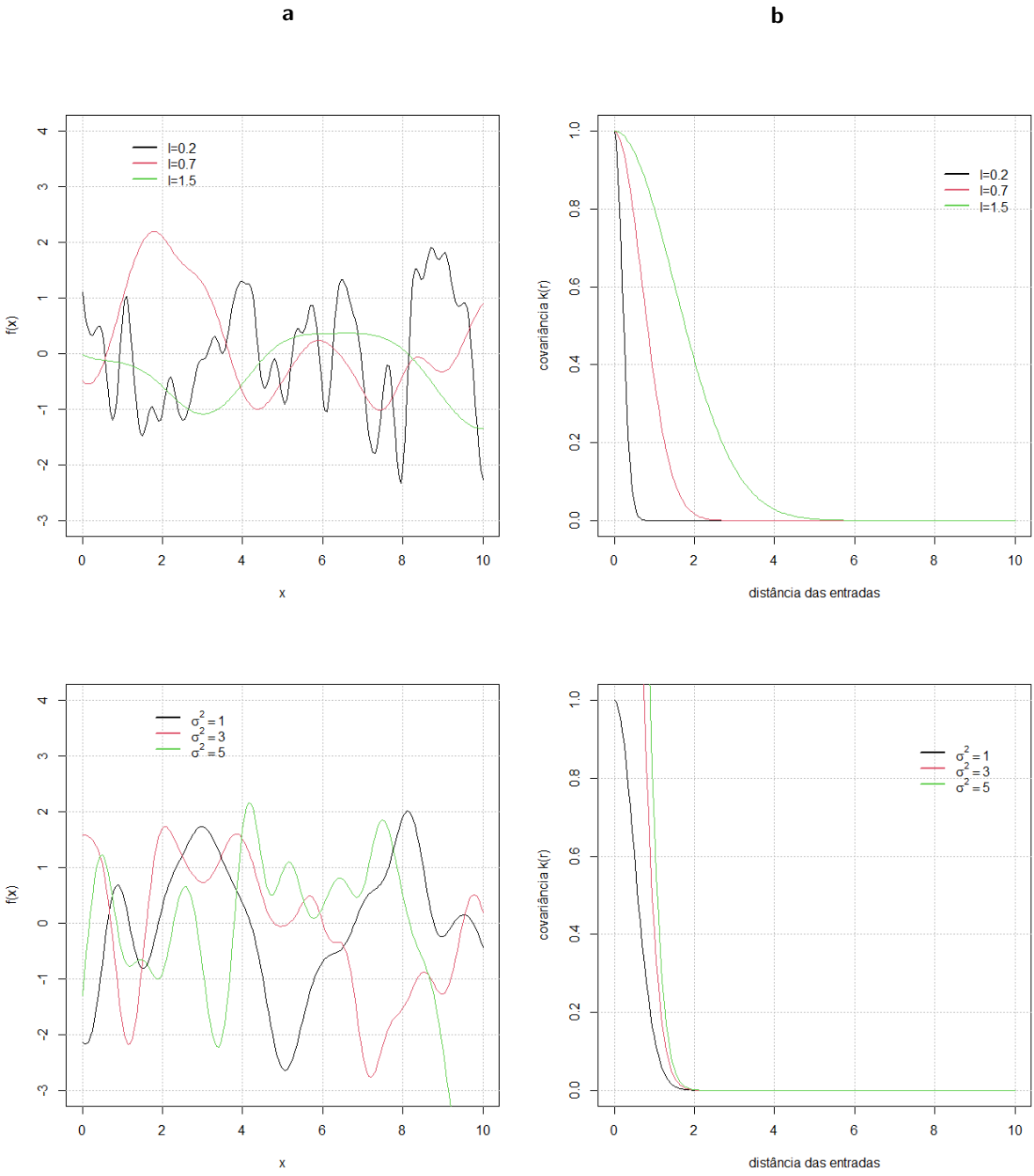
A função de covariância Exponencial Quadrática (EQ) tem sido largamente utilizada em situações práticas, principalmente no contexto de problemas de *Machine Learning* (KUSS, 2006), e possui a seguinte forma básica

$$k_{EQ}(\mathbf{r}) = \sigma_s^2 \exp\left(-\frac{\mathbf{r}^2}{2l^2}\right), \tag{2.60}$$

em que $l > 0$ é o parâmetro que define a característica comprimento-escala. Intuitivamente, este parâmetro controla a taxa de decaimento da covariância, controlando a distância a qual duas observações tornam-se aproximadamente independentes. Valores pequenos da característica comprimento-escala fazem as amostras de funções geradas variarem mais rápido (KUSS, 2006). Já o hiperparâmetro σ_s^2, controla a amplitude de variação vertical do processo gerado. Valores maiores de σ_s^2 dizem respeito à amostra de função com maior variação ao longo do espaço de entradas, valores menores provocam o efeito contrário. Na Figura 7, temos um exemplo unidimensional de algumas amostras geradas de um GP com função de covariância Exponencial Quadrática, onde podemos perceber como as amostras de funções geradas se comportam quando variamos os hiperparâmetros l e σ_s^2. Cabe mencionar que a função de covariância Exponencial Quadrática carrega fortes suposições de que a função latente a ser modelada é bastante suave, pois tal função de covariância é infinitamente diferenciável.

Vale ressaltar que na classe de funções de covariância anisotrópicas, a matriz $\mathbf{W}$ é usada para descrever o comportamento comprimento-escala. Por exemplo, se a característica comprimento-escala é igual em todas as direções, então $\mathbf{W} = l\mathbf{I}$, será equivalente ao caso

Figura 7 – Coluna (a): amostras geradas de GPs com função de covariância EQ variando respectivamente (de cima para baixo): $l = 0.2, 0.7$ e 1.5, fixando $\sigma_s^2 = 1$; $\sigma_s^2 = 1, 3$ e 5, fixando $l = 0.5$. Coluna (b): Função de covariância EQ como função de $\mathbf{r}$ e para diferentes valores de l e σ_s^2.

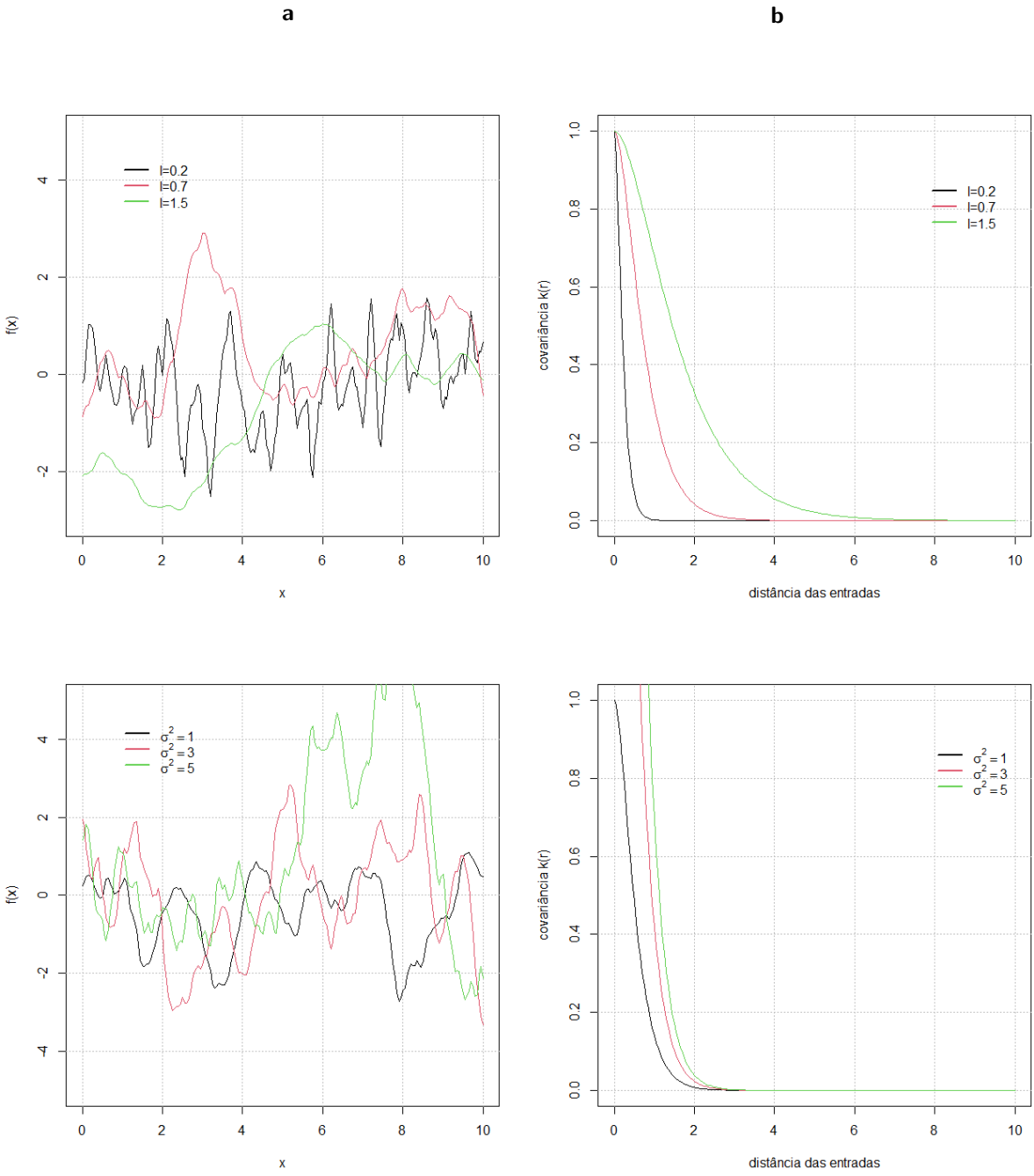


Fonte: Autoria própria.

isotrópico. Esta suposição assume que todas as entradas têm o mesmo comprimento-escala e são igualmente informativas sobre o comportamento local de f (KUSS, 2006).

Uma outra classe de funções de covariância bastante requisitada, e que possui a Exponencial Quadrática como caso especial, é a classe Matérn de funções de covariância,

Figura 8 – Coluna (a): amostras geradas de GPs com função de covariância Matern variando respectivamente (de cima para baixo): $l = 0.2, 0.7$ e 1.5, fixando $\sigma_s^2 = 1$ e $\nu = 3/2$; $\sigma_s^2 = 1, 3$ e 5, fixando $l = 0.5$ e $\nu = 3/2$. Coluna (b): Função de covariância Matern como função de $\mathbf{r}$ e para diferentes valores de l e σ_s^2.



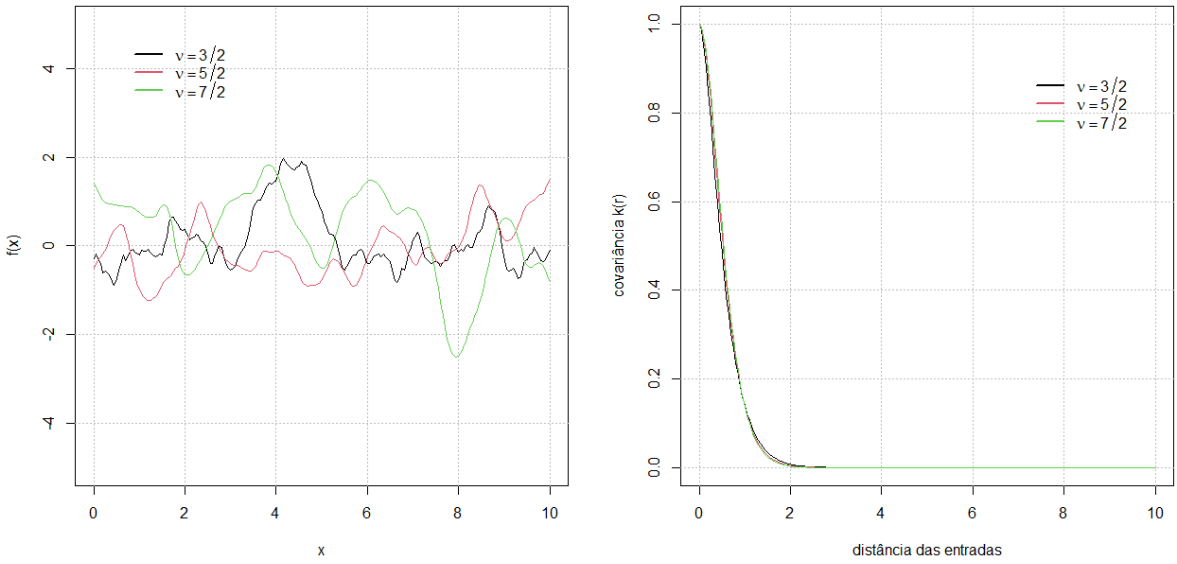
Fonte: Autoria própria.

que possui a seguinte forma funcional

$$k_{Matern}(\mathbf{r}) = \sigma_s^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\mathbf{r}}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}\mathbf{r}}{l} \right), \tag{2.61}$$

em que ν e l são hiperparâmetros positivos e K_{ν} é a função Bessel. O hiperparâmetro

Figura 9 – Gráfico à esquerda: amostras geradas de GPs com função de covariância Matern variando respectivamente $\nu = 3/2, 5/2$ e $7/2$, fixando $\sigma_s^2 = 1$ e $l = 0.5$. Gráfico à direita: Função de covariância Matern como função de $\mathbf{r}$ e para diferentes valores de ν.



Fonte: Autoria própria.

ν controla a diferenciabilidade da função de covariância, e quando $\nu \to \infty$, obtemos a função de covariância Exponencial Quadrática (STEIN, 1999). Esta classe de funções de covariância se torna bastante simples quando configuramos $\nu = p + 1/2$, em que p é um inteiro não negativo, pois a mesma se torna em uma simples função que é produto de um termo exponencial e uma função polinomial de ordem p. Nas Figuras 8 e 9, temos alguns exemplos de amostras geradas de GP com a função de covariância Matérn sob diferentes valores dos hiperparâmetros ν, l e σ_s^2. Como exemplo de caso especial da classe de função de covariância Matérn, podemos obter a função de covariância de Laplace quando $\nu = 1/2$, chegando ao seguinte resultado

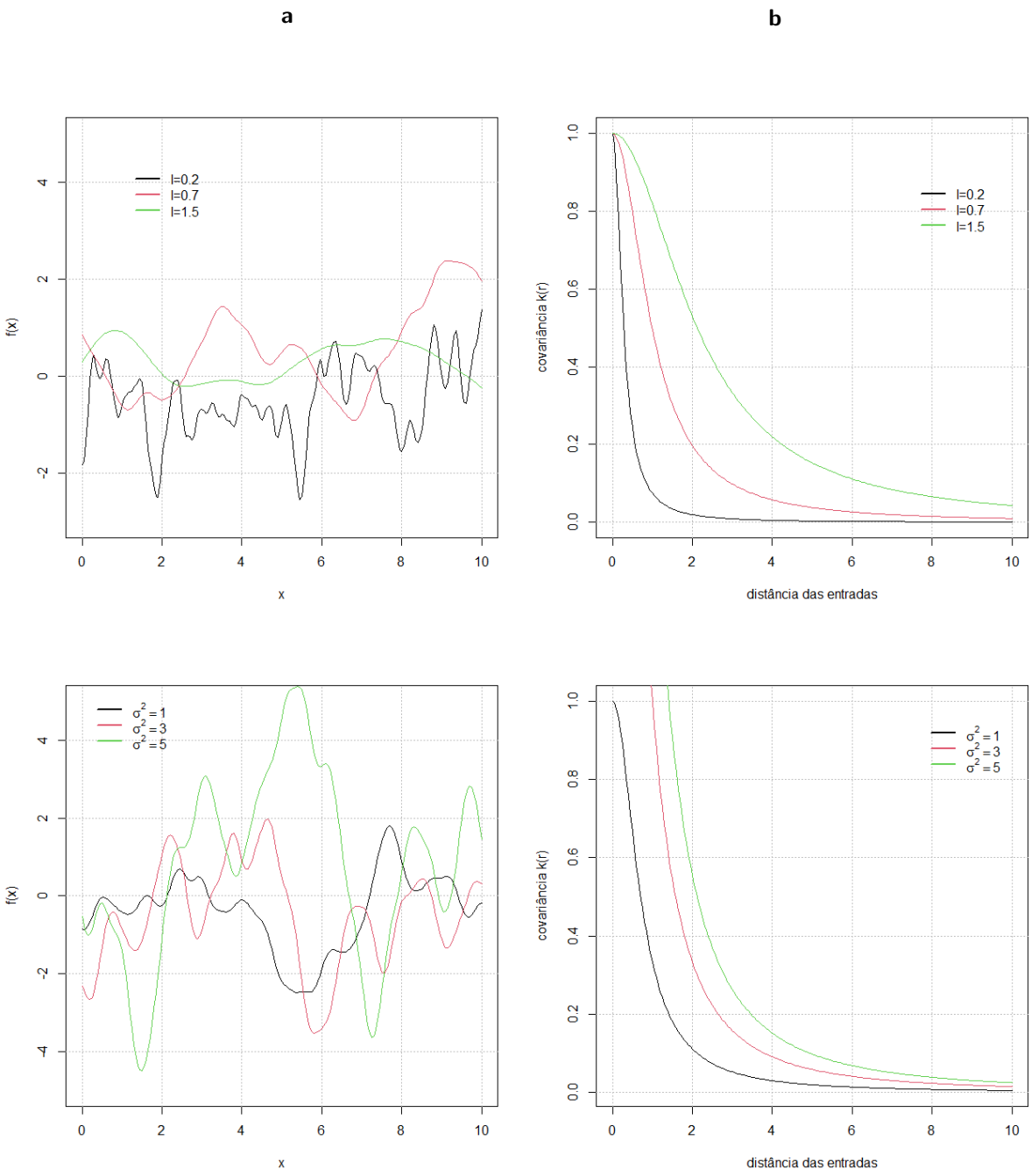
$$k_L(\mathbf{r}) = \sigma_s^2 \exp\left(-\frac{\mathbf{r}}{l}\right). \tag{2.62}$$

Uma outra função de covariância estacionária que pode ser citada, é a função de covariância Radial Quadrática (RQ). Esta pode ser dada pela seguinte forma

$$k_{RQ}(\mathbf{r}) = \sigma_s^2 \left(1 + \frac{\mathbf{r}^2}{2\alpha l^2}\right)^{-\alpha}, \tag{2.63}$$

em que $\alpha, l > 0$. Esta função de covariância pode ser vista como uma mistura de escala de funções de covariância Exponenciais Quadráticas com diferentes características de

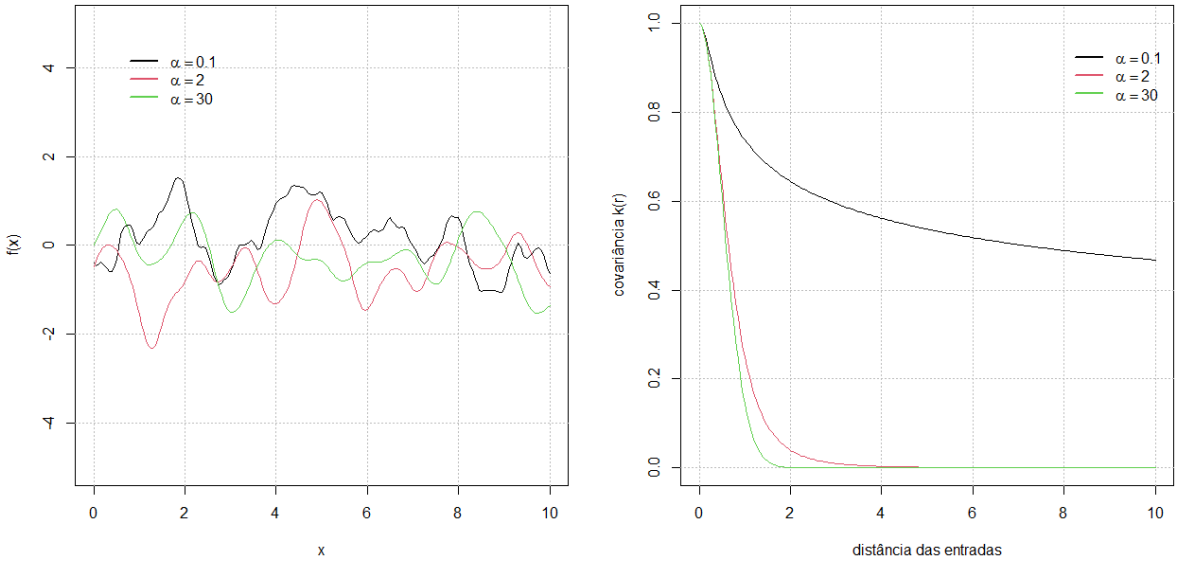
Figura 10 – Coluna (a): amostras geradas de GPs com função de covariância RQ variando respectivamente (de cima para baixo): $l = 0.2, 0.7$ e 1.5, fixando $\sigma_s^2 = 1$ e $\alpha = 1$; $\sigma_s^2 = 1, 3$ e 5, fixando $l = 0.5$ e $\alpha = 1$. Coluna (b): Função de covariância RQ como função de $\mathbf{r}$ e para diferentes valores de l e σ_s^2.



Fonte: Autoria própria.

comprimento-escala. Williams e Rasmussen (2006) mostram que, usando uma notação ligeiramente diferente, o limite da função de covariância Radial Quadrática para $\alpha \longrightarrow \infty$, é uma função de covariância Exponencial Quadrática com característica comprimento-escala l. Amostras geradas de GP utilizando esta função de covariância, estão inseridas nas

Figura 11 – Gráfico à esquerda: amostras geradas de GPs com função de covariância RQ para $\alpha = 0.1, 2$ e 30, fixando $\sigma_s^2 = 1$ e $l = 0.5$. Gráfico à direita: Função de covariância RQ como função de $\mathbf{r}$ e para diferentes valores de α.



Fonte: Autoria própria.

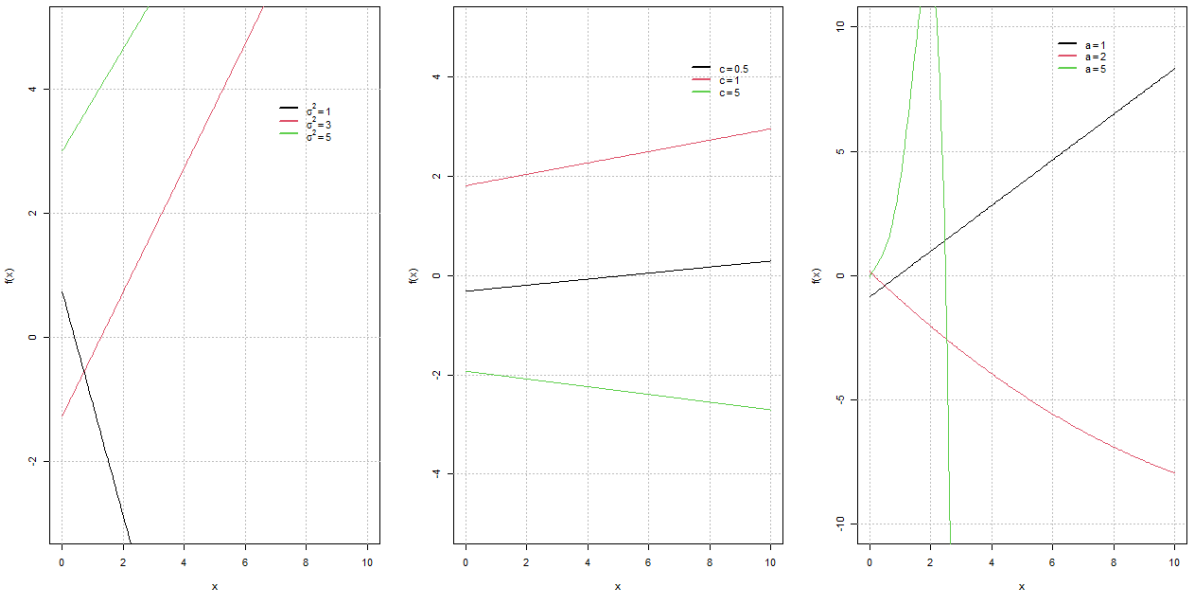
Figuras 10 e 11. Na Figura 11, podemos ver como a sensibilidade de α influencia no decaimento da covariância à medida que aumentamos a distância no espaço de entradas.

Funções de covariância estacionárias abragem muitos casos de interesse prático, porém, há situações nas quais os dados apresentam alguma forma de não-estacionariedade. Gibbs (1998) apresenta algumas particularidades úteis em formas não estacionárias, já que as mesmas são difíceis de serem obtidas, por não se conseguir determinar de forma precisa como ocorrem as mudanças da função no espaço de entradas $\boldsymbol{\chi}$. A função de covariância de Rede Neural pode ser citada como exemplo, e pode ser vista em mais detalhes em Williams e Rasmussen (2006). Outros exemplos de funções de covariância podem ser obtidos da classe polinomial de funções de covariância, que possui a seguinte relação

$$k_p(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \left(\mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x}' + c\right)^a, \tag{2.64}$$

em que $a \in \mathbb{N}$, $c \geq 0$, e $\boldsymbol{\Sigma}$ é uma matriz positiva definida $(D+1) \times (D+1)$. Para $a = 1$, temos como caso especial a função de covariância linear $k_p(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x}' + c$. Nesse caso, podemos perceber que o modelo linear bayesiano pode ser obtido como um caso particular dos modelos de Processo Gaussiano (KUSS, 2006). Na Figura 12, temos exemplos de amostras geradas de GPs utilizando essa função de covariância. Na Figura 12, as amostras foram geradas para diferentes valores de σ_s^2, c e a.

Figura 12 – Amostras geradas de GPs da classe de função de covariância polinomial para diferentes valores de σ_s^2, c e a, quando $\mathbf{\Sigma} = \mathbf{I}$.



Fonte: Autoria própria.

3 CLASSIFICAÇÃO COM PROCESSOS GAUSSIANOS E T-STUDENT

Nas aplicações práticas, o conjunto de todas as entradas $\mathbf{x} \in \mathbb{R}^D$, com $D \in \mathbb{N}$, é organizado em uma matriz de entradas $\mathbf{X}$ (ou matriz de características). Problemas, nos quais para cada vetor de entrada $\mathbf{x}$ há um respectivo valor de saída y[1] são chamados de aprendizado supervisionado. Em contrapartida, quando esta saída não está definida, os problemas correspondentes passam a ser chamados de aprendizado não supervisionado. Nesse tipo de situação, o objetivo consite em encontrar grupos de exemplos similares nos dados, semelhante a uma análise de clusters, procedimento comum na Estatística Multivariada. As aplicações realizadas neste trabalho serão feitas sob a estrutura de aprendizado supervisionado.

Como dito anteriormente, o problema de classificação em linhas gerais tem por objetivo atribuir a um vetor de entrada $\mathbf{x}$ D-dimensional uma classe C_k dentre K classes distintas. Em um senso comum, as classes são escolhidas para serem disjuntas (BISHOP, 2006), cada classe correspondendo a uma região distinta do espaço de entradas $\boldsymbol{\chi} = \mathbb{R}^D$. Os modelos de classificação, que constroem fronteiras de decisão que são funções lineares dos vetores de entradas, são conhecidos por modelos lineares. As superfícies de decisão resultantes para estes modelos são hiperplanos D-1 dimensionais no espaço de entradas D-dimensional. Os modelos de processos latentes, entretanto, são mais flexíveis e podem construir as mais variadas formas de fronteira de decisão, não se limitando unicamente ao caso linear.

Bishop (2006) ressalta que em uma tarefa de classificação, onde se processa o problema em dois estágios, o estágio da inferência e o estágio da decisão, há pelo menos três abordagens distintas que poderiam ser implementadas. A mais simples envolve construir funções discriminantes que atribuem diretamente cada vetor de entrada $\mathbf{x}$ a uma classe específica. A mais complexa envolve modelar as distribuições condicionais $p(C_k|\mathbf{x})$ de classe em um estágio de inferência e então, subsequentemente, usar esta distribuição para tomar decisões. Nesta última categoria, conforme Jain *et al.* (2000), ainda há uma dicotomia que pode ser destacada: se o problema consiste em uma tarefa de classificação paramétrica ou não-paramétrica. Os classificadores de Processos Gaussianos e t-Student, dentro desta dicotomia, estão inseridos na classe dos modelos não-paramétricos.

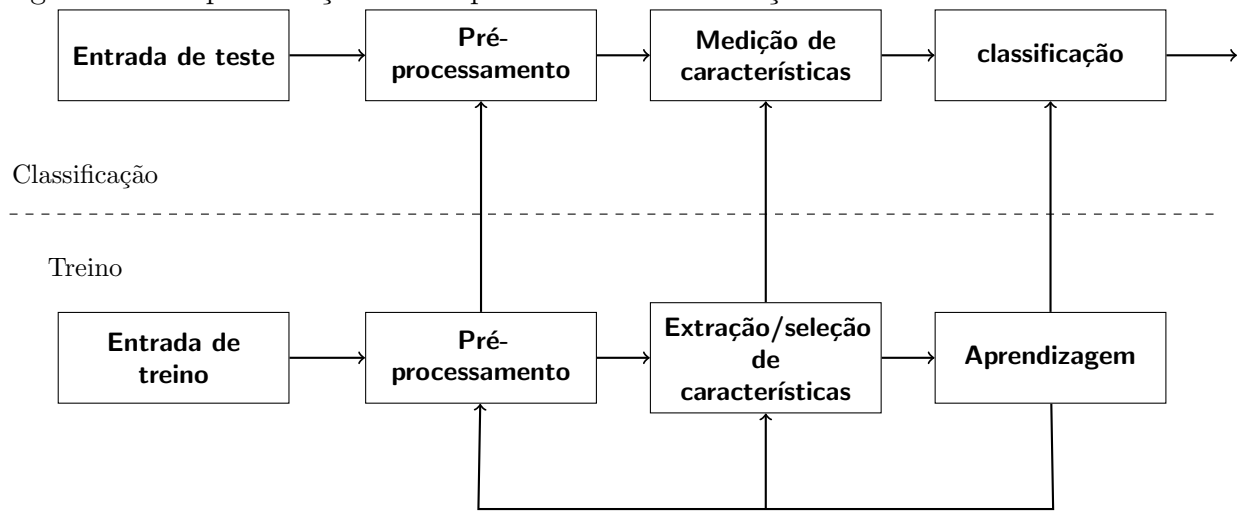
[1] Em classificação, esta saída pertence a um conjunto de valores discretos.

Um dos objetivos a ser alcançado por um classificador é o poder de generalização, que neste caso seria a habilidade de classificar corretamente novas entradas, sendo esta métrica a mais comumente utilizada para avaliar a eficácia de modelos de classificação. Às vezes uma boa performance do classificador no conjunto de treinamento não implica em um bom desempenho para novas entradas de teste. Alguns fatores como o tamanho da amostra de treinamento e os valores específicos dessa amostra interferem na habilidade de generalização do classificador, uma vez que os dados de teste são diferentes dos que foram utilizados na fase de treinamento.

Na Figura 13 temos a representação de um sistema de classificação. Pela Figura, podemos de forma sucinta resumir o modo de operação de um classificador em duas fases: fase de "treino" e fase de "teste" (representada por "Classificação"). Tanto no estágio de treino quanto na fase de testes, as variáveis de entrada podem passar por um pré-processamento[2], cujo objetivo é poder eliminar qualquer tipo de ruído, e/ou outras interferências, contribuíndo assim para uma representação compacta das variáveis que definem os padrões (JAIN *et al.*, 2000). No modo de treino, a etapa de extração de características pode ser útil se o banco de dados possuir padrões com alta dimensionalidade. A extração de características é utilizada para reduzir a dimensionalidade das variáveis de entrada , extraindo características que tornem as classes bem mais evidentes. Jain *et al.* (2000) citam que algumas áreas como: Mineração de Dados (*Data Mining*), Pesquisa na Web (*Web Searching*), Recuperação de Dados Multimídia (*Retrieval of Multimedia Data*), Reconhecimento de Face e Reconhecimento de Dígitos Manuscritos, utilizam algum procedimento de extração de características, já que o conjunto de padrões disponíveis nestas aplicações são tipicamente milhares (MORAIS, 2010). Ainda no modo de treino, a etapa de aprendizagem é representada por alguma técnica/procedimento que permita construir a regra de decisão do classificador. Por fim, no modo de classificação (fase de teste) para uma nova entrada de teste, o classificador fará as medições das características daquele padrão e o atribuirá a uma classe de acordo com a regra de decisão desenvolvida no modo de treinamento.

[2] Um exemplo de pré-processamento, são as normalizações feitas nas variáveis de entrada. Algumas podem ser citadas: 1) Z-score; 2) Normalização pela Norma e 3) Normalização decimal.

Figura 13 – Representação de um problema de classificação estatístico.



Fonte: Autoria própria.

3.1 Modelos lineares probabilísticos

Os modelos lineares probabilísticos são classificadores que, como o próprio nome diz, classificam os padrões de entrada com base em probabilidade. Dessa forma um dado padrão $\mathbf{x}$, pertencente aos $\mathbb{R}^D$ proveniente da classe C_k, é visto como uma observação aleatória da distribuição de probabilidade de classe-condicional $p(\mathbf{x}|C_k)$. Esta aleatoriedade pode ser fundamentada no argumento de que as medições dos padrões carregam certas variações aleatórias, seja por parte do instrumento de medição, ou pelas diferentes características que cada padrão possui (THEODORIDIS; KOUTROUMBAS, 2009).

Em aplicações práticas, as classes de um determinado problema de classificação são comumente conhecidas por rótulos. Para podermos formular a regra de classificação, é conveniente representar as classes de forma que permita a interpretação probabilística. Para isso, utilizaremos o esquema de representação 1-de-K, que consiste de um vetor $\mathbf{t}$ de comprimento K, em que o componente t_k correspondente à classe verdadeira C_k assume valor 1, e as demais posições assumem valor zero (Bishop, 2006). Por exemplo, se K=5, e um padrão de treinamento pertence à classe 2, o vetor $\mathbf{t}$ será dado por $\mathbf{t} = [0,1,0,0,0]^T$.

O ponto de início de uma abordagem probabilística para classificação é a partir da distribuição conjunta $p(C_k, \mathbf{x})$, com $k = 1, \ldots, K$ (WILLIAMS; RASMUSSEN, 2006). Usando a regra do produto de probabilidades, a distribuição conjunta pode ser fatorada como $p(C_k)p(\mathbf{x}|C_k)$ ou $p(\mathbf{x})p(C_k|\mathbf{x})$. Isso dá origem a duas abordagens diferentes para classificação. A primeira abordagem é chamada de generativa e calculamos a probabilidade

condicional $p(C_k|\mathbf{x})$ (posteriori) para $k = 1,\ldots,K$, através do teorema de Bayes, modelando as distribuições de classe-condicional $p(\mathbf{x}|C_k)$ para $k = 1,\ldots,K$ e as distribuições a priori $p(C_1),\ldots,p(C_K)$. E a segunda é chamada de abordagem discriminativa, onde $p(C_k|\mathbf{x})$ é modelado de forma direta (WILLIAMS; RASMUSSEN, 2006). Descreveremos as duas abordagens mais adiante, detalhadamente. Veremos posteriormente, quando tratarmos de modelos de classificação utilizando Processos Gaussianos e t-Student, que estes utilizam os fundamentos dos modelos lineares apresentados nessa Seção.

Em classificação utilizando modelos probabilísticos, é muito comum o uso da abordagem bayesiana para construir as fronteiras de um classificador através da teoria das decisões (pode-se consultar Berger (1985), para mais detalhes sobre teoria da decisão). Portanto, a seguir faremos uma breve introdução à teoria das decisões.

Teoria das decisões

Para um dado padrão de entrada $\mathbf{x}$, teremos estimativas de probabilidades provindas das distribuições condicionais $p(C_k|\mathbf{x})$, para $k = 1,\ldots K$. Contudo, apenas obter estimativas de probabilidade não é o suficiente se não há um procedimento de tomada de decisões que possa nos conduzir a uma direção plausível do que fazer. Necessitamos muitas vezes tomar uma ação específica que seja ótima segundo algum senso comum. Neste ínterim, a teoria das decisões nos propicia um procedimento apropriado para tomada de decisões.

Primeiramente temos que ter em mente que a regra de classificação que será construída, dividirá o espaço de características em R_k, com $k = 1,\ldots,K$ regiões, uma para cada classe[3]. Se o objetivo consistir em minimizar a chance de tomar uma ação errada, ou seja, de atribuir um padrão $\mathbf{x}$ a uma classe errada, podemos pensar intuitivamente que isso é alcançado quando $\mathbf{x}$ pertence a maior probabilidade posterior $p(C_k|\mathbf{x})$, que em outras palavras, é equivalente a escolher a região R_k de tal forma que o padrão $\mathbf{x}$ seja atribuído a classe para o qual a probabilidade a posterior $p(C_k|\mathbf{x})$ seja maior (BISHOP, 2006). O classificador de Bayes, segundo esse critério, é então obtido da minimização da taxa de erro. Considere o caso binário, por exemplo C_1 e C_2, e as distribuições a posteriori $p(C_1|\mathbf{x}) \propto p(\mathbf{x}|C_1)p(C_1)$ e $p(C_2|\mathbf{x}) \propto p(\mathbf{x}|C_2)p(C_2)$.

[3] Pode haver o caso de ter mais de uma região para uma determinada classe; isso vai depender do problema em questão a ser resolvido.

Com isso, temos a seguinte probabilidade de erro total

$$P_e = p(C_1)\int_{R_2} p(\mathbf{x}|C_1)d\mathbf{x} + p(C_2)\int_{R_1} p(\mathbf{x}|C_2)d\mathbf{x}, \tag{3.1}$$

que é mínima quando a região R_1 do espaço de entradas for escolhida de tal forma que $R_1 : p(C_1|\mathbf{x}) > p(C_2|\mathbf{x})$. O caso multiclasse é direto e segue o mesmo caminho.

Contudo, há um problema com esse critério. Theodoridis e Koutroumbas (2009) argumentam que utilizar a minimização da taxa de erro como forma de obtenção do classificador, às vezes não é adequado, pois tal critério atribui a mesma importância a todos os erros. Tomemos como exemplo a situação em que precisamos diagnosticar um paciente como tendo câncer ou não. Se o paciente que não tem câncer é diagnosticado com a doença, as consequências poderiam ser o sofrimento por parte do paciente até a descoberta de que este não tinha câncer por meio de uma investigação mais apurada do diagnóstico. Se o inverso acontecer, ou seja, se o paciente têm câncer e é diagnosticado como não tendo, as consequências poderiam ser mais desastrosas, como a morte prematura do paciente por falta de tratamento. Percebemos, através deste exemplo, que a nossa intuição nos levaria a nos precavermos muito mais em não cometermos o segundo erro do que o primeiro. Assim dependendo do problema em questão, devemos atribuir alguma forma de penalidade de acordo com a significância do erro.

Para formalizar essa questão, definimos primeiramente uma função $L(k,k')$ chamada de função perda, que refere-se à perda incorrida por fazer a decisão k' quando a classe verdadeira é C_k. Usualmente define-se $L(k,k) = 0$ para todo k. Através da função de perda, definimos a função de risco (a perda esperada) $R(k'|\mathbf{x}) = \sum_k L(k,k')p(C_k|\mathbf{x})$ e a regra de decisão ótima k^* é aquela que minimiza o risco $R(k'|\mathbf{x})$ (WILLIAMS; RASMUSSEN, 2006).

Há várias opções que podem ser escolhidas como funções de perda, dentre as mais comuns, temos: perda quadrática $L(k,k') = (k'-k)^2$ e perda absoluta $L(k,k') = |k'-k|$. Há uma opção, contudo, que nos possibilita uma significativa simplificação na regra de decisão, nos permitindo chegar no mesmo critério de decisão obtido pela minimização da taxa de erro: a função de perda zero-um, que penaliza com uma unidade uma classificação incorreta com zero caso contrário. A função de perda zero-um pode ser dada como a seguir

$$L(k,k') = \begin{cases} 0, & k = k' \\ 1, & k \neq k' \end{cases}. \tag{3.2}$$

Com esta função de perda, a regra de classificação conhecida como classificador de Bayes (também conhecido como Máximo a posteriori (MAP)) é dada por

$$p(C_k|\mathbf{x}) > p(C_j|\mathbf{x}) \quad \forall \quad k \neq j. \tag{3.3}$$

Nas regiões do espaço de características, onde a probabilidade máxima $\max_j p(C_j|\mathbf{x})$ é relativamente baixa, pode haver a ocorrência de classificação incorreta. Dentre os motivos para tal situação, poderíamos apontar a existência de uma forte região de sobreposição entre classes, ou a falta de padrões de treinamento suficientes para tal classe (WILLIAMS; RASMUSSEN, 2006). Uma forma de contornar este problema seria propor uma opção de rejeição, onde estabeleceríamos um certo limite $\theta \in (0,1)$, tal que se $\max_j p(C_j|\mathbf{x}) \geq \theta$, prosseguiríamos com a classificação da entrada $\mathbf{x}$ à classe C_j, caso contrário, rejeitaríamos, e estabeleceríamos um sistema mais sofisticado para classificação. Este valor limite $\theta \in (0,1)$ controla a quantidade de exemplos de treinamentos que serão rejeitados. Por exemplo, se $\theta = 1$, todos os padrões de entrada serão rejeitados, mas se para um determinado problema de K classes definirmos $\theta < 1/K$, então nenhum padrão será rejeitado (BISHOP, 2006). A taxa de erro geralmente aumentará à medida que a taxa de rejeição aumentar, logo, θ deve ser escolhido com muita cautela.

Modelos generativos probabilísticos

Os modelos generativos probabilísticos modelam as distribuições de classe-condicional $p(\mathbf{x}|C_k)$, bem como as distribuições a priori das classes $p(C_k)$, originando a distribuição a posteriori $p(C_k|\mathbf{x})$ através do teorema de Bayes (BISHOP, 2006). O adjetivo generativo surge do fato de tais modelos permitirem a obtenção da distribuição marginal dos dados $p(\mathbf{x})$, possibilitando a geração de dados sintéticos no espaço de entradas por meio de amostragem. Para podermos apresentar o modelo em questão, vamos iniciar pelo caso mais simples, K=2, e subsequentemente o caso mais geral, ou seja, para $K > 2$. Assim, temos que a distribuição a posteriori para a classe C_1 pode ser dada como

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)} = \frac{1}{1+\exp(-a)} = \sigma(a), \tag{3.4}$$

em que

$$a = \log\left[\frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}\right], \tag{3.5}$$

$\sigma(a)$ é uma função sigmóide, que neste caso é a função logística

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{3.6}$$

A função sigmóide (forma de S) logística é também conhecida por "*squashing function*" (função de compressão) por conseguir mapear todo o eixo real em um intervalo finito. Ela apresenta algumas características importantes que podem ser destacadas, entre elas, temos a simetria

$$\sigma(-a) = 1 - \sigma(a). \tag{3.7}$$

Vale notar que na expressão (3.4) apenas escrevemos a probabilidade a posteriori de forma conveniente, tal que possamos apresentar $a(\mathbf{x})$ como um funcional de $\mathbf{x}$. Por exemplo, se considerarmos $a(\mathbf{x})$ como uma função linear de $\mathbf{x}$, então, neste caso a probabilidade a posteriori torna-se um modelo linear generalizado (BISHOP, 2006), e assim poderíamos utilizar todo o aparato de ferramentas disponíveis dessa classe de modelos.

Para o caso em que $K > 2$, temos a seguinte expressão para a distribuição a posteriori de $p(C_k|\mathbf{x})$

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}. \tag{3.8}$$

Esta função é conhecida por exponencial normalizada, ou função *softmax*. A quantidade a_k é dada por

$$a_k = \log[p(\mathbf{x}|C_k)p(C_k)]. \tag{3.9}$$

Para a construção das fronteiras de decisão, precisamos propor uma distribuição para os dados de entrada, precisamente para as densidades de classe-condicionais. As entradas $\mathbf{x}$ podem ser de diferentes tipos: discretas, contínuas e mistas. Mas, para efeito de ilustração, vamos apenas apresentar o caso contínuo, onde podemos atribuir uma distribuição Gaussiana às densidades de classe-condicionais e então analisar a forma das probabilidades a posteriori resultantes (BISHOP, 2006). Assumindo que todas as densidades de classe-condicionais têm a mesma matriz de covariância em comum, temos para cada classe C_k, a seguinte densidade

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}. \tag{3.10}$$

Considerando a situação inicial de K=2, utilizando (3.5) e (3.6), e após algumas manipulações algébricas, temos a seguinte expressão

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0), \tag{3.11}$$

em que

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{3.12}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\left[\frac{p(C_1)}{p(C_2)}\right]. \tag{3.13}$$

A distribuição a posteriori dada em (3.11) tem argumentos que são funções lineares de $\mathbf{x}$. Isso porque os termos quadráticos em $\mathbf{x}$ foram cancelados devido a suposição de matriz de covariância comum. As fronteiras de decisão para este modelo correspondem a superfícies onde as probabilidades a posteriori $p(C_k|\mathbf{x})$ são constantes, e portanto, darão origem a fronteiras de decisão lineares no espaço de entradas.

Para o caso em que $K > 2$, utilizando (3.8) e (3.9) temos

$$a_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}, \tag{3.14}$$

em que

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \tag{3.15}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + ln[p(C_k)]. \tag{3.16}$$

Constatamos novamente que o termo funcional dado em (3.14) tem argumentos que são funções lineares das entradas, como consequência da suposição de matriz de covariância comum (BISHOP, 2006). Cabe nesse momento ressaltar que para podermos identificar de forma completa a distribuição a posteriori das classes $p(C_k|\mathbf{x})$ pela fórmula de Bayes, devemos estimar os parâmetros das densidades de classe-condicionais, assim como, das respectivas distribuições a priori de classe. Isso pode ser feito, dependendo das escolhas das densidades de classe-condicionais, pelo método da máxima verossimilhança.

Por fim, se admitirmos que as densidades de classe-condicionais não compartilham a mesma matriz de covariância, o cancelamento dos termos quadráticos em $\mathbf{x}$ que antes era possível, não ocorrerá mais, dando origem a um outro classificador, comumente conhecido na literatura por Discriminante Quadrático. Vale notar que devido ao fato de os argumentos de entrada não serem mais funções lineares de $\mathbf{x}$, as fronteiras de decisão respectivas também não serão lineares (serão quadráticas).

Solução por máxima verossimilhança

Uma vez que determinamos uma forma paramétrica para as densidades de classe-condicionais $p(\mathbf{x}|C_k)$, podemos determinar as estimativas dos valores dos parâmetros e das distribuições a priori de classe usando máxima verossimilhança (BISHOP, 2006). Vamos considerar apenas o caso para K=2 para exemplificar. Consideremos também que as densidades de classe-condicionais compartilham a mesma matriz de covariância. Assim, dado um conjunto de dados $\mathscr{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, onde $y_i = 1$ denotando a classe C_1 e $y_i = 0$ denotando a classe C_2 respectivamente, considere também que a probabilidade de classe a priori $p(C_1) = \pi$ e $p(C_2) = 1 - \pi$. Então, temos que a função de verossimilhança é dada por

$$p(\mathbf{y}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} [\pi N(\mathbf{x}_i|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{y_i} [(1-\pi)N(\mathbf{x}_i|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-y_i}, \tag{3.17}$$

em que $\mathbf{y} = [y_1, \ldots, y_n]^T$, $\boldsymbol{\mu_1}$ é um vetor de médias $(D+1) \times 1$, $\boldsymbol{\mu_2}$ é um vetor de médias $(D+1) \times 1$ e $\boldsymbol{\Sigma}$ é uma matriz de covariância positiva definida $(D+1) \times (D+1)$. Constatamos que a verossimilhança pôde ser fatorada devido a independência dos padrões, ou seja, cada padrão se restringir a uma única classe. Para proceder com a maximização, trabalhamos com o log da função de verossimilhança, que é dada pela seguinte expressão

$$\begin{aligned} \log p(\mathbf{y}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) &= \sum_{i=1}^{n} [y_i \log \pi + y_i \log N(\mathbf{x}_i|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + \\ &\quad + (1-y_i)\log(1-\pi) + (1-y_i)\log N(\mathbf{x}_i|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]. \end{aligned} \tag{3.18}$$

Derivando com relação a π e igualando a zero, chegamos na seguinte expressão final

$$\begin{aligned} \sum_{i=1}^{n} [(1-\hat{\pi})y_i - \hat{\pi}(1-y_i)] &= 0 \\ \sum_{i=1}^{n} [y_i - \hat{\pi}] &= n_1 - n\hat{\pi} \\ \hat{\pi} &= \frac{n_1}{n}, \end{aligned} \tag{3.19}$$

em que n_1 é o número total de pontos de dados que pertencem à classe C_1, e por consequência, temos que $n = n_1 + n_2$. Pelo valor do resultado acima, percebemos que o estimador da probabilidade a priori da classe C_1 nada mais é que a média aritmética dos pontos de dados dessa classe. Procedendo da mesma maneira com os parâmetros $\boldsymbol{\mu}_1$ e $\boldsymbol{\mu}_2$, temos os

seguintes resultados

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_1 &= \sum_{i=1}^{n} -y_i(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1)^T\boldsymbol{\Sigma}^{-1} = \mathbf{0} \\
&= \sum_{i=1}^{n} -y_i(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1)^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma} = \mathbf{0}^T\boldsymbol{\Sigma} \\
&= \frac{1}{n_1}\sum_{i=1}^{n} y_i\mathbf{x}_i \qquad (3.20) \\
\hat{\boldsymbol{\mu}}_2 &= \sum_{i=1}^{n}(1-y_i)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_2)^T\boldsymbol{\Sigma}^{-1} = \mathbf{0} \\
&= \sum_{i=1}^{n}[(1-y_i)\mathbf{x}_i - (1-y_i)\hat{\boldsymbol{\mu}}_2] = \mathbf{0} \\
&= \frac{1}{n_2}\sum_{i=1}^{n}(1-y_i)\mathbf{x}_i \qquad (3.21)
\end{aligned}$$

que são a média dos vetores de entradas atribuídos à classe C_1 e a média dos vetores de entradas atribuídos à classe C_2 respectivamente. Para a matriz de covariância $\boldsymbol{\Sigma}$, tomando os termos da log verossimilhança que dependem de $\boldsymbol{\Sigma}$, temos

$$\begin{aligned}
&-\frac{1}{2}\sum_{i=1}^{n} y_i\log|\hat{\boldsymbol{\Sigma}}| - \frac{1}{2}\sum_{i=1}^{n} y_i(\mathbf{x}_i - \boldsymbol{\mu}_1)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_1) \\
&-\frac{1}{2}\sum_{i=1}^{n}(1-y_i)\log|\hat{\boldsymbol{\Sigma}}| - \frac{1}{2}\sum_{i=1}^{n}(1-y_i)(\mathbf{x}_i - \boldsymbol{\mu}_2)^T\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_2) \\
&= -\frac{n}{2}\log|\hat{\boldsymbol{\Sigma}}| - \frac{n}{2}Tr\{\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{S}\}, \qquad (3.22)
\end{aligned}$$

em que

$$\begin{aligned}
\mathbf{S} &= \frac{n_1}{n}\mathbf{S}_1 + \frac{n_2}{n}\mathbf{S}_2 \qquad (3.23) \\
\mathbf{S}_1 &= \frac{1}{n_1}\sum_{i=1}^{n_1}(\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T \qquad (3.24) \\
\mathbf{S}_2 &= \frac{1}{n_2}\sum_{i=1}^{n_2}(\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^T. \qquad (3.25)
\end{aligned}$$

Bishop (2006) argumenta que o resultado padrão de solução por máxima verossimilhança para a matriz de covariância é $\hat{\boldsymbol{\Sigma}} = \mathbf{S}$, que representa uma média ponderada das matrizes de covariância de cada uma das duas classes separadamente. Este resultado é facilmente estendido para o caso multiclasse, mas, vale notar que a suposição de matriz de covariância comum deve ser assumida.

Modelos discriminativos probabilísticos

Nos modelos discriminativos probabilísticos, utilizamos uma abordagem diferente dos modelos apresentados anteriormente. Para esta classe de modelos, utilizamos a forma funcional dos modelos lineares generalizados explicitamente para determinar seus parâmetros de forma direta através da função de verossimilhança definida por meio das distribuições condicionais $p(C_k|\mathbf{x})$ (BISHOP, 2006).

Neste momento, vale ressaltar que tanto o modelo generativo quanto o modelo discriminativo possuem suas vantagens, e a escolha por uma das duas abordagens dependerá do problema e dos objetivos que se quer alcançar. A abordagem generativa por exemplo, por permitir a obtenção da distribuição marginal dos dados $p(\mathbf{x}) = \sum_j p(\mathbf{x}|C_j)p(C_j)$, pode ser útil para lidar com dados faltantes, *outliers* e pontos de dados não-rotulados (WILLIAMS; RASMUSSEN, 2006). Por outro lado, se $\mathbf{x}$ possui alta dimensionalidade, será necessário utilizar conjuntos de treinamento grandes para obter as densidades de classe-condicionais para uma acurácia razoável. Mas, se apenas desejamos resolver um problema de classificação, a abordagem discriminativa é atrativa, pois visa modelar diretamente aquilo que desejamos $p(C_k|\mathbf{x})$, e contém em geral menos parâmetros a serem estimados (BISHOP, 2006).

Nos modelos generativos, determinamos separadamente os parâmetros das densidades de classe-condicionais e distribuições a priori de cada classe pelo método da máxima verossimilhança, para então utilizarmos o teorema de Bayes para obtermos as distribuições a posteriori de cada classe $p(C_k|\mathbf{x})$. No modelo discriminativo, estimaremos os parâmetros de $p(C_k|\mathbf{x})$ de forma direta através de algum método de estimação. Apresentaremos de início a forma funcional de $p(C_k|\mathbf{x})$ quando $k = 2$. Assim, temos para a classe C_1 a seguinte expressão

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}), \tag{3.26}$$

com $p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$. Aqui $\sigma(.)$ é uma função sigmóide, podendo ser a função logística, ou também, a função de distribuição acumulada da Gaussiana, conhecida como probito. No contexto dos modelos lineares generalizados, a função logística é a função de ligação canônica do modelo atribuído a dados categóricos, e tal modelo é normalmente conhecido por regressão logística. Utilizaremos este modelo como base para desenvolver o modelo discriminativo. Um tratamento utilizando a função probito (conhecida como regressão probito) pode ser encontrado em Bishop (2006).

Utilizando a expressão (3.6) para a regressão logística, necessitaremos futuramente de sua derivada que pode ser obtida da seguinte forma

$$\frac{d\sigma}{da} = \frac{\exp(-a)}{(1+\exp(-a))^2} = \sigma(a)(1-\sigma(a)), \tag{3.27}$$

que podemos expressar em termos da função logística. Assim, dado um conjunto de dados $\mathscr{D} = \{(t_i, y_i)|i = 1,\ldots,n\}$, com os rótulos $t_i \in \{0,1\}$ e definindo $y_i = p(C_1|\mathbf{x}_i)$, temos a seguinte função de verossimilhança

$$p(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{n} y_i^{t_i}(1-y_i)^{1-t_i}, \tag{3.28}$$

em que $\mathbf{t} = [t_1,\ldots,t_n]^T$. Tendo esta função de verossimilhança, devemos a partir deste momento proceder com a maximização da expressão acima para encontrar os devidos parâmetros. Theodoridis e Koutroumbas (2009) citam que alguns algoritmos de otimização com resultados comentados podem ser encontrados em Anderson (1982) e McLachlan (1992). Contudo, há um algoritmo padrão muito utilizado para encontrar as soluções dos parâmetros, chamado Mínimos Quadrados Iterativamente Reponderados (IRLS) *(Iteratively Reweighted Least Squares)*, baseado no método de Newton-Raphson. Apesar de, como mostra Minka (2003), haver algoritmos de otimização mais rápidos, nos atentaremos a mostrar apenas a forma do algoritmo IRLS.

Para apresentar a forma do algoritmo IRLS, é conveniente fazermos uma transformação não linear das entradas $\mathbf{x}$ e aplicar o modelo neste espaço transformado através de um vetor de funções de base $\boldsymbol{\phi}(\mathbf{x})$, com dimensão M. As fronteiras de decisões resultantes serão lineares no espaço de entradas $\boldsymbol{\phi}$, e isto corresponderá a fronteiras de decisões não lineares no espaço de entradas originais $\mathbf{x}$ (BISHOP, 2006). Utilizaremos esta abordagem para podermos apresentar o IRLS na sua forma mais ampla, em termos de funções de base.

Seguindo o procedimento feito por Bishop (2006), utiliza-se uma função de erro dada pelo negativo do log da função de verossimilhança, chamada de função de erro de entropia cruzada. Esta função de erro é baseada em fundamentos da teoria da informação, e referências como Viterbi e Omura (1979) e Cover e Thomas (1991) podem ser consutadas para obtenção de mais detalhes. Esta função é dada a seguir

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}) = -\sum_{i=1}^{n}\{t_i \log y_i + (1-t_i)\log(1-y_i)\}. \tag{3.29}$$

Tomando o gradiente da função erro com relação a $\mathbf{w}$, e fazendo uso da derivada da função logística dada na expressão (3.27), temos a seguinte forma simplificada

$$\nabla E(\mathbf{w}) = -\sum_{i=1}^{n}\{-t_i(1-y_i)\boldsymbol{\phi}_i + (1-t_i)y_i\boldsymbol{\phi}_i\} = \sum_{i=1}^{n}(y_i - t_i)\boldsymbol{\phi}_i \tag{3.30}$$

em que $y_i = \sigma(a_i)$, $\boldsymbol{\phi}_i = \boldsymbol{\phi}(\mathbf{x}_i)$, com $a_i = \mathbf{w}^T\boldsymbol{\phi}_i$. A expressão resultante em (3.30) não possui uma solução fechada, pois y_i é uma função não linear. Com isso, é necessário o uso de métodos numéricos para minimizar a função erro. O método mais comum para executar essa tarefa é o método de Newton-Raphson que, como já mencionado, dará a base para o algoritmo IRLS. A forma do algoritmo de Newton-Raphson é dada a seguir

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \mathbf{H}^{-1}\nabla E(\mathbf{w}), \tag{3.31}$$

em que $\mathbf{H}$ é a matriz hessiana, composta das segundas derivadas de $E(\mathbf{w})$ com relação a $\mathbf{w}$. A matriz hessiana possui uma dependência em $\mathbf{w}$ através dos termos $y_i(1-y_i)$. Como $0 < y_i < 1$, pela propriedade da função logística, pode-se mostrar que $\mathbf{u}^T\mathbf{H}\mathbf{u} > 0$ para qualquer vetor arbitrário $\mathbf{u}$, não nulo, e assim a matriz hessiana é positiva definida, e por consequência possui mínimo único (BISHOP, 2006). Antes de aplicar o método de Newton-Raphson, vamos expressar o gradiente da função erro junto com a respectiva hessiana na forma matricial

$$\nabla E(\mathbf{w}) = \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) \tag{3.32}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{i=1}^{n} y_i(1-y_i)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T = \boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi}, \tag{3.33}$$

em que a matriz $\boldsymbol{\Phi}$ é de dimensão $n \times M$ e $\mathbf{R}$ é uma matriz diagonal $n \times n$ composta dos elementos $R_{ii} = y_i(1-y_i)$. Aplicando o método de Newton-Raphson à função erro, obtemos a seguinte expressão

$$\begin{aligned}
\mathbf{w}^{\tau+1} &= \mathbf{w}^{\tau} - (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) \\
&= (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})(\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\mathbf{w}^{\tau} + (\boldsymbol{\Phi}^t\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) \\
&= (\boldsymbol{\Phi}^t\mathbf{R}\boldsymbol{\Phi})^{-1}[\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi}\mathbf{w}^{\tau} - \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t})] \\
&= (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{R}\mathbf{z},
\end{aligned} \tag{3.34}$$

em que $\mathbf{z}$ é um vetor n-dimensional dado por $\mathbf{z} = \boldsymbol{\Phi}\mathbf{w}^{\tau} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$. A expressão final dada em (3.34) é a forma do sistema de equações

normais[4] para a solução de mínimos quadrados ponderado pela matriz $\mathbf{R}$. Devido ao fato da matriz diagonal $\mathbf{R}$ não ser constante, mas depender do vetor de parâmetros $\mathbf{w}$ para poder ser atualizada, o sistema de equações normais é atualizado iterativamente, onde em cada iteração do algoritmo, utilizamos um novo vetor de parâmetros $\mathbf{w}$ para atualizar a matriz $\mathbf{R}$.

3.2 Classificação com Processo Gaussiano - GPC

Os modelos de classificação de Processos Gaussianos (o mesmo também se aplica aos modelos de Processo t-Student) são não-paramétricos e probabilísticos, diferenciando daqueles que apenas fornecem uma guia de a qual classe o padrão provavelmente pertence (WILLIAMS; RASMUSSEN, 2006). Em classificação com Processo Gaussiano, aproveitamos a forma funcional do modelo dado em (3.26)[5], onde substituímos a forma linear dos argumentos de entrada da função sigmóide por um processo latente $f(\mathbf{x})$, cuja distribuição a priori será um Processo Gaussiano. A saída dada pelo processo latente é então comprimida pela função sigmoidal, ou seja, $\boldsymbol{\pi}(\mathbf{x}) = p(y = 1|\mathbf{x}) = \boldsymbol{\sigma}(f(\mathbf{x}))$. Esta compressão torna-se essencial para possibilitar respotas probabilísticas de classe. Na Figura 14 ilustramos como obter este processo utilizando a função logística, que será utilizada ao longo deste trabalho para o desenvolvimento do classificador de GP binário. No decorrer do texto, estaremos seguindo a mesma notação e roteiro de apresentação de Williams e Rasmussen (2006) para GPC.

Vale ressaltar que por $\boldsymbol{\pi}$ ser uma função determinística de f, que é estocástica, $\boldsymbol{\pi}$ também se tornará estocástica. Ao contrário do caso de regressão, em classificação com Processo Gaussiano pode-se assumir que o processo latente é livre de ruídos e que todos os pontos de dados de treinamento são corretamente rotulados. Bishop (2006) comenta que é conveniente adicionar um termo ruidoso (chamado de "*jitter*") por razões numéricas, governado de tal forma que as matrizes de covariância sejam positivas definidas.

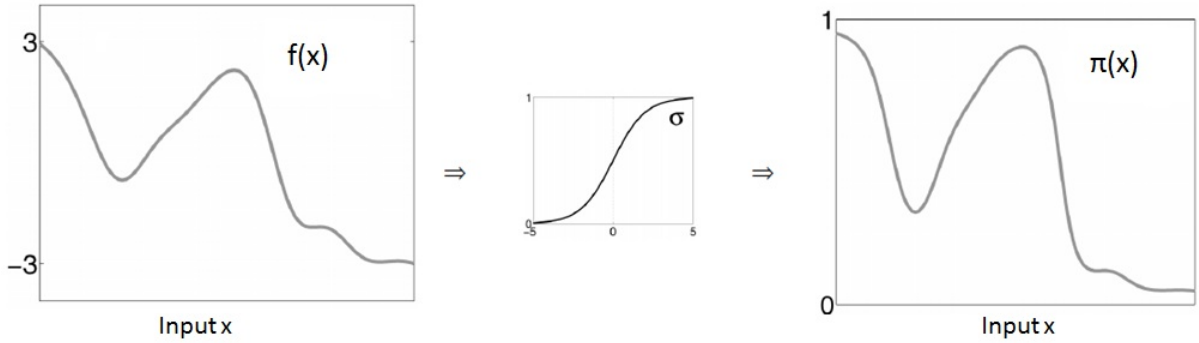
O papel de f no classificador de Processo Gaussiano é apenas de permitir uma conveniente formulação. Não estamos interessados em observar seus valores, mas sim em

---

[4] O sistema de equações normais é o resultado padrão obtido pelo método dos mínimos quadrados para o modelo de regressão linear dado por $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$, onde $\mathbf{t} \in \mathbb{R}$.

[5] A forma funcional dada em (3.26) normalmente é utilizada para o problema de classificação binária, para o problema de classificação multiclasse utiliza-se a função *softmax* dada em (3.8).

Figura 14 – Ilustração de como obter $\pi(\mathbf{x})$ de $f(\mathbf{x})$. Imagem à esquerda representa a curva de uma função latente $f(\mathbf{x})$. Imagem central representa a curva da função logística dada por (3.6). Imagem à direita representa o resultado da compressão para obtenção da probabilidade de classe $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$.



Fonte: Imagem adaptada de Williams e Barber (1998).

π, e em especial para um caso de teste π_* para uma dada entrada $\mathbf{x}_*$. Para obter isso, temos um passo a passo para possibilitar tal inferência.

Primeiramente, cabe ressaltar que os modelos que desenvolveremos nesta Seção fundamentam-se no modelo linear discriminativo apresentado anteriormente. Iniciaremos portanto, apresentando o modelo linear discriminativo binário (duas classes) por meio da abordagem bayesiana para facilitar a compreensão do desenvolvimento do modelo de classificação de Processo Gaussiano. O problema para o caso multiclasse seguirá o mesmo raciocínio, e por consequência é uma extensão natural do caso binário. Vamos definir os seguintes rótulos[6] $y = 1$ para a classe C_1, e $y = -1$ para a classe C_2, e definir $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$. Dessa forma, temos a seguinte expressão para a forma funcional do modelo

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(f(\mathbf{x})) = \sigma(\mathbf{x}^T\mathbf{w}), \tag{3.35}$$

com $p(y = -1|\mathbf{x}, \mathbf{w}) = 1 - p(y = 1|\mathbf{x}, \mathbf{w})$.

A função de verossimilhança pode ser escrita de forma conveniente utilizando a propriedade de simetria da função logística em (3.7). Assim, dado um conjunto de dados $\mathscr{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, assumindo que os rótulos y_i são gerados independentemente condicionais em $f(\mathbf{x})$, podemos obter a seguinte expressão

[6] Essa forma de definição de rótulo -1/1 é apresentada apenas para questões de introdução do modelo GPC, nas aplicações futuras deste trabalho utilizaremos a rotulação 0/1, sem nenhum prejuízo ao modelo GPC binário que será apresentado.

$$p(y_i|\mathbf{x}_i,\mathbf{w}) = \sigma(\mathbf{x}_i^T\mathbf{w}) = \sigma(y_i f_i), \quad \forall \quad i = 1,\ldots,n, \tag{3.36}$$

em que $f_i = f(\mathbf{x}_i) = \mathbf{x}_i^T\mathbf{w}$.

Atribuindo uma distribuição a priori Normal Multivariada para $\mathbf{w}$ por simplicidade, ou seja $\mathbf{w} \sim N_{D\times 1}(\mathbf{0},\boldsymbol{\Sigma}_\mathbf{w})$, temos a seguinte log posteriori não normalizada

$$\begin{aligned} p(\mathbf{w}|\mathbf{X},\mathbf{y}) &\propto \log\, \mathrm{p}(\mathbf{y}|\mathbf{X},\mathbf{w}) + \log\, \mathrm{p}(\mathbf{w}) \\ &\propto \sum_{i=1}^{n} \log\sigma(y_i f_i) - \frac{D+1}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}_\mathbf{w}| - \frac{1}{2}\mathbf{w}^T\boldsymbol{\Sigma}_\mathbf{w}\mathbf{w}. \end{aligned} \tag{3.37}$$

Essa posteriori não possui uma forma analiticamente tratável, e por isso métodos de aproximação serão necessários. Pode-se mostrar que a posteriori da expressão (3.37) possui máximo único. Williams e Rasmussen (2006) argumentam que funções sigmoides, tais como a logística e a cumulativa Gaussiana, proporcionam uma log verossimilhança que é função côncava de $\mathbf{w}$. E como a penalidade quadrática em $\mathbf{w}$ advinda da distribuição a priori, é também côncava, logo a log posteriori possui solução única. Essa informação é importante quando se aplica métodos de aproximação local (próximo à moda da distribuição) para se obter inferências para a distribuição a posteriori. Da mesma forma, tanto os modelos de classificação de GP quanto os de classificação de TP, não possuem resultados que são analiticamente tratáveis (distribuição a posteriori, distribuição preditiva e probabilidade preditiva de classe). Como ferramenta utilizada para obter aproximações para estes classificadores, serão utilizados métodos MCMC, que apresentaremos mais adiante.

Prosseguindo, para fazermos predições para uma nova entrada de teste $\mathbf{x}_*$, procedemos da seguinte forma:

$$p(y_* = 1|\mathbf{x}_*,\mathscr{D}) = \int p(y_* = 1|\mathbf{w},\mathbf{x}_*)p(\mathbf{w}|\mathscr{D})d\mathbf{w}, \tag{3.38}$$

em que $p(y_* = 1|\mathbf{w},\mathbf{x}_*) = \sigma(\mathbf{x}_*^T\mathbf{w})$.

Nesse ponto, tendo apresentado as distribuições a posteriori e preditiva para um caso de teste, para o modelo binário discriminativo linear, podemos agora derivar o modelo de GP para classificação por meio destes resultados. Assim, substituindo em (3.37) a parte linear em $\mathbf{w}$ por um Processo Gaussiano, necessitamos encontrar em seguida a distribuição a posteriori sobre o processo latente. A log posteriori pode ser dada da

seguinte forma

$$\begin{aligned} \log \mathrm{p}(\mathbf{f}|\mathbf{X},\mathbf{y}) &= \log \mathrm{p}(\mathbf{y}|\mathbf{f}) + \log \mathrm{p}(\mathbf{f}|\mathbf{X}) \\ &= -\sum_{i=1}^{n} \log(1+\exp(-y_i f_i)) - \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi. \end{aligned} \tag{3.39}$$

Com a distribuição a posteriori em mãos, o procedimento subsequente é encontrar a distribuição preditiva referente a um caso de teste. Esta distribuição pode ser adquirida da seguinte forma

$$p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*) = \int p(f_*|\mathbf{X},\mathbf{x}_*,\mathbf{f})p(\mathbf{f}|\mathbf{X},\mathbf{y})d\mathbf{f}. \tag{3.40}$$

Por fim, o passo seguinte após obtermos a distribuição preditiva para f_*, é produzir uma predição probabilística para um caso de teste, que pode ser obtida da seguinte maneira

$$\bar{\pi}_* = p(y_* = 1|\mathbf{X},\mathbf{y},\mathbf{x}_*) = \int \sigma(f_*)p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*)df_*. \tag{3.41}$$

Classificação multiclasse

Prosseguindo, apresentaremos agora o caso de classificação multiclasse. Considere o seguinte vetor extendido $\mathbf{f}$ com K processos latentes em cada um dos n pontos de dados

$$\mathbf{f} = [f_1^1,\ldots,f_n^1,f_1^2,\ldots,f_n^2,\ldots,f_1^K,\ldots,f_n^K]^T. \tag{3.42}$$

Este vetor tem dimensão Kn. Na notação acima, o sobrescrito refere-se a quantidades pertencentes a uma classe particular k para $k = 1,\ldots,K$, e o subescrito refere-se ao ponto de treinamento $i = 1,\ldots,n$. Dessa forma o vetor de valores latentes de uma classe particular é $\mathbf{f}_i$. Como distribuição a priori para $\mathbf{f}$ em (3.42) temos $\mathbf{f} \sim N_{Kn}(\mathbf{0},\mathbf{K})$, em que $\mathbf{0}$ é um vetor nulo de dimensão Kn e $\mathbf{K}$ é uma matriz de dimensão $Kn \times Kn$ diagonal em blocos nas matrizes $\mathbf{K}_1,\ldots,\mathbf{K}_K$, cada uma tendo dimensão $n \times n$.

Cada uma das matrizes $\mathbf{K}_k$ representa a covariância entre os valores da função latente dentro da classe k. Note que a matriz $\mathbf{K}$ é diagonal em blocos, pois supomos que não existe correlação entre classes, ou seja, que os K processos latentes são independentes (WILLIAMS; BARBER, 1998). Vale também ressaltar que supor K processos latentes em um problema com K classes é redundante, mas, como argumenta Bernardo *et al.* (1998), torna-se conveniente proceder dessa forma para evitar assimetrias arbitrárias na distribuição a priori.

Para obtermos as probabilidades de classe, definimos um vetor $\mathbf{y}$ de mesmo comprimento que $\mathbf{f}$, que para cada ponto de treinamento $i = 1,\ldots,n$, possui valor 1 na posição da classe verdadeira e 0 nas $K-1$ posições restantes. As probabilidades de classe são definidas através da função *softmax* (equação (3.8)) da seguinte forma

$$p(y_i^k|\mathbf{f}_i) = \pi_i^k = \frac{\exp(f_i^k)}{\sum_{k'}\exp(f_i^{k'})}, \tag{3.43}$$

em que $\sum_{k'}\pi_i^{k'} = 1$, e $\boldsymbol{\pi}$ tem dimensão Kn com entradas π_i^k. A log posteriori não normalizada possui a seguinte forma

$$\log\, \mathrm{p}(\mathbf{f}|\mathbf{X},\mathbf{y}) = -\frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} + \mathbf{y}^T\mathbf{f} - \sum_{i=1}^{n}\log(\sum_{k=1}^{K}\exp f_i^k) - \frac{1}{2}\log|\mathbf{K}| - \frac{Kn}{2}\log 2\pi. \tag{3.44}$$

Para fazermos predições para uma nova quantidade $\mathbf{x}_*$, necessitamos encontrar a distribuição de $p(\mathbf{f}_*|\mathbf{X},\mathbf{x}_*,\mathbf{y})$, onde $\mathbf{f}_* = [f_*^1,\ldots,f_*^K]^T$. Dada a posteriori em (3.44), a distribuição preditiva pode ser obtida de forma similar ao caso binário, resultando na seguinte expressão

$$p(\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{X},\mathbf{x}_*,\mathbf{f})p(\mathbf{f}|\mathbf{X},\mathbf{y})d\mathbf{f}. \tag{3.45}$$

Para podermos obter uma predição média probabilística $\bar{\boldsymbol{\pi}}_*$ no caso multiclasse, um caminho simples seria amostrar pontos de dados da distribuição em (3.45) e calcular o valor da função *softmax* (3.43) em cada ponto. Com estes valores, calculamos a média de Monte Carlo para obter uma estimativa da predição média.

3.3 Classificação com Processo t-Student - TPC

O modelo de classificação utilizando Processos t-Student surge como uma proposta de classificador robusto a *outliers*. Em um cenário idealizado, assume-se que os rótulos de classe estão atribuídos corretamente aos dados. Contudo, em situações práticas, pode haver a ocorrência de fatores desconhecidos que influenciam na atribuição do rótulo, onde ao invés de observarmos o rótulo verdadeiro y, estamos observando um rótulo corrompido $\tilde{y}$. Como sugere Bootkrajang (2016), garantir uma rotulação perfeita dos dados se tornará muito dispendiosa, devido a grande complexidade dos problemas de classificação atuais. Assim, torna-se necessário preocupar-se em propor métodos e/ou técnicas de classificação que levem em consideração este tipo de estrutura nos dados, para

tentar amenizar os seus efeitos. Neste ínterim, a proposta do classificador de Processo t-Student torna-se relevante como uma forma de lidar com este tipo de problema.

Prosseguindo, vamos introduzir o classificador de Processo t-Student proposto neste trabalho. O fundamento deste classificador é exatamente o mesmo ao que foi utilizado para os classificadores de GP, porém, ao invés de utilizarmos um Processo Gaussiano como distribuição a priori, utilizaremos um Processo t-Student. Assim, para o caso binário, utilizando como distribuição a priori a forma da expressão (2.46), a log posteriori não normalizada para o classificador t-Student é da seguinte forma

$$\begin{aligned} \log \mathrm{p}(\mathbf{f}|\mathbf{X},\mathbf{y}) &= \log \mathrm{p}(\mathbf{y}|\mathbf{f}) + \log \mathrm{p}(\mathbf{f}|\mathbf{X}) \\ &= -\sum_{i=1}^{n} \log(1+\exp(-y_i f_i)) + \log \mathrm{B} - \frac{1}{2}\log|\mathbf{K}| - \frac{v+n}{2}\log\left(1+\frac{1}{v}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right), \end{aligned} \tag{3.46}$$

em que $\mathrm{B} = \frac{\Gamma[(v+n)/2]}{\Gamma(v/2)v^{(n/2)}\pi^{(n/2)}}$.

Para o caso multiclasse, a log posteriori para o classificador TP pode ser dada por

$$\log \mathrm{p}(\mathbf{f}|\mathbf{X},\mathbf{y}) = \log \mathrm{B}_2 + \mathbf{y}^T\mathbf{f} - \sum_{i=1}^{n}\log(\sum_{k=1}^{K}\exp f_i^k) - \frac{1}{2}\log|\mathbf{K}| - \frac{v+Kn}{2}\log\left(1+\frac{1}{v}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right), \tag{3.47}$$

em que $\mathrm{B}_2 = \frac{\Gamma[(v+Kn)/2]}{\Gamma(v/2)v^{(Kn/2)}\pi^{(Kn/2)}}$.

Para fazermos predições referentes a um caso de teste, procedemos da mesma maneira como quando tratávamos sobre o classificador de GP. Primeiro, em relação ao caso binário, encontramos a distribuição do processo latente referente a um caso de teste $p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*)$ utilizando a expressão da integral em (3.40). Em seguida, podemos obter uma predição probabilística utilizando a expressão em (3.41). Para o caso multiclasse, os passos são equivalentes ao caso binário, primeiro encontrando a distribuição dos processos latentes referentes a uma nova quantidade de teste $\mathbf{x}_*$ utilizando a expressão (3.45), e por fim, obtém-se uma predição probabilística de classe através de estimativas de Monte Carlo, como explicado na Seção anterior.

Para os problemas de classificação, as integrais (como as expressões (3.40) e (3.41)) resultantes dos processos de inferência não são mais analiticamente tratáveis, uma vez que a verossimilhança não é mais Gaussiana. Métodos de inferência aproximada devem ser utilizados. Uma abordagem que poderíamos utilizar, seriam os métodos MCMC, que são conhecidos por serem assintoticamente exatos, e lidam muito bem com problemas complexos. Apesar de na prática os recursos computacionais limitarem as amostras MCMC

geradas para aproximação, extensões de métodos MCMC têm possibilitado uma melhor eficiência computacional. Exemplo disso é a técnica de Monte Carlo Hamiltoniano, que se valendo de informações do gradiente da distribuição a posteriori não normalizada, permite inspecionar o espaço paramétrico do problema em estudo de forma mais eficiente, diminuindo o esforço computacional em gerar amostras independentes da posteriori.

Como mencionado no Capítulo 1, utilizaremos os métodos de amostragem MCMC para desenvolvermos os modelos de classificação com Processos Gaussianos e t-Student. Para este trabalho, decidiu-se utilizar o NUTS, que é uma extensão do método HMC. A seguir, descreveremos de forma sucinta este último método, para então adiante, apresentarmos o NUTS.

3.4 Monte Carlo Hamiltoniano - HMC

O Monte Carlo Hamiltoniano usa simulações em um sistema físico fictício para gerar novos estados propostos da distribuição a posteriori. Esses novos estados são gerados de acordo com a regra de Métropolis. Neal (2011) resume a construção do método HMC em alguns passos. O primeiro consiste em definir uma função hamiltoniana em termos da distribuição de probabilidade cujos valores iremos amostrar. Segundo, adicionamos às variáveis que estamos interessados em amostrar (que neste caso, são variáveis nomeadas de "posição") variáveis auxiliares, chamadas de "variáveis de momento". Por fim, o HMC alterna atualizações simples das variáveis de momento através da regra de Métropolis, na qual um novo estado da cadeia é proposto por uma trajetória simulada de acordo com a dinâmica hamiltoniana.

Pode ocorrer de o novo estado proposto pela dinâmica hamiltoniana vir a ser muito distante do estado atual, mas, este vem a manter uma alta probabilidade de aceitação pela regra de Métropolis, pelo fato de o HMC utilizar a informação do gradiente da distribuição a posteriori. Isto evita a lenta exploração do espaço paramétrico que normalmente ocorre em abordagens MCMC com simples distribuições propostas de caminho aleatório, tais como a técnica de Métropolis-Hastings (NEAL, 2011).

Em uma aplicação não física[7] para MCMC, as variáveis de posição do

[7] O HMC desenvolvidio em Duane *et al.* (1987) utiliza conceitos da mecânica hamiltoniana e foi proposto para fazer simulações da teoria de campo da cromodinâmica quântica.

hamiltoniano corresponderão às variáveis de interesse. A energia potencial será dada pelo logaritmo negativo da densidade das variáveis de posição. E as variáveis de momento serão introduzidas artificialmente, uma para cada variável de posição. Dessa forma, considerando $Q(\boldsymbol{\theta}|\mathscr{D})$ a distribuição a posteriori não normalizada do vetor de parâmetros $\boldsymbol{\theta}$, todo novo estado gerado para a cadeia de $\boldsymbol{\theta}$, sob a ótica hamiltoniana, é interpretado como a localização de uma partícula com posição $\mathbf{q}$, vetor d-dimensional, e momento $\mathbf{p}$, também um vetor d-dimensional (NEAL, 2011). Segundo Kuss (2006), novos estados são

gerados utilizando a simulação discreta de uma trajetória da dinâmica hamiltoniana. O hamiltoniano que descreve o sistema é dado por

$$H(\mathbf{q},\mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}) = -\ln Q(\boldsymbol{\theta}|\mathscr{D}) + K(\mathbf{p}), \tag{3.48}$$

em que $K(\mathbf{p})$ para aplicações MCMC pode ser dada por $K(\mathbf{p}) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2$, em que $\mathbf{M}$ é uma matrix $d \times d$ simétrica positiva definida (NEAL, 2011). A energia cinética $K(\mathbf{p})$ é descrita de forma conveniente para representar o núcleo de uma distribuição Normal Multivariada de vetor média nulo e matriz de covariância $\mathbf{M}$.

Para podermos amostrar da distribuição a posteriori de $\boldsymbol{\theta}$, necessitamos "traduzir" esta distribuição junto com a introdução das variáveis de momento, em uma função de energia para obtermos uma distribuição conjunta de $p(\boldsymbol{\theta},\mathbf{p})$, e assim haver a possibilidade de gerar as simulações hamiltonianas. Utilizando os conceitos da Mecânica Estatística, podemos propor uma distribuição de probabilidade da seguinte forma para $p(\boldsymbol{\theta},\mathbf{p})$

$$p(\theta,\mathbf{p}) = \frac{1}{Z}\exp\left(\frac{-H(\boldsymbol{\theta},\mathbf{p})}{T}\right), \tag{3.49}$$

em que $-H(\boldsymbol{\theta},\mathbf{p})$ é uma função de energia, Z é uma constante de normalização positiva e T[8] é considerada a temperatura do sistema. Considerando que o hamiltoniano do sistema dado em (3.48) constitui-se da soma das energias cinéticas e potencial, e que $T = 1$, a distribuição de probabilidade dada em (3.49) pode ser escrita na seguinte forma proporcional

$$P(\boldsymbol{\theta},\mathbf{p}) \propto \exp\left(-U(\mathbf{q})\right)\exp\left(-K(\mathbf{p})\right) = Q(\boldsymbol{\theta}|\mathscr{D})\exp\left(-\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2\right). \tag{3.50}$$

Por (3.50), constatamos que a distribuição conjunta de $p(\boldsymbol{\theta},\mathbf{p})$ pode ser fatorada

[8] A unidade de temperatura aqui é a mesma considerada em Neal (2011), e omitiremos qualquer menção à mesma, sem ocorrência de danos ao desenvolvimento da apresentação do HMC.

pelo produto das distribuições marginais, e assim concluímos que a distribuição de $U(\mathbf{q})$ e $K(\mathbf{p})$ são independentes. A ideia a partir de agora consiste em gerar valores aleatórios da distribuição de $\mathbf{p}$, dada por uma distribuição Gaussiana Multivariada independente dos valores atuais das variáveis de posição. Em seguida, uma atualização de Métropolis é realizada usando dinâmica Hamiltoniana para propor um novo estado. Iniciando no estado atual $(\mathbf{q},\mathbf{p})$, a simulação do sistema é feita utilizando as equações de Hamilton

$$\dot{\boldsymbol{\theta}} = \frac{\partial H(\boldsymbol{\theta},\mathbf{p})}{\partial \mathbf{p}} = [\mathbf{M}^{-1}\mathbf{p}], \tag{3.51}$$

$$\dot{\mathbf{p}} = -\frac{\partial H(\boldsymbol{\theta},\mathbf{p})}{\partial \mathbf{q}} = -\frac{\partial \ln Q(\boldsymbol{\theta}|\mathscr{D})}{\partial \mathbf{q}}, \tag{3.52}$$

em que o gradiente da distribuição a posteriori não normalizada equivale a mudança no momento. A simulação do sistema é discretizada utilizando o método Leapfrog, que permite a reversibilidade da cadeia (KUSS, 2006). Esta discretização requer configurar alguns parâmetros, a saber: o número de passos L e o tamanho do passo $\boldsymbol{\varepsilon}$. Estes parâmetros devem ser otimizados de tal forma que proporcionem um bom desempenho do algoritmo HMC.

As variáveis de momento ao fim da trajetória L-passos proporcionam um novo estado da cadeia $(\mathbf{q}^*,\mathbf{p}^*)$. O estado proposto é então aceito como o próximo estado da cadeia de Markov com probabilidade[9]

$$\min[1,\exp(-H(\mathbf{q}^*,\mathbf{p}^*)+H(\mathbf{q},\mathbf{p}))]. \tag{3.53}$$

Se o estado proposto não for aceito, ele permanece no estado atual. Para obter mais detalhes sobre o algoritmo HMC e suas propriedades, pode-se consultar Kuss (2006), Neal (2011), Betancourt (2017).

3.4.1 No-U-Turn Sampler

No algoritmo HMC, para que a simulação da trajetória hamiltoniana se torne possível, as equações diferenciais de Hamilton em (3.51) e (3.52) são aproximadas pelo

[9] De acordo com a regra de Métropolis, os novos estados da Cadeia de Markov para o HMC seriam aceitos se $p \geq u$, onde u é um valor amostrado da distribuição Uniforme no intervalo unitário, e p é dado por (3.53).

método Leapfrog. Neste método há dois parâmetros de ajuste manual: o número de passos L, e o tamanho do passo $\boldsymbol{\varepsilon}$. Estes parâmetros precisam ser manuseados de tal forma a proporcionar um bom desempenho do HMC. Se L por exemplo, for muito pequeno, as trajetórias simuladas do hamiltoniano exibirão comportamento de caminho aleatório indesejados, se por outro lado, L for muito grande, o HMC fará cálculos desnecessários.

Para evitar a intervenção manual destes parâmetros , mantendo ainda assim um bom desempenho do HMC, Hoffman e Gelman (2014) propuseram um algoritmo

chamado de NUTS (*No-U-Turn Sampler*), que veio a ser uma extensão do Monte Carlo Hamiltoniano. No NUTS, a necessidade de configurar o número de passos L é eliminada, e o tamanho do passo $\boldsymbol{\varepsilon}$ é obtido por um esquema de média dual que também evita a intervenção manual. O NUTS funciona através de um algoritmo recursivo para gerar um conjunto de possíveis pontos candidatos que abrangem uma ampla faixa da distribuição de interesse, parando automaticamente quando ele volta a repetir os passos anteriores já realizados na simulação da trajetória.

Para as aplicações no presente trabalho, o software Stan (STAN DEVELOPMENT TEAM, 2020) que implementa o algoritmo NUTS será utilizado. O mesmo será manuseado via interface do software R (R CORE TEAM, 2020).

3.4.2 GPC e TPC via HMC

Para os modelos de classificação utilizando processos latentes, tais como os Processos Gaussianos e t-Student, o esquema de amostragem através do HMC pode ser sintetizado em alguns passos. Primeiro, obtemos a distribuição a posteriori dos processos latentes dada por

$$p(\mathbf{f}|\mathbf{y}, \boldsymbol{\psi}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\psi}), \tag{3.54}$$

em que $\boldsymbol{\psi}$, como já mencionado anteriormente, é um conjunto de hiperparâmetros presentes na função de covariância da distribuição a priori de $\mathbf{f}$. A distribuição de $p(\mathbf{y}|\mathbf{f})$ é a função de verossimilhança, que relaciona-se de forma não-linear em $\mathbf{f}$, para obtermos as probabilidades de classe, e que pode ser de dois tipos: a função logística dada em (3.6) se o problema de classificação é binário, ou a função softmax (3.8) se o problema refere-se a uma tarefa de classificação multiclasse. Normalmente em aplicações práticas utilizando MCMC, assumimos que todos os parâmetros do modelo são desconhecidos. Dessa forma, conforme

Kuss (2006), assumindo diferenciabilidade da verossimilhança e das distribuições a priori, amostras podem ser geradas usando a seguinte posteriori conjunta

$$p(\mathbf{f}, \boldsymbol{\psi}|\mathbf{y}, \boldsymbol{\delta}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\psi})p(\boldsymbol{\psi}|\boldsymbol{\delta}), \tag{3.55}$$

em que $\boldsymbol{\delta}$ seriam os hiper-hiper-parâmetros da distribuição de $p(\boldsymbol{\psi}|\boldsymbol{\delta})$. Para implementação HMC, devemos calcular as derivadas da distribuição a posteriori dada em (3.55). Para isso, é conveniente utilizar a log transformação dessa distribuição, que resulta na seguinte forma

$$\log \mathrm{p}(\mathbf{f}, \boldsymbol{\psi}|\mathbf{y}, \boldsymbol{\delta}) \;\; = \;\; \log \mathrm{p}(\mathbf{y}|\mathbf{f}) + \log \mathrm{p}(\mathbf{f}|\mathbf{X}, \boldsymbol{\psi}) + \log \mathrm{p}(\boldsymbol{\psi}|\boldsymbol{\delta}). \tag{3.56}$$

Como mencionado na apresentação do algoritmo HMC, a log posteriori do modelo normalmente faz a função da energia potencial na dinâmica hamiltoniana, e os parâmetros do modelo as respectivas variáveis de posição. De posse de amostras retiradas da distribuição a posteriori conjunta em (3.55) pelo HMC, podemos fazer predições obtendo estimativas para a distribuição de $p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*, \boldsymbol{\psi})$ (ou $p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*, \boldsymbol{\psi})$, caso multiclasse) e consequentemente, para a distribuição de $\bar{\boldsymbol{\pi}}_*$.

Kuss (2006), referindo-se aos modelos de GP sob HMC, recomenda que as atualizações dos valores latentes $\mathbf{f}$ e os hiperparâmetros $\boldsymbol{\psi}$ sejam feitas de forma separada, devido aos diferentes custos computacionais que cada uma dessas atualizações acarreta. Por exemplo, quando atualizamos os valores latentes $\mathbf{f}$, os cálculos necessários em (3.56) são menos dispendiosos, limitando-se a calcular apenas a log verossimilhança, o termo quadrático proveniente da distribuição a priori de GP e a log priori de $\boldsymbol{\psi}$. Por outro lado, atualizar (3.56) para diferentes valores de $\boldsymbol{\psi}$ requer calcular a inversa e o determinante da matriz de covariância $\mathbf{K}$.

Tanto para o classificador de GP quanto para o de TP, será adicionado um termo ruidoso ("*jitter*") à matriz de covariância $\mathbf{K}$ $n \times n$ nas simulações futuras para melhorar o condicionamento da mesma. Para facilitar a tarefa de amostragem dos elementos de $\mathbf{f}$, utilizaremos a decomposição de Cholesky, que consiste em obtermos uma matriz triangular inferior $\mathbf{L}$ $n \times n$ tal que $\mathbf{K} = \mathbf{L}\mathbf{L}^T$. A decomposição de Cholesky também é útil para calcular a inversa da matriz $\mathbf{K}$ ou produtos tais que $\mathbf{K}^{-1}\mathbf{f}$, que são necessários para fazer predições para entradas de teste, e encontrar a log verossimilhança e suas derivadas (BERNARDO *et al.*, 1998).

Com a decomposição de Cholesky em mãos, reparametrizações lineares podem ser feitas para amostrar os valores latentes $\mathbf{f}$ por meio de uma melhor e mais eficiente implementação prática. Para os Processos Gaussianos, a representação estocástica é dada por $\mathbf{f} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z} \sim N_n(\boldsymbol{\mu}, \mathbf{K})$, em que $\mathbf{z}$ é um vetor de variáveis aleatórias independentes normalmente distribuídas de média zero e variância 1, e $\boldsymbol{\mu}$ é um vetor de médias. Para os Processos t-Student, a representação dos processos latentes é dada por $\mathbf{f} = \boldsymbol{\mu} + \sqrt{W}\mathbf{L}\mathbf{z} \sim MVT_n(v, \boldsymbol{\mu}, \mathbf{K})$ (HOFERT, 2013), em que $W = v/\chi_v^2$ (χ_v^2 denota uma variável seguindo distribuição Chi-quadrado com $v > 0$ graus de liberdade).

4 APLICAÇÃO EM DADOS REAIS

Este Capítulo destina-se à aplicação do modelo proposto em dados reais, cuja finalidade é avaliar o desempenho do modelo proposto comparado ao classificador GPC. As aplicações serão divididas em duas partes: em dados de classificação binária e dados de classificação multiclasse.

Referente à tarefa de classificação binária, serão utilizados dois bancos de dados bem conhecidos: Conjunto de dados de Coluna Vertebral (CV) e conjunto de dados *Wisconsin Diagnostic Breast Cancer* (WDBC)[1]. O primeiro banco de dados consiste de 310 observações 6-dimensional (ou seja, o vetor de características $\mathbf{x} \in \mathbb{R}^6$), cujas classes são "normal" (100 observações com rótulo "0") e "anormal" (210 observações com rótulo "1"). O segundo banco de dados é composto de 569 observações 30-dimensional ($\mathbf{x} \in \mathbb{R}^{30}$), cujas classes são "benigno" (357 observações com rótulo "0") e "maligno" (212 observações com rótulo "1"). Para a tarefa de classificação multiclasse, o banco de dados utilizado é referente aos dados de Coluna Vertebral (CV) versão multiclasse (3 classes). Este banco de dados é composto das mesmas variáveis que constam na versão binária, cujas classes agora são "Hérnia de disco" (60 observações), "Espondilolistese" (150 observações) e "Normal e Anormal" (100 observações). Os bancos de dados foram obtidos do repositório UCI - *Machine Learning Repository* (DUA; GRAFF, 2017).

Prosseguindo, para as simulações tanto na tarefa de classificação binária quanto no caso multiclasse, os seguintes classificadores foram aplicados: GPC, TPC com graus de liberdade $\nu = 2, 4$ e 8, e TPC com graus de liberdade sendo estimados a partir dos próprios dados. Vale relembrar que os graus de liberdade da distribuição t-Student controlam o peso das caudas, estando intrinsicamente ligado à robustez da distribuição, então é útil analisar como os graus de liberdade podem influenciar no desempenho do classificador TPC.

Para analisar como os classificadores se comportam na presença de *outliers*, procedemos da seguinte forma: 1) rodamos os códigos sem perturbações nas variáveis 2) depois rodamos cada código inserindo um certa $P\%$ de *outliers* nas variáveis de entrada. Essas porcentagens foram fixadas em $5\%, 10\%$ e 20%, conforme o esquema presente em

[1] Em tradução livre: "Diagnóstico de câncer de mama de Wisconsin".

Barreto e Barros (2016)[2].

Para gerarmos as perturbações nas variáveis de entrada, selecionamos aleatoriamente em cada variável, alguns pontos de dados de acordo com a porcentagem estabelecida *P%* para os *outliers*, e a estes somamos (ou subtraímos) um valor bem grande (de acordo com a escala de variação das variáveis). Posteriormente reescalamos as variáveis de entrada tal que elas tenham média 0 e variância 1[3]. As perturbações foram geradas apenas nos dados de treino, pois o objetivo é analisar como os *outliers* influenciam na habilidade de predição dos classificadores.

A função de covariância que utilizamos especifica covariâncias tais que pontos com entradas próximas dão origem a predições similares. Uma função de covariância que atende muito bem a esta tarefa é a Exponencial Quadrática como em Bernardo *et al.* (1998):

$$\mathrm{cov}(\mathbf{x}_i, \mathbf{x}_j) = \alpha^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{1}{\rho_d^2}(x_{i,d} - x_{j,d})^2\right) + \delta_{i,j}\sigma^2, \tag{4.1}$$

em que o vetor de parâmetros que iremos estimar é $\boldsymbol{\eta} = (\rho_1, \ldots, \rho_D, \alpha)$, sendo $\delta_{i,j}$ o delta de Kronecker que é 1 se $i = j$ e 0 caso contrário. A função de covariância em (4.1) é bastante relacionada à função de covariância ARD (Determinação de Relevância Automática (ARD)) de Neal (1996), desde que o inverso dos hiperparâmetros de comprimento-escala ρ_d determinam o quão relevante aquela entrada d é. O hiperparâmetro α especifica a escala geral (ou vertical) da distribuição a priori. O parâmetro σ^2 faz o papel do termo ruidoso "*Jitter*", e não será estimado, sendo a ele atribuído apenas um valor pequeno tal que permita um bom condicionamento da matriz de covariância. As distribuições a priori que utilizamos para os hiperparâmetros foram: $\alpha \sim N(0,1)$ e $\rho_d \sim Gamma(\frac{a}{2}, \frac{a}{2w})$, como em Neal (1997). Com esta parametrização para ρ_d, $E(\rho_d) = w$; quando a é pequeno, a distribuição a priori torna-se menos informativa, quando a é grande, ocorre o contrário. Para os graus de liberdade v do classificador TPC utilizamos a distribuição a priori: $Gamma(2, 0.1)$.

As simulações computacionais foram desenvolvidas utilizando a interface da

---

[2] Conforme Kim e Ghahramani (2008), isto não significa que o conjunto de treinamento tem 0%, 5%, 10% e 20% de *outliers*, pois alterar os valores das variáveis de entrada de determinado ponto observado, não necessariamente o torna um *outlier*. Mas, a tendência é que quanto mais aumentarmos as perturbações nos dados de treinamento, mais pontos *outliers* surgirão.

[3] Esse procedimento é comum em implementação de classificadores, principalmente quando as variáveis de entrada possuem intervalos de variação diferentes.

plataforma Stan no software R através do pacote 'rstan'. No Apêndice A consta uma pequena introdução sobre a linguagem de modelagem Stan, que pode ser utilizada para possibilitar um melhor esclarecimento dos códigos dos classificadores utilizados nas aplicações. Estes estão disponíveis no Apêndice B.

4.1 Aplicação 1: classificação binária

Para os dados de Coluna Vertebral, rodou-se cada um dos 5 classificadores 15 vezes. Para os dados WDBC por outro lado, rodou-se cada um dos classificadores apenas 10 vezes, devido ao fato do custo computacional para este banco de dados ser bem maior quando comparado aos dados de Coluna Vertebral, por conter muitos mais variáveis de entrada. Todos os experimentos apresentados neste capítulo, foram realizados em duas máquinas com as seguintes configurações: Intel Core i3-2370M com 2.40 gigahertz (4 gb RAM) e Intel Core i5-3470 com 3.20 gigahertz (8 gb RAM).

Os experimentos foram configurados da seguinte forma: para os dados de Coluna Vertebral, 600 iterações com *burn-in*[4] de 200 iterações a cada replicação, e para os dados WDBC, 500 iterações com *burn-in* de 150 iterações a cada replicação. Para os dados de Coluna Vertebral, Inicialmente foram utilizados 80% dos dados para "treinar" os modelos, e os 20% restantes para avaliar o desempenho dos mesmos. Para os dados WDBC, por conter mais pontos de dados, utilizou-se 50% dos dados para treino e 50% para testes. Ao final das replicações, retiramos a média e os respectivos desvios padrões das taxas de acerto, bem como a média e os desvios padrões das estimativas dos hiperparâmetros para cada modelo. Para cada quantidade $P\% = 0\%, 5\%, 10\%$ e 20% de *outliers*, os resultados ao final das simulações para os dados de CV e WDBC encontram-se respectivamente nas Tabelas 1 e 2. Os melhores resultados em cada um dos casos de $P\%$ de *outliers* são destacados em negrito.

Analisando os resultados da Tabela 1, percebemos que não há diferenças expressivas na performance do classificador TPC proposto em relação ao modelo GPC, apesar de se perceber que para $P\% = 5\%, 10\%$ e 20% as médias de taxa de acerto serem levemente maiores para o classificador TPC, a saber os classificadores TPC(v=4), TPC(v=2) e

[4] *Burn-in* é o período inicial antes da cadeia alcançar a distribuição de equilíbrio.

Tabela 1 – Desempenho dos classificadores GPC e TPC nos dados de CV.

P% outliers	0	5	10	20
	acerto(%)	acerto(%)	acerto(%)	acerto(%)
GPC	**85,59 ± 3,57**	85,59 ± 5,00	83,54 ± 3,35	80,53 ± 3,97
TPC (v=2)	84,83 ± 3,64	85,80 ± 5,14	**84,83 ± 3,15**	**81,18 ± 3,88**
TPC (v=4)	85,48 ± 3,70	**86,02 ± 4,78**	84,30 ± 3,13	80,32 ± 4,40
TPC (v=8)	**85,69 ± 3,80**	85,69 ± 5,02	84,08 ± 3,38	80,32 ± 4,05
TPC ($\hat{v}$)	85,37 ± 4,24	85,48 ± 5,02	84,19 ± 3,12	80,53 ± 4,24

Fonte: Autoria própria.

TPC(v=2) respectivamente.

Entretanto, quando analisamos as médias de taxa de acerto, apesar de não serem diferenças tão expressivas, os resultados trazem evidências que precisam ser consideradas e melhor exploradas para se averiguar possíveis melhorias em favor do classificador proposto, pois percebemos que as médias de taxa de acerto para o TPC apresentam um padrão de se manterem maior, à medida que aumentamos a quantidade de ruído nos dados. No gráfico à esquerda da Figura 15, para os dados de CV, temos uma visão espacial de como o desempenho dos classificadores se comporta em função da quantidade *P%* de *outliers* nos dados.

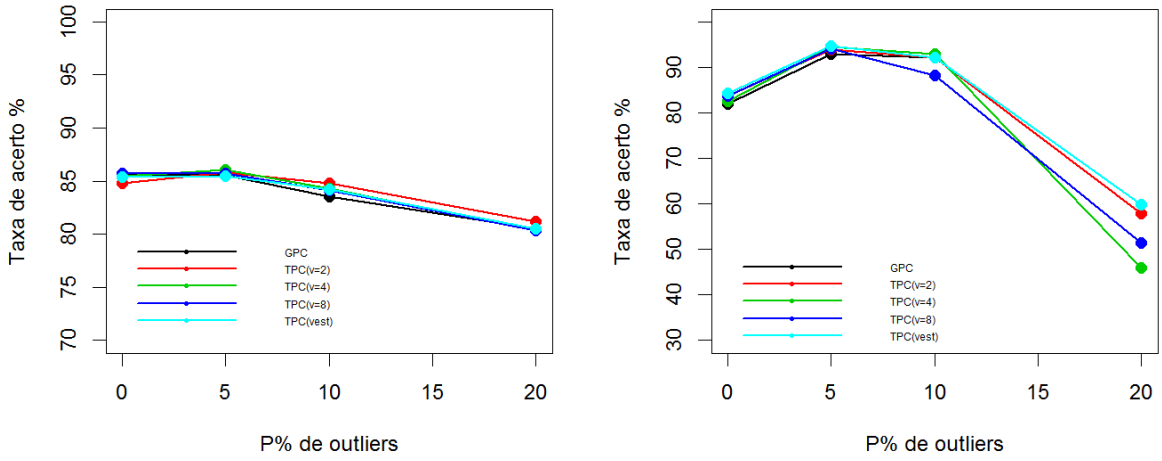
Tabela 2 – Desempenho dos classificadores GPC e TPC nos dados WDBC.

P% outliers	0	5	10	20
	acerto(%)	acerto(%)	acerto(%)	acerto(%)
GPC	81,96 ± 2,27	92,84 ± 1,63	92,21 ± 1,13	**59,89 ± 9,31**
TPC (v=2)	**83,71 ± 2,31**	93,89 ± 1,87	92,10 ± 2,49	57,85 ± 16,32
TPC (v=4)	82,59 ± 1,98	94,56 ± 1,67	**92,84 ± 1,44**	45,85 ± 17,98
TPC (v=8)	83,71 ± 2,48	94,07 ± 1,80	88,31 ± 9,56	51,36 ± 14,72
TPC ($\hat{v}$)	84,31 ± 5,26**	**94,59 ± 1,74***	92,17 ± 1,90	59,85 ± 13,83

Fonte: Autoria própria. *$\hat{v} = 2,77 \pm 1,01$;**$\hat{v} = 18,27 \pm 5,37$

Para os resultados da Tabela 2 (conjunto de dados WDBC), as performances dos classificadores apresentam um padrão diferente. O desempenho do classificador TPC é melhor que o GPC para $P\% = 0\%$ e 5%. Respectivamente, considerando a maior média de taxa de acerto, tem-se em cada um desses casos que os melhores classificadores foram o TPC ($v = 2$), TPC ($\hat{v} = 2,77$). A maior diferença entre as médias de taxa de acerto foi atingida pelo TPC($\hat{v} = 18,27$), quando avaliamos $P\% = 0\%$, com uma diferença de quase 3%. Para este conjunto de dados, percebe-se uma maior variabilidade nos resultados quando inserimos uma grande quantidade de ruído, como pode ser verificado para o caso $P\% = 20\%$, em que o alto desvio-padrão das médias de taxa de acerto demonstram a

Figura 15 – Média das taxas de acerto em função da quantidade P% de *outliers*. Gráfico à esquerda: dados de Coluna Vertebral. Gráfico à direita: dados WDBC.



Fonte: Autoria própria.

instabilidade de desempenho dos classificadores. No gráfico à direita da Figura 15, tem-se o desempenho dos classificadores para os dados WDBC em função da quantidade *P%* de *outliers*.

Tabela 3 – Estimação dos hiperparâmetros para os classificadores GPC e TPC para os dados de Coluna Vertebral.

P% outliers =0	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
GPC	2,86 ± 0,06	4,18 ± 0,16	3,80 ± 0,27	4,32 ± 0,33	3,07 ± 0,24	2,69 ± 0,29	0,94 ± 0,06
TPC(v=2)	1,50 ± 0,03	4,28 ± 0,19	4,13 ± 0,31	4,69 ± 0,36	3,58 ± 0,26	2,98 ± 0,34	1,13 ± 0,10
TPC(v=4)	1,88 ± 0,03	4,22 ± 0,20	4,04 ± 0,32	4,60 ± 0,39	3,45 ± 0,20	2,93 ± 0,36	1,08 ± 0,10
TPC(v=8)	2,25 ± 0,04	4,18 ± 0,23	3,96 ± 0,33	4,45 ± 0,37	3,30 ± 0,23	2,84 ± 0,34	1,02 ± 0,07
TPC($\hat{\nu} = 11,38 \pm 1,30$)	2,12 ± 0,05	4,19 ± 0,17	4,00 ± 0,26	4,50 ± 0,36	3,30 ± 0,23	2,87 ± 0,36	1,04 ± 0,09
P% outliers =5	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
GPC	2,86 ± 0,19	5,00 ± 0,64	4,14 ± 0,73	4,78 ± 0,70	2,41 ± 0,91	1,22 ± 0,28	0,50 ± 0,05
TPC(v=2)	1,50 ± 0,06	5,04 ± 0,77	4,30 ± 0,82	5,18 ± 0,90	2,82 ± 0,95	1,49 ± 0,33	0,58 ± 0,08
TPC(v=4)	1,89 ± 0,07	5,02 ± 0,70	4,27 ± 0,93	5,04 ± 0,86	2,63 ± 0,89	1,41 ± 0,26	0,56 ± 0,07
TPC(v=8)	2,26 ± 0,10	4,98 ± 0,69	4,18 ± 0,75	5,01 ± 0,80	2,48 ± 0,90	1,31 ± 0,24	0,54 ± 0,06
TPC($\hat{\nu} = 10,49 \pm 2,62$)	2,06 ± 0,11	4,97 ± 0,70	4,20 ± 0,85	5,05 ± 0,82	2,69 ± 0,83	1,35 ± 0,26	0,55 ± 0,07
P% outliers =10	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
GPC	2,83 ± 0,24	5,13 ± 0,68	4,88 ± 0,72	4,96 ± 0,76	2,58 ± 1,20	1,03 ± 0,58	0,42 ± 0,03
TPC(v=2)	1,49 ± 0,06	5,35 ± 0,80	5,11 ± 0,86	5,25 ± 0,79	2,50 ± 1,16	1,05 ± 0,29	0,48 ± 0,04
TPC(v=4)	1,88 ± 0,08	5,22 ± 0,82	5,06 ± 0,70	5,11 ± 0,84	2,48 ± 1,05	1,04 ± 0,38	0,46 ± 0,04
TPC(v=8)	2,24 ± 0,13	5,17 ± 0,80	4,98 ± 0,72	4,98 ± 0,77	2,51 ± 1,06	1,00 ± 0,33	0,45 ± 0,04
TPC($\hat{\nu} = 11,23 \pm 3,43$)	2,03 ± 0,11	5,29 ± 0,81	5,01 ± 0,75	5,07 ± 0,76	2,52 ± 1,09	1,05 ± 0,43	0,45 ± 0,03
P% outliers =20	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
GPC	2,53 ± 0,17	5,14 ± 0,61	4,51 ± 0,90	5,00 ± 0,62	4,72 ± 0,84	2,76 ± 1,57	0,32 ± 0,02
TPC(v=2)	1,47 ± 0,04	5,33 ± 0,83	4,76 ± 1,12	5,13 ± 0,71	4,50 ± 1,11	2,13 ± 1,61	0,35 ± 0,03
TPC(v=4)	1,79 ± 0,08	5,15 ± 0,74	4,71 ± 0,93	5,11 ± 0,72	4,54 ± 1,14	2,20 ± 1,72	0,35 ± 0,03
TPC(v=8)	2,08 ± 0,10	5,19 ± 0,69	4,59 ± 1,01	5,03 ± 0,78	4,79 ± 0,98	2,48 ± 1,49	0,34 ± 0,03
TPC($\hat{\nu} = 14,13 \pm 2,95$)	2,04 ± 0,10	5,07 ± 0,83	4,62 ± 0,84	5,00 ± 0,68	4,74 ± 0,90	2,37 ± 1,48	0,34 ± 0,03

Fonte: Autoria própria.

Na Tabela 3, temos as estimativas dos hiperparâmetros da função de covariância

para os dados de Coluna Vertebral. As estimativas relativas aos hiperparâmetros dos dados WDBC constam nas Tabelas 4 e 5. Pela Tabela 3, percebe-se que não há diferenças expressivas entre os hiperparâmetros comprimento-escala dos classificadores GPC e TPC, considerando todos os casos de graus de liberdade especificados. Para o hiperparâmetro de sinal de variância α, percebe-se que as estimativas para o classificador TPC com graus de liberdade $\nu = 2$ e $\nu = 4$ apresentam valores menores em todos os casos de *P%* de *outliers*. Nas Tabelas 4 e 5, analisando as estimativas dos hiperparâmetros da função de covariância para os dados WDBC, constatamos que tanto as estimativas dos hiperparâmetros comprimento-escala ρ_d e as estimativas do hiperparâmetro α para os classificadores GPC e TPC não apresentam diferenças expressivas em seus valores que possam justificar alguma tendência.

Tabela 4 – Estimação dos hiperparâmetros para os classificadores GPC e TPC para os dados WDBC - *P% outliers* = 0 e 5.

P% outliers =0											
Modelo	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}
GPC	0,80 ± 0,14	1,02 ± 0,07	1,01 ± 0,08	1,01 ± 0,03	1,03 ± 0,05	1,03 ± 0,07	1,00 ± 0,07	1,01 ± 0,09	0,97 ± 0,06	1,01 ± 0,06	0,98 ± 0,06
TPC(v=2)	0,77 ± 0,08	1,02 ± 0,05	1,01 ± 0,06	0,98 ± 0,06	1,02 ± 0,07	0,98 ± 0,06	0,97 ± 0,08	1,00 ± 0,04	1,03 ± 0,06	1,00 ± 0,06	1,01 ± 0,07
TPC(v=4)	0,70 ± 0,11	1,01 ± 0,09	1,02 ± 0,05	1,06 ± 0,06	0,98 ± 0,06	0,99 ± 0,06	1,01 ± 0,06	1,02 ± 0,09	0,98 ± 0,06	0,99 ± 0,05	0,99 ± 0,06
TPC(v=8)	0,71 ± 0,10	0,98 ± 0,08	0,96 ± 0,08	1,03 ± 0,05	1,04 ± 0,09	0,99 ± 0,08	0,98 ± 0,06	1,02 ± 0,07	1,03 ± 0,10	1,02 ± 0,07	1,02 ± 0,05
TPC($\hat{\nu} = 18,27 \pm 5,37$)	0,89 ± 0,28	1,26 ± 0,85	1,33 ± 1,02	1,27 ± 0,82	1,23 ± 0,83	1,44 ± 1,22	1,41 ± 1,02	1,17 ± 0,60	1,21 ± 0,73	1,42 ± 1,37	1,34 ± 1,08
Modelo		ρ_{11}	ρ_{12}	ρ_{13}	ρ_{14}	ρ_{15}	ρ_{16}	ρ_{17}	ρ_{18}	ρ_{19}	ρ_{20}
GPC	–‖–	1,00 ± 0,05	1,05 ± 0,07	0,99 ± 0,05	1,00 ± 0,06	1,04 ± 0,06	1,00 ± 0,06	1,00 ± 0,04	1,01 ± 0,04	1,03 ± 0,06	1,00 ± 0,07
PC(v=2)	–‖–	1,04 ± 0,09	1,02 ± 0,10	1,05 ± 0,08	0,98 ± 0,06	0,99 ± 0,09	0,99 ± 0,10	1,00 ± 0,07	1,01 ± 0,04	1,00 ± 0,07	1,03 ± 0,08
TPC(v=4)	–‖–	1,00 ± 0,08	0,97 ± 0,09	1,00 ± 0,04	1,01 ± 0,09	1,02 ± 0,07	0,95 ± 0,06	1,00 ± 0,06	0,97 ± 0,05	0,98 ± 0,11	1,00 ± 0,10
TPC(v=8)	–‖–	1,00 ± 0,06	0,96 ± 0,08	0,97 ± 0,03	1,01 ± 0,06	0,98 ± 0,06	0,97 ± 0,05	1,01 ± 0,06	1,01 ± 0,05	0,99 ± 0,07	0,98 ± 0,07
TPC($\hat{\nu} = 18,27 \pm 5,37$)	–‖–	1,28 ± 0,91	1,46 ± 1,36	1,26 ± 0,86	1,20 ± 0,68	1,42 ± 1,33	1,35 ± 1,04	1,32 ± 0,98	1,36 ± 1,19	1,42 ± 1,32	1,28 ± 0,89
Modelo		ρ_{21}	ρ_{22}	ρ_{23}	ρ_{24}	ρ_{25}	ρ_{26}	ρ_{27}	ρ_{28}	ρ_{29}	ρ_{30}
GPC	–‖–	1,00 ± 0,07	0,98 ± 0,05	0,95 ± 0,07	0,99 ± 0,09	0,99 ± 0,08	0,99 ± 0,08	0,99 ± 0,06	0,98 ± 0,11	1,02 ± 0,05	0,99 ± 0,07
TPC(v=2)	–‖–	0,97 ± 0,09	1,02 ± 0,12	0,98 ± 0,09	1,03 ± 0,07	0,99 ± 0,08	0,99 ± 0,07	1,02 ± 0,07	0,98 ± 0,05	1,00 ± 0,08	1,02 ± 0,07
TPC(v=4)	–‖–	0,98 ± 0,08	0,98 ± 0,05	0,97 ± 0,06	1,05 ± 0,08	1,02 ± 0,09	1,00 ± 0,08	1,02 ± 0,07	1,01 ± 0,08	1,00 ± 0,08	1,01 ± 0,07
TPC(v=8)	–‖–	1,02 ± 0,08	1,00 ± 0,10	1,05 ± 0,06	1,01 ± 0,06	1,03 ± 0,07	0,98 ± 0,07	1,00 ± 0,05	1,01 ± 0,06	1,00 ± 0,07	1,00 ± 0,05
TPC($\hat{\nu} = 18,27 \pm 5,37$)	–‖–	1,21 ± 0,60	1,32 ± 0,93	1,13 ± 0,55	1,19 ± 0,56	1,28 ± 0,93	1,28 ± 1,07	1,28 ± 0,86	1,24 ± 0,69	1,47 ± 1,33	1,31 ± 0,92
P% outliers =5											
Modelo	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}
GPC	3,25 ± 0,08	4,20 ± 0,71	4,36 ± 1,05	3,83 ± 0,72	4,20 ± 0,84	2,89 ± 0,59	3,56 ± 0,63	2,71 ± 0,39	2,90 ± 0,87	3,63 ± 0,73	2,84 ± 0,43
TPC(v=2)	1,57 ± 0,03	4,67 ± 0,86	4,72 ± 1,09	3,93 ± 0,80	4,32 ± 0,82	2,89 ± 0,86	3,80 ± 0,55	2,67 ± 0,45	2,58 ± 0,99	3,75 ± 0,99	3,02 ± 0,55
TPC(v=4)	2,03 ± 0,03	4,36 ± 0,49	4,42 ± 1,08	3,89 ± 0,73	4,28 ± 0,87	2,96 ± 0,71	3,62 ± 0,55	2,71 ± 0,65	2,66 ± 0,86	3,58 ± 0,78	2,81 ± 0,31
TPC(v=8)	2,49 ± 0,03	4,39 ± 0,71	4,48 ± 1,09	3,88 ± 0,68	4,39 ± 0,80	2,96 ± 0,73	3,62 ± 0,57	2,74 ± 0,62	2,46 ± 1,04	3,65 ± 0,95	2,79 ± 0,58
TPC($\hat{\nu} = 2,77 \pm 1,01$)	1,49 ± 0,12	4,50 ± 0,71	4,51 ± 1,25	4,08 ± 0,68	4,34 ± 0,95	2,90 ± 0,73	3,80 ± 0,70	2,42 ± 0,67	2,85 ± 1,00	3,84 ± 1,30	2,94 ± 0,65
Modelo		ρ_{11}	ρ_{12}	ρ_{13}	ρ_{14}	ρ_{15}	ρ_{16}	ρ_{17}	ρ_{18}	ρ_{19}	ρ_{20}
GPC	–‖–	3,38 ± 0,78	4,26 ± 0,86	3,74 ± 0,47	3,76 ± 0,96	3,26 ± 1,39	2,66 ± 0,86	3,11 ± 0,72	2,95 ± 0,41	3,30 ± 0,90	3,00 ± 0,52
TPC(v=2)	–‖–	3,49 ± 0,87	4,42 ± 0,87	3,75 ± 0,47	3,69 ± 0,98	3,35 ± 1,40	2,80 ± 0,86	3,49 ± 0,76	2,94 ± 0,46	3,35 ± 0,82	3,02 ± 0,64
TPC(v=4)	–‖–	3,52 ± 0,92	4,42 ± 0,71	3,70 ± 0,32	3,82 ± 0,78	3,40 ± 1,56	2,74 ± 0,82	3,23 ± 0,73	2,72 ± 0,39	3,23 ± 0,98	2,99 ± 0,75
TPC(v=8)	–‖–	3,44 ± 0,89	4,56 ± 0,84	3,68 ± 0,40	3,79 ± 0,85	3,05 ± 1,11	2,69 ± 0,77	3,21 ± 0,91	2,79 ± 0,52	3,22 ± 0,77	2,93 ± 0,67
TPC($\hat{\nu} = 2,77 \pm 0,1$)	–‖–	3,63 ± 0,96	4,46 ± 0,76	3,92 ± 0,52	3,82 ± 0,95	3,21 ± 1,55	2,80 ± 1,12	3,35 ± 0,73	2,97 ± 0,67	3,31 ± 0,94	3,22 ± 0,97
Modelo		ρ_{21}	ρ_{22}	ρ_{23}	ρ_{24}	ρ_{25}	ρ_{26}	ρ_{27}	ρ_{28}	ρ_{29}	ρ_{30}
GPC	–‖–	2,69 ± 0,78	4,26 ± 1,32	2,60 ± 0,83	3,02 ± 1,04	2,57 ± 0,68	3,64 ± 0,86	3,38 ± 0,81	2,25 ± 1,02	3,59 ± 0,71	3,29 ± 0,64
TPC(v=2)	–‖–	2,85 ± 0,90	4,26 ± 1,92	2,62 ± 0,85	3,01 ± 1,10	2,61 ± 0,70	3,64 ± 0,77	3,35 ± 0,61	2,55 ± 0,92	3,68 ± 0,75	3,54 ± 0,57
TPC(v=4)	–‖–	2,59 ± 0,58	4,26 ± 1,48	2,75 ± 0,64	2,99 ± 1,19	2,71 ± 0,84	3,60 ± 0,80	3,34 ± 0,77	2,38 ± 0,95	3,74 ± 0,78	3,45 ± 0,51
TPC(v=8)	–‖–	2,63 ± 0,70	4,23 ± 1,44	2,72 ± 0,76	3,16 ± 1,23	2,59 ± 0,72	3,64 ± 0,93	3,39 ± 0,83	2,53 ± 0,94	3,77 ± 0,64	3,46 ± 0,55
TPC($\hat{\nu} = 2,77 \pm 0,1$)	–‖–	2,49 ± 0,75	4,45 ± 1,39	2,63 ± 0,84	3,12 ± 1,25	2,77 ± 0,71	4,09 ± 1,08	3,61 ± 0,88	2,28 ± 1,41	4,02 ± 0,74	3,40 ± 0,79

Fonte: Autoria própria.

4.2 Aplicação 2: classificação multiclasse.

Para a aplicação multiclasse estabelecemos apenas 5 replicações. Para os experimentos, temos a seguinte configuração: 220 iterações, com o *burn-in* de 100 iterações a cada replicação. Procedemos assim, diminuindo o número de replicações, como forma de amenizar um pouco do custo computacional exigido, que para o caso de classificação multiclasse, aumenta com o número de classes no banco de dados em estudo. Para os dados de CV, inicialmente foram utilizados 80% dos dados para treino e 20% para teste.

Tabela 5 – Estimação dos hiperparâmetros para os classificadores GPC e TPC para os dados WDBC -*P% outliers* = 10 e 20.

P% outliers =10											
Modelo	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}
GPC	3,14 ± 0,49	5,57 ± 1,08	5,54 ± 1,11	5,68 ± 1,15	5,67 ± 1,06	5,91 ± 1,16	5,87 ± 1,21	5,65 ± 1,74	4,66 ± 2,20	5,94 ± 1,20	5,72 ± 1,02
TPC(v=2)	1,54 ± 0,07	5,87 ± 1,34	6,09 ± 1,30	6,11 ± 1,13	6,22 ± 0,72	6,26 ± 1,25	6,36 ± 0,74	5,35 ± 2,47	5,19 ± 1,19	6,22 ± 0,62	5,83 ± 1,05
TPC(v=4)	2,02 ± 0,11	6,21 ± 0,62	5,87 ± 1,08	6,29 ± 0,71	5,81 ± 1,47	6,09 ± 2,00	6,14 ± 1,43	6,03 ± 1,30	3,78 ± 2,82	6,28 ± 0,61	6,17 ± 0,47
TPC(v=8)	2,43 ± 0,18	5,98 ± 1,12	6,25 ± 0,45	5,92 ± 0,86	6,02 ± 0,75	6,36 ± 0,67	6,18 ± 0,84	5,21 ± 1,83	5,27 ± 1,44	5,95 ± 1,19	5,50 ± 1,68
TPC($\hat{\nu} = 4,54 \pm 3,85$)	1,73 ± 0,46	5,95 ± 1,07	6,36 ± 1,26	5,90 ± 0,83	5,87 ± 0,94	6,19 ± 1,00	5,62 ± 1,98	5,93 ± 1,26	4,52 ± 2,73	5,63 ± 1,66	5,83 ± 0,95
Modelo		ρ_{11}	ρ_{12}	ρ_{13}	ρ_{14}	ρ_{15}	ρ_{16}	ρ_{17}	ρ_{18}	ρ_{19}	ρ_{20}
GPC	–‖–	6,10 ± 1,28	5,65 ± 1,25	5,76 ± 1,18	5,15 ± 1,72	4,95 ± 2,09	5,58 ± 1,00	5,46 ± 1,38	5,53 ± 1,32	5,72 ± 1,17	5,47 ± 1,16
TPC(v=2)	–‖–	5,69 ± 1,61	6,24 ± 1,16	6,33 ± 1,04	6,21 ± 0,53	5,86 ± 1,30	5,84 ± 1,37	6,04 ± 0,71	6,20 ± 1,14	5,74 ± 1,21	5,91 ± 1,05
TPC(v=4)	–‖–	5,88 ± 1,64	6,04 ± 1,39	6,23 ± 1,23	5,46 ± 2,21	5,66 ± 1,54	5,77 ± 1,45	5,70 ± 1,62	6,19 ± 0,80	5,74 ± 1,79	5,38 ± 1,59
TPC(v=8)	–‖–	5,95 ± 1,13	6,15 ± 0,88	6,05 ± 1,08	5,96 ± 0,86	5,88 ± 1,07	5,86 ± 1,23	6,09 ± 0,92	6,11 ± 0,75	5,92 ± 0,95	5,81 ± 1,07
TPC($\hat{\nu} = 4,54 \pm 3,85$)	–‖–	5,27 ± 1,80	6,27 ± 0,68	5,92 ± 1,32	5,92 ± 1,54	5,24 ± 2,07	5,27 ± 1,95	6,15 ± 0,91	5,92 ± 1,82	5,57 ± 0,96	5,24 ± 1,96
Modelo		ρ_{21}	ρ_{22}	ρ_{23}	ρ_{24}	ρ_{25}	ρ_{26}	ρ_{27}	ρ_{28}	ρ_{29}	ρ_{30}
GPC	–‖–	5,14 ± 2,26	5,12 ± 1,48	3,44 ± 2,58	4,48 ± 2,38	5,51 ± 1,41	5,59 ± 2,49	5,33 ± 1,59	3,22 ± 3,22	5,10 ± 1,98	5,85 ± 0,89
TPC(v=2)	–‖–	4,48 ± 2,52	5,50 ± 1,36	5,04 ± 2,28	3,98 ± 2,59	5,97 ± 1,68	6,23 ± 1,41	5,98 ± 1,62	2,02 ± 2,35	5,71 ± 1,76	6,04 ± 1,14
TPC(v=4)	–‖–	5,35 ± 2,00	5,03 ± 2,15	3,28 ± 2,85	4,72 ± 2,69	5,75 ± 1,44	5,65 ± 1,72	6,11 ± 1,12	3,01 ± 3,21	5,95 ± 1,23	6,35 ± 0,91
TPC(v=8)	–‖–	3,84 ± 2,32	5,66 ± 1,23	3,81 ± 2,26	4,69 ± 1,85	6,00 ± 0,83	5,86 ± 1,09	5,16 ± 2,23	3,71 ± 2,71	5,37 ± 2,43	5,92 ± 0,88
TPC($\hat{\nu} = 4,54 \pm 3,85$)	–‖–	4,52 ± 2,10	5,15 ± 1,71	3,05 ± 2,45	3,82 ± 2,39	5,19 ± 2,02	5,59 ± 2,14	5,43 ± 2,08	3,31 ± 2,53	5,72 ± 1,59	5,93 ± 1,54
P% outliers =20											
Modelo	α	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}
GPC	0,76 ± 0,10	1,00 ± 0,08	1,00 ± 0,09	0,98 ± 0,06	0,97 ± 0,08	1,01 ± 0,05	0,99 ± 0,05	1,01 ± 0,06	1,03 ± 0,09	1,00 ± 0,07	0,98 ± 0,07
TPC(v=2)	0,77 ± 0,11	0,95 ± 0,07	1,00 ± 0,06	0,97 ± 0,05	0,98 ± 0,07	1,03 ± 0,15	0,95 ± 0,08	0,96 ± 0,07	0,98 ± 0,06	1,02 ± 0,06	1,01 ± 0,06
TPC(v=4)	0,76 ± 0,13	1,04 ± 0,05	1,00 ± 0,07	1,01 ± 0,05	0,96 ± 0,06	1,02 ± 0,06	1,02 ± 0,07	0,99 ± 0,04	1,00 ± 0,06	1,00 ± 0,10	1,01 ± 0,04
TPC(v=8)	0,79 ± 0,14	1,00 ± 0,05	1,00 ± 0,05	1,00 ± 0,08	1,01 ± 0,08	0,99 ± 0,07	1,01 ± 0,05	1,06 ± 0,05	0,98 ± 0,06	0,98 ± 0,10	1,00 ± 0,08
TPC($\hat{\nu} = 19,84 \pm 1,01$)	0,70 ± 0,06	0,99 ± 0,05	1,01 ± 0,07	1,02 ± 0,04	0,98 ± 0,03	1,02 ± 0,10	1,02 ± 0,09	0,99 ± 0,08	0,96 ± 0,04	1,00 ± 0,09	1,02 ± 0,07
Modelo		ρ_{11}	ρ_{12}	ρ_{13}	ρ_{14}	ρ_{15}	ρ_{16}	ρ_{17}	ρ_{18}	ρ_{19}	ρ_{20}
GPC	–‖–	1,00 ± 0,04	1,00 ± 0,06	1,01 ± 0,06	0,99 ± 0,06	0,99 ± 0,06	0,96 ± 0,07	1,04 ± 0,07	0,96 ± 0,10	0,97 ± 0,10	1,00 ± 0,08
TPC(v=2)	–‖–	0,98 ± 0,05	1,01 ± 0,09	1,02 ± 0,07	1,01 ± 0,08	1,00 ± 0,08	0,99 ± 0,07	1,03 ± 0,07	1,04 ± 0,08	0,94 ± 0,06	0,99 ± 0,07
TPC(v=4)	–‖–	1,03 ± 0,07	0,99 ± 0,07	1,01 ± 0,08	1,02 ± 0,04	1,01 ± 0,06	1,03 ± 0,06	0,96 ± 0,06	1,05 ± 0,07	1,01 ± 0,09	0,97 ± 0,05
TPC(v=8)	–‖–	0,97 ± 0,05	1,01 ± 0,06	0,97 ± 0,05	1,00 ± 0,05	1,01 ± 0,10	1,00 ± 0,06	1,02 ± 0,08	1,01 ± 0,03	0,99 ± 0,06	0,97 ± 0,07
TPC($\hat{\nu} = 19,84 \pm 1,01$)	–‖–	1,02 ± 0,06	0,98 ± 0,07	0,97 ± 0,08	1,02 ± 0,04	0,99 ± 0,04	1,00 ± 0,07	0,99 ± 0,06	0,99 ± 0,06	1,04 ± 0,08	1,01 ± 0,07
Modelo		ρ_{21}	ρ_{22}	ρ_{23}	ρ_{24}	ρ_{25}	ρ_{26}	ρ_{27}	ρ_{28}	ρ_{29}	ρ_{30}
GPC	–‖–	1,01 ± 0,07	0,99 ± 0,09	1,00 ± 0,03	0,98 ± 0,05	0,97 ± 0,06	1,01 ± 0,03	1,02 ± 0,07	0,97 ± 0,07	1,03 ± 0,06	0,97 ± 0,06
TPC(v=2)	–‖–	1,00 ± 0,06	1,06 ± 0,04	0,99 ± 0,03	0,97 ± 0,09	0,98 ± 0,06	0,99 ± 0,07	1,01 ± 0,10	0,98 ± 0,04	0,99 ± 0,06	0,97 ± 0,10
TPC(v=4)	–‖–	0,99 ± 0,07	1,02 ± 0,06	1,01 ± 0,04	0,97 ± 0,11	1,00 ± 0,05	1,02 ± 0,08	1,03 ± 0,06	0,96 ± 0,09	0,98 ± 0,10	1,02 ± 0,07
TPC(v=8)	–‖–	1,00 ± 0,07	1,01 ± 0,06	1,02 ± 0,06	0,97 ± 0,05	1,00 ± 0,10	1,00 ± 0,09	1,00 ± 0,05	0,97 ± 0,08	1,00 ± 0,05	1,05 ± 0,07
TPC($\hat{\nu} = 19,84 \pm 1,01$)	–‖–	0,96 ± 0,04	0,98 ± 0,05	1,02 ± 0,06	0,99 ± 0,09	1,00 ± 0,08	1,03 ± 0,05	0,99 ± 0,06	1,02 ± 0,06	1,00 ± 0,04	1,01 ± 0,09

Fonte: Autoria própria.

Na Tabela 6 constam os resultados das médias e desvios padrões das taxas de acerto para os dados de Coluna Vertebral versão multiclasse.

Tabela 6 – Desempenho dos classificadores GPC e TPC nos dados de Coluna Vertebral versão multiclasse.

P% outliers	0	5	10	20
	acerto(%)	acerto(%)	acerto(%)	acerto(%)
GPC	**86.12 ± 2.44**	84.19 ± 3.49	81.93 ± 2.65	80.00 ± 3.34
TPC (v=2)	85.16 ± 2.10	**84.19 ± 1.76**	83.22 ± 1.83	80.64 ± 3.78
TPC (v=4)	86.45 ± 3.34	84.83 ± 2.69	83.54 ± 2.39	83.87 ± 5.22
TPC (v=8)	85.80 ± 2.65	83.87 ± 2.55	82.58 ± 3.49	81.29 ± 4.64
TPC ($\hat{\nu}^*$)	85.48 ± 2.28	84.51 ± 2.44	**83.87 ± 1.14***	**83.87 ± 4.41****

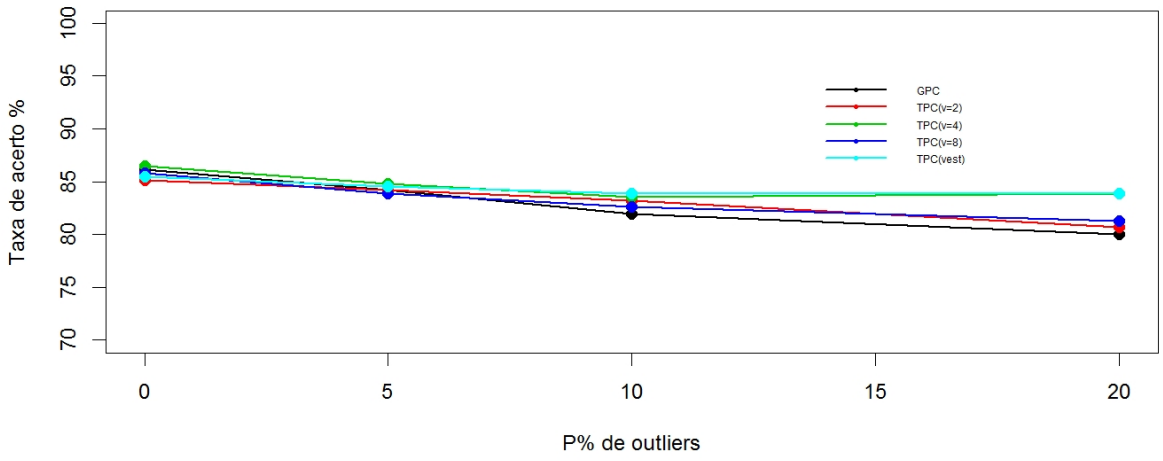
Fonte: Autoria própria. *$\hat{\nu} = 4.42 \pm 0.89$;**$\hat{\nu} = 4.23 \pm 1.52$

Analisando os resultados da Tabela 6, o desempenho do classificador TPC passa a ser melhor em relação ao GPC quando se considera ***P%***=10% e 20%. Os classificadores TPC com melhor desempenho nestes dois casos são, respectivamente, o TPC($\hat{\nu} = 4.42$), e o TPC($\hat{\nu} = 4.23$). A maior diferença entre as médias de taxas de acerto entre os classificadores foi atingida pelo TPC($\hat{\nu} = 4.23$) quando avaliamos o $P\% = 20\%$, com um aumento de aproximadamente 3% quando comparado ao GPC. Pela Tabela, destaca-se ainda que para este banco de dados, o TPC com graus de liberdade ν próximo de 4 parece ser o que oferece a melhor estabilidade da taxa de acerto à medida que aumentamos a quantidade de ruído nos dados. Vale destacar que para a versão multiclasse deste banco

de dados, diferentemente do caso binário, o desempenho do classificador TPC mostrou-se ser significativamente melhor quando comparado ao GPC, à medida que mais *outliers* são inseridos nos dados. No caso binário dos dados de CV, tínhamos apontado indícios da robustez do TPC, ao constatar que suas médias de taxa de acerto eram maiores para $P\% = 5\%, 10\%$ e 20%, mesmo não sendo tão expressivas. No caso multiclasse, entretanto, estes indícios mostraram-se mais evidentes.

Para os dados em questão, à semelhança do caso binário, quando aumentamos a quantidade de ruído nos dados, o desempenho dos classificadores também diminui como consequência do grande montante de *outliers*. Apesar disso, o TPC ainda mantém o melhor desempenho de média de taxa de acerto, atendendo à proposta de ser robusto na presença de dados contaminados por *outliers*. Na Figura 16 constam as médias das taxas de acerto dos classificadores em função da quantidade $P\%$ de *outliers* nos dados.

Figura 16 – Média das taxas de acerto em função da quantidade $P\%$ de *outliers* para os dados de Coluna Vertebral multiclasse.



Fonte: Autoria própria.

Na Tabela 7 constam as estimativas dos hiperparâmetros para os classificadores GPC e TPC para os dados de Coluna Vertebral versão multiclasse. Para o caso multiclasse, o comportamento das estimativas dos hiperparâmetros entre os classificadores parece não apresentar um padrão nítido de tendência quando as avaliamos entre as quantidades $P\%$ de *outliers*. Não descarta-se, contudo, o que pode vir a ser um dos pontos para extensões desta pesquisa, que através de uma análise mais profunda de estudo destes valores de hiperparâmetros, possa-se descobrir padrões de comportamento que possibilitem *insight's*

mais precisos. Agora, quando avaliamos os valores dos hiperparâmetros dentro das classes, percebe-se que o sinal de variância α é maior para a terceira classe em todos os casos de quantidade de *P%* de *outliers*. Os hiperparâmetros comprimento-escala em todos os casos *P%* de *outliers* também são maiores para a terceira classe, especificamente para as primeiras 5 variáveis.

Tabela 7 – Estimação dos hiperparâmetros para os classificadores GPC e TPC para os dados de Coluna Vertebral versão multiclasse.*

P% outliers =0											
Modelo	α_1	α_2	α_3	$\rho_{1,1}$	$\rho_{1,2}$	$\rho_{1,3}$	$\rho_{1,4}$	$\rho_{1,5}$	$\rho_{1,6}$	$\rho_{2,1}$	$\rho_{2,2}$
GPC	1,34 ± 0,60	0,93 ± 0,56	3,17 ± 0,04	2,24 ± 0,82	2,07 ± 0,81	2,53 ± 0,81	1,38 ± 0,44	1,56 ± 0,39	1,81 ± 0,68	1,71 ± 0,84	1,57 ± 0,64
TPC(v=2)	1,04 ± 0,51	0,91 ± 0,51	1,13 ± 0,88	3,31 ± 1,49	3,36 ± 1,55	3,34 ± 1,52	2,28 ± 1,11	3,02 ± 1,89	1,63 ± 0,87	2,80 ± 1,81	2,81 ± 1,74
TPC(v=4)	0,83 ± 0,45	0,58 ± 0,33	2,09 ± 0,05	2,58 ± 1,27	2,36 ± 1,15	2,58 ± 1,31	1,55 ± 0,45	1,58 ± 0,48	1,84 ± 0,83	1,68 ± 0,86	1,55 ± 0,87
TPC(v=8)	0,90 ± 0,80	1,20 ± 0,58	2,04 ± 1,03	2,09 ± 1,52	2,13 ± 1,63	2,25 ± 1,93	1,50 ± 0,73	2,05 ± 1,61	1,33 ± 0,78	2,56 ± 0,85	2,51 ± 1,28
TPC($\hat{\nu}=7,80\pm1,60$)	0,54 ± 0,35	1,00 ± 0,41	2,12 ± 0,22	1,72 ± 0,83	1,64 ± 0,80	1,72 ± 1,04	1,23 ± 0,39	1,35 ± 0,59	1,21 ± 0,70	2,56 ± 1,06	2,39 ± 0,72
Modelo		$\rho_{2,3}$	$\rho_{2,4}$	$\rho_{2,5}$	$\rho_{2,6}$	$\rho_{3,1}$	$\rho_{3,2}$	$\rho_{3,3}$	$\rho_{3,4}$	$\rho_{3,5}$	$\rho_{3,6}$
GPC	-||-	1,72 ± 0,90	1,35 ± 0,35	1,23 ± 0,27	1,29 ± 0,42	4,08 ± 0,34	4,77 ± 0,41	4,50 ± 0,29	4,13 ± 0,41	4,71 ± 0,54	0,74 ± 0,04
TPC(v=2)	-||-	2,86 ± 1,75	2,53 ± 1,88	2,21 ± 1,47	1,15 ± 0,60	2,76 ± 1,80	3,23 ± 2,09	3,12 ± 1,99	3,08 ± 1,99	3,16 ± 2,10	0,94 ± 0,10
TPC(v=4)	-||-	1,73 ± 1,12	1,40 ± 0,42	1,25 ± 0,45	1,32 ± 0,55	3,97 ± 0,29	4,66 ± 0,06	4,69 ± 0,37	4,17 ± 0,57	4,58 ± 0,48	0,92 ± 0,07
TPC(v=8)	-||-	2,68 ± 1,51	2,12 ± 1,22	1,90 ± 0,95	1,46 ± 0,55	3,37 ± 1,42	4,04 ± 1,72	3,69 ± 1,47	3,53 ± 1,37	4,38 ± 1,91	0,92 ± 0,16
TPC($\hat{\nu}=7,80\pm1,60$)	-||-	2,58 ± 1,01	1,93 ± 0,51	1,60 ± 0,37	1,70 ± 0,55	3,86 ± 0,45	4,61 ± 0,44	4,38 ± 0,38	3,96 ± 0,62	5,12 ± 0,64	0,91 ± 0,07
P% outliers =5											
Modelo	α_1	α_2	α_3	$\rho_{1,1}$	$\rho_{1,2}$	$\rho_{1,3}$	$\rho_{1,4}$	$\rho_{1,5}$	$\rho_{1,6}$	$\rho_{2,1}$	$\rho_{2,2}$
GPC	1,44 ± 0,51	0,78 ± 0,36	3,31 ± 0,09	2,53 ± 0,68	2,08 ± 0,64	2,44 ± 0,51	0,89 ± 0,19	0,80 ± 0,22	2,09 ± 0,58	1,55 ± 0,59	1,23 ± 0,09
TPC(v=2)	0,55 ± 0,22	0,50 ± 0,17	1,88 ± 0,06	2,23 ± 0,77	1,88 ± 0,87	2,00 ± 0,88	0,76 ± 0,11	1,00 ± 0,21	1,67 ± 0,55	2,06 ± 1,06	1,26 ± 0,29
TPC(v=4)	0,71 ± 0,40	0,56 ± 0,34	2,16 ± 0,06	2,35 ± 1,19	1,89 ± 0,95	2,22 ± 1,08	0,91 ± 0,23	0,90 ± 0,13	1,80 ± 0,76	1,78 ± 1,07	1,27 ± 0,25
TPC(v=8)	0,62 ± 0,42	0,90 ± 0,29	2,56 ± 0,05	1,74 ± 0,95	1,30 ± 0,56	1,46 ± 0,83	0,93 ± 0,16	0,86 ± 0,23	1,45 ± 0,67	2,23 ± 1,06	1,61 ± 0,61
TPC($\hat{\nu}=6,01\pm1,42$)	0,80 ± 0,34	0,50 ± 0,23	2,21 ± 0,10	2,40 ± 0,88	1,90 ± 0,93	2,42 ± 1,13	1,00 ± 0,24	0,85 ± 0,31	1,78 ± 0,41	1,62 ± 0,97	1,08 ± 0,27
Modelo		$\rho_{2,3}$	$\rho_{2,4}$	$\rho_{2,5}$	$\rho_{2,6}$	$\rho_{3,1}$	$\rho_{3,2}$	$\rho_{3,3}$	$\rho_{3,4}$	$\rho_{3,5}$	$\rho_{3,6}$
GPC	-||-	1,53 ± 0,56	1,06 ± 0,22	0,91 ± 0,19	1,19 ± 0,17	4,84 ± 0,78	5,56 ± 0,66	5,26 ± 0,63	4,98 ± 0,85	5,49 ± 0,36	0,29 ± 0,03
TPC(v=2)	-||-	1,75 ± 0,78	1,16 ± 0,09	0,58 ± 0,26	1,20 ± 0,33	4,82 ± 0,72	5,48 ± 0,71	5,29 ± 0,41	4,67 ± 1,09	5,42 ± 0,53	0,38 ± 0,04
TPC(v=4)	-||-	1,66 ± 0,90	0,99 ± 0,40	0,80 ± 0,26	1,26 ± 0,49	4,71 ± 0,87	5,48 ± 0,86	5,24 ± 0,43	4,80 ± 0,86	5,61 ± 0,25	0,36 ± 0,04
TPC(v=8)	-||-	2,22 ± 1,03	0,88 ± 0,09	0,70 ± 0,27	1,34 ± 0,27	4,57 ± 0,73	5,45 ± 0,44	5,22 ± 0,61	4,50 ± 0,44	5,51 ± 0,43	0,33 ± 0,04
TPC($\hat{\nu}=6,01\pm1,42$)	-||-	1,36 ± 0,50	1,10 ± 0,31	0,88 ± 0,27	1,42 ± 0,33	4,60 ± 0,90	5,31 ± 0,82	5,36 ± 0,47	4,92 ± 0,98	5,29 ± 0,61	0,34 ± 0,03
P% outliers =10											
Modelo	α_1	α_2	α_3	$\rho_{1,1}$	$\rho_{1,2}$	$\rho_{1,3}$	$\rho_{1,4}$	$\rho_{1,5}$	$\rho_{1,6}$	$\rho_{2,1}$	$\rho_{2,2}$
GPC	1,53 ± 0,40	0,78 ± 0,38	3,35 ± 0,10	2,87 ± 0,71	2,36 ± 0,78	2,90 ± 0,65	0,63 ± 0,21	0,78 ± 0,40	2,42 ± 0,92	1,72 ± 0,85	1,11 ± 0,16
TPC(v=2)	0,84 ± 0,41	0,60 ± 0,52	1,52 ± 0,77	3,47 ± 1,51	2,62 ± 1,16	3,74 ± 1,44	0,79 ± 0,14	1,80 ± 2,26	2,05 ± 1,13	2,11 ± 1,57	2,07 ± 1,89
TPC(v=4)	0,72 ± 0,31	0,55 ± 0,21	2,21 ± 0,11	2,44 ± 0,94	2,17 ± 0,84	2,72 ± 0,91	0,85 ± 0,29	0,67 ± 0,21	2,33 ± 1,26	1,95 ± 0,86	1,26 ± 0,60
TPC(v=8)	1,04 ± 0,34	0,45 ± 0,31	2,51 ± 0,05	2,88 ± 0,71	2,42 ± 0,71	3,20 ± 1,01	0,57 ± 0,20	0,60 ± 0,28	2,80 ± 0,99	1,46 ± 1,15	1,05 ± 0,26
TPC($\hat{\nu}=4,42\pm0,89$)	0,93 ± 0,21	0,39 ± 0,13	2,07 ± 0,19	3,43 ± 1,05	2,52 ± 0,59	3,26 ± 0,74	0,61 ± 0,19	0,66 ± 0,27	2,87 ± 0,63	1,49 ± 0,30	1,06 ± 0,18
Modelo		$\rho_{2,3}$	$\rho_{2,4}$	$\rho_{2,5}$	$\rho_{2,6}$	$\rho_{3,1}$	$\rho_{3,2}$	$\rho_{3,3}$	$\rho_{3,4}$	$\rho_{3,5}$	$\rho_{3,6}$
GPC	-||-	1,63 ± 0,75	1,13 ± 0,14	0,76 ± 0,21	1,16 ± 0,17	5,06 ± 0,82	5,62 ± 0,54	5,79 ± 0,57	5,45 ± 0,65	5,23 ± 0,84	0,22 ± 0,01
TPC(v=2)	-||-	2,53 ± 2,23	1,96 ± 2,44	0,84 ± 0,24	0,82 ± 0,34	4,38 ± 2,21	4,86 ± 2,12	4,63 ± 2,15	4,61 ± 2,16	3,86 ± 1,82	0,44 ± 0,42
TPC(v=4)	-||-	1,79 ± 0,90	0,91 ± 0,38	0,68 ± 0,29	1,32 ± 0,57	4,93 ± 1,06	5,58 ± 0,59	5,31 ± 0,55	5,33 ± 0,59	5,17 ± 0,97	0,25 ± 0,02
TPC(v=8)	-||-	1,49 ± 1,00	0,89 ± 0,12	0,98 ± 0,28	1,32 ± 0,44	4,96 ± 1,18	5,91 ± 0,64	5,44 ± 0,62	5,40 ± 0,92	5,12 ± 0,63	0,24 ± 0,01
TPC($\hat{\nu}=4,42\pm0,89$)	-||-	1,21 ± 0,32	1,02 ± 0,55	0,72 ± 0,21	1,12 ± 0,44	4,66 ± 1,23	5,81 ± 0,53	5,72 ± 0,34	5,61 ± 0,73	5,15 ± 0,80	0,25 ± 0,02
P% outliers =20											
Modelo	α_1	α_2	α_3	$\rho_{1,1}$	$\rho_{1,2}$	$\rho_{1,3}$	$\rho_{1,4}$	$\rho_{1,5}$	$\rho_{1,6}$	$\rho_{2,1}$	$\rho_{2,2}$
GPC	1,11 ± 0,30	1,00 ± 0,57	3,21 ± 0,51	2,32 ± 0,50	2,23 ± 0,69	2,39 ± 0,60	0,83 ± 0,17	1,39 ± 0,42	2,05 ± 0,63	1,95 ± 0,84	1,55 ± 0,78
TPC(v=2)	0,48 ± 0,27	0,57 ± 0,16	1,90 ± 0,05	2,26 ± 1,22	2,36 ± 1,26	2,27 ± 1,57	0,66 ± 0,37	0,79 ± 0,52	2,30 ± 1,76	1,44 ± 0,72	1,05 ± 0,63
TPC(v=4)	0,73 ± 0,28	0,45 ± 0,19	2,22 ± 0,05	2,55 ± 0,88	2,72 ± 1,02	2,80 ± 1,23	0,64 ± 0,27	0,54 ± 0,18	3,01 ± 1,43	1,25 ± 0,48	0,95 ± 0,32
TPC(v=8)	1,07 ± 0,70	0,87 ± 0,66	2,14 ± 1,10	2,39 ± 0,76	2,32 ± 0,80	3,35 ± 2,22	1,45 ± 1,52	2,01 ± 2,39	1,75 ± 1,13	2,43 ± 2,18	1,98 ± 1,58
TPC($\hat{\nu}=4,23\pm1,52$)	0,98 ± 0,59	0,70 ± 0,32	1,52 ± 0,86	3,35 ± 1,53	3,75 ± 1,88	3,83 ± 1,77	0,64 ± 0,32	1,22 ± 1,08	2,22 ± 1,82	2,43 ± 1,90	2,11 ± 1,85
Modelo		$\rho_{2,3}$	$\rho_{2,4}$	$\rho_{2,5}$	$\rho_{2,6}$	$\rho_{3,1}$	$\rho_{3,2}$	$\rho_{3,3}$	$\rho_{3,4}$	$\rho_{3,5}$	$\rho_{3,6}$
GPC	-||-	1,52 ± 0,72	1,67 ± 1,26	0,92 ± 0,39	1,36 ± 0,38	4,86 ± 1,07	5,20 ± 0,95	5,37 ± 0,86	5,24 ± 0,74	5,27 ± 0,61	0,24 ± 0,18
TPC(v=2)	-||-	2,20 ± 1,33	1,67 ± 1,14	0,96 ± 0,70	1,97 ± 0,84	4,69 ± 1,06	5,62 ± 1,00	5,62 ± 0,36	5,37 ± 0,57	5,69 ± 0,27	0,19 ± 0,02
TPC(v=4)	-||-	1,30 ± 0,39	1,16 ± 0,35	0,72 ± 0,17	1,40 ± 0,70	5,04 ± 0,73	5,35 ± 0,95	5,64 ± 0,47	5,42 ± 0,70	5,42 ± 0,30	0,19 ± 0,02
TPC(v=8)	-||-	2,43 ± 2,18	2,17 ± 2,34	1,14 ± 0,77	1,08 ± 0,53	3,83 ± 1,82	5,00 ± 2,23	4,58 ± 2,07	4,45 ± 2,12	4,60 ± 1,94	0,40 ± 0,50
TPC($\hat{\nu}=4,23\pm1,52$)	-||-	2,39 ± 1,79	2,43 ± 2,02	1,74 ± 1,79	1,14 ± 0,64	3,73 ± 1,75	4,17 ± 2,25	4,46 ± 2,11	4,06 ± 1,94	4,16 ± 1,91	0,47 ± 0,45

*em $\rho_{i,j}$: **i** refere-se à classe $i=1,2,3$ e **j** refere-se à variável $j=1,2,\ldots,6$

Fonte: Autoria própria.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve por objetivo propor um novo classificador de processo latente que pudesse oferecer robustez em dados cujas variáveis de entrada $\mathbf{x}$ fossem contaminadas por *outliers*, utilizando uma distribuição a priori de Processo t-Student. O ganho que almejávamos alcançar com a utilização desta distribuição a priori, foi avaliado em relação ao classificador de Processo Gaussiano já proposto na literatura.

Diante das aplicações em dados reais, concluímos que o modelo proposto alcançou resultados bem promissores. Quando avaliamos as aplicações na tarefa de classificação binária, percebemos que a robustez do classificador TPC não ficou tão nítida nos resultados, apesar de, para os dados de Coluna Vertebral, as médias de taxa de acerto terem sido sensivelmente maiores para o classificador TPC à medida que aumentávamos a quantidade de ruído nos dados. No caso de classificação multiclasse, entretanto, o classificador TPC apresentou desempenho expressivamente melhor quando comparado ao GPC, mantendo maior média de taxa de acerto à medida que aumentávamos a quantidade de *outliers* nos dados. Podemos também constatar, ao avaliar cuidadosamente os resultados, que não houve nenhum cenário de $P\%$ de *outliers* no qual o TPC foi inferior em desempenho ao GPC, o que nos leva a concluir que o classificador TPC apresenta desempenho tão satisfatório quanto o GPC, mesmo em dados que teoricamente não apresentam *outliers*. Nos caso binário para os dados WDBC, por exemplo, quando avaliamos $P\% = 0\%$, o desempenho do TPC chegou a ser maior que o GPC.

Neste trabalho abordamos técnicas MCMC para fazer inferências sobre os modelos de classificação utilizados. Técnicas MCMC requerem muito esforço computacional, por isso adequamos em cada aplicação, tanto para o problema de classificação binária quanto o caso multiclasse, de acordo com os recursos computacionais disponíveis, o número de pontos de dados para treino, a quantidade de iterações a cada replicação e o número de replicações. Como uma proposta de extensão futura deste trabalho, almejando fazer aplicações em bancos de dados que normalmente utilizam milhares de observações para treino, tais como problemas de Reconhecimento de Face e de Dígitos Manuscritos, uma proposta alternativa para alcançar este objetivo seria utilizar métodos de inferência bayesiana aproximada, como os métodos variacionais, aproximação de Laplace ou o EP.

Os resultados neste trabalho foram bem promissores, e gostaríamos, como veio a ser planejado, de desenvolver um Capítulo de simulação. Devido a todos os

desdobramentos da pesquisa, e pela ocorrência de imprevistos, esta ideia teve de ser adiada para extensões futuras deste trabalho. Por isso, nos limitamos apenas a avaliar o desempenho do classificador proposto nas aplicações em bancos de dados reais.

Na literatura de classificação de padrões, há uma gama de classificadores utilizados em aplicações práticas. Dependendo do problema, alguns destes podem ser mais considerados do que outros. Neste trabalho, afim de sermos justos em nossa comparação de desempenho, utilizamos como modelo *baseline* o classificador GPC, uma vez que o TPC proposto utiliza os mesmos fundamentos de construção deste último. Não nos estendemos a comparar o desempenho do TPC com outros classificadores por questões de tempo e de objetivo traçado, além de que isso implicaria, como consequência, ter de mudar o foco desta pesquisa. Entretanto, uma ideia, para continuidade desta pesquisa, poderia ser justamente avaliar o TPC em relação a classificadores que se propõem serem robustos a *outliers* nos dados de entrada $\mathbf{x}$, ou a depender das aplicações em dados reais que formos utilizar, avaliar o TPC junto a classificadores que são normalmente utilizados nestas aplicações.

Concluindo, podemos resumir as propostas de extensões futuras desta pesquisa nos seguintes itens:

- Utilizar métodos alternativos de inferência bayesiana aproximada para aplicar o classificador TPC em bancos de dados que normalmente utilizam milhares de observações para treino;
- Utilizar bancos de dados para aplicações específicas em problemas existentes na literatura;
- Avaliar o desempenho do TPC junto a outros classificadores que se propõem serem robustos a *outliers* nos dados de entrada $\mathbf{x}$;
- Desenvolver um capítulo de simulação para melhor descrição das propriedades do classificador proposto neste trabalho.

REFERÊNCIAS

ABRAHAMSEN, P. **A review of Gaussian random fields and correlation functions**. 2th. ed. Oslo: Norsk Regnesentral: Norwegian Computing Center, 1997.

ADLER, R. J. **The geometry of random fields**. Philadelphia: SIAM, 2010.

AGGARWAL, C. C. **Managing and mining sensor data**. New York: Springer, 2013.

ANDERSON, J. 7 logistic discrimination. **Handbook of statistics**, Netherlands, v. 2, p. 169-191, 1982.

ANDRADE, J. A. A. On the robustness of the student t process as an alternative to the gaussian process. **Bernoulli Journal**. Submetido à publicação.

BARRERA, J.; TERADA, R.; JR, R. H.; HIRATA, N. S. Automatic programming of morphological machines by pac learning. **Fundamenta Informaticae**, Netherlands, v. 41, n. 1, 2, p. 229-258, 2000.

BARRETO, G. A.; BARROS, A. L. B. A robust extreme learning machine for pattern classification with outliers. **Neurocomputing**, Netherlands, v. 176, p. 3-13, 2016.

BASU, D.; TIWARI, R. A note on the dirichlet processs. *In*: **Selected works of Debabrata Basu**. New York: Springer, 2011. p. 355-369.

BENDRE, S.; BARNETT, V.; LEWIS, T. **Outliers in statistical data**. [*S. l.*]: JSTOR, 1994.

BERGER, B.; RAUSCHER, F. Robust gaussian process modelling for engine calibration. **IFAC Proceedings Volumes**, United Kingdom, v. 45, n. 2, p. 159-164, 2012.

BERGER, J. O. **Statistical decision theory and Bayesian Analysis**. New York: Springer Science & Business Media, 1985.

BERNARDO, J.; BERGER, J.; DAWID, A.; SMITH, A. *et al.* Regression and classification using gaussian process priors. **Bayesian statistics**, Oxford, v. 6, p. 475, 1998.

BETANCOURT, M. A conceptual introduction to Hamiltonian Monte Carlo. **arXiv preprint arXiv:1701.02434**, 2017.

BISHOP, C. M. **Pattern recognition and machine learning**. New York: Springer, 2006.

BLOCH, I. On fuzzy distances and their use in image processing under imprecision. **Pattern Recognition**, United Kingdom, v. 32, n. 11, p. 1873-1895, 1999.

BOOTKRAJANG, J. A generalised label noise model for classification in the presence of annotation errors. **Neurocomputing**, Netherlands, v. 192, p. 61–71, 2016.

CAMPOS, T. E. de. **Técnicas de seleção de características com aplicações em reconhecimento de faces**. 2001. 138 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2001.

COVER, T. M.; THOMAS, J. A. Information theory and statistics. **Elements of Information Theory**, New York, v. 1, n. 1, p. 279–335, 1991.

DAWID, A. P. Some matrix-variate distribution theory: notational considerations and a bayesian application. **Biometrika**, United Kingdom, v. 68, n. 1, p. 265-274, 1981.

DOKSUM, K. Tailfree and neutral random probabilities and their posterior distributions. **The Annals of Probability**, United States, p. 183–201, 1974.

DUA, D.; GRAFF, C. **UCI Machine Learning Repository**. Irvine, CA, 2017. Disponível em: http://archive.ics.uci.edu/ml. Acesso em: 01 Dez. 2020.

DUANE, S.; KENNEDY, A. D.; PENDLETON, B. J.; ROWETH, D. Hybrid Monte Carlo. **Physics letters B**, Netherlands, v. 195, n. 2, p. 216–222, 1987.

DUDA, R. O.; HART, P. E. **Pattern classification and scene analysis**. New York: Wiley, 1973. v. 3.

FERGUSON, T. S. A bayesian analysis of some nonparametric problems. **The Annals of Statistics**, United States, v. 1, n. 2, p. 209-230, 1973.

GAMERMAN, D. **Markov chain Monte Carlo:** stochastic simulation for Bayesian inference. London: Chapman & Hall, 1997.

GAMERMAN, D.; LOPES, H. F. **Markov chain Monte Carlo:** stochastic simulation for Bayesian inference. London: Chapman & Hall/ CRC, 2006.

GELMAN A., C. J. B.; RUBIN, D. B.; STERN, H. S. **Bayesian data analysis.** London: Chapman and Hall, 2004.

GIBBS, M.; MACKAY, D. **Efficient implementation of Gaussian processes.** Cambridge, 1997. Não publicado.

GIBBS, M. N. **Bayesian Gaussian processes for regression and classification**. 1998. 128 f. Tese (Doutorado) - University of Cambridge, Cambridge, 1998.

GROOT, M. H. D.; SCHERVISH, M. J. **Probability and statistics**. Boston: Addison-Wesley, 2002.

HJORT, N. L. *et al.* Nonparametric bayes estimators based on beta processes in models for life history data. **The Annals of Statistics**, United States, v. 18, n. 3, p. 1259-1294, 1990.

HOFERT, M. On sampling from the multivariate t distribution. **The R Journal**, [Vienna], v. 5, n. 2, p. 129–136, 2013.

HOFFMAN, M. D.; GELMAN, A. The no-u-turn sampler: adaptively setting path lengths

in hamiltonian monte carlo. **J. Mach. Learn. Res.**, [Cambridge, Mass.], v. 15, n. 1, p. 1593-1623, 2014.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: a review. **IEEE Transactions on pattern analysis and machine intelligence**, United States, v. 22, n. 1, p. 4–37, 2000.

JYLANKI, P.; VANHATALO, J.; VEHTARI, A. Robust gaussian process regression with a student-t likelihood. **Journal of Machine Learning Research**, [Cambridge, Mass.], v. 12, n. 11, 2011.

KALBFLEISCH, J. D. Non-parametric bayesian analysis of survival time data. **Journal of the Royal Statistical Society:** Series B (Methodological), United Kingdom, v. 40, n. 2, p. 214–221, 1978.

KIBRIA, B. G.; JOARDER, A. H. A short review of multivariate t-distribution. **Journal of Statistical Research**, Bangladesh, v. 40, n. 1, p. 59–72, 2006.

KIM, H.-C.; GHAHRAMANI, Z. Outlier robust gaussian process classification. *In*: **Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)**. [S.l.], 2008. p. 896-905.

KOCIJAN, J. **Modelling and control of dynamic systems using Gaussian process models**. [*S. l.*]: Springer, 2016.

KOLMOGOROV, A. N. Interpolation und extrapolation von stationaren zufalligen folgen. **Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya**, Russian Academy of Sciences, Steklov Mathematical Institute of Russian, Moskow, v. 5, n. 1, p. 3-14, 1941.

KUSS, M. **Gaussian process models for robust regression, classification, and reinforcement learning**. 2006. 189 f. Tese (Doutorado) - Technische Universität Darmstadt, Germany, 2006.

MACKAY, D. J. A practical bayesian framework for backpropagation networks. **Neural computation**, Cambridge, Mass., v. 4, n. 3, p. 448–472, 1992.

MARDIA, K. V. **Multivariate analysis**. London; New York: Academic Press, 1979.

MATTOS, C. L. C. **Recurrent gaussian processes and robust dynamical modeling**. 2017. 189 f. Tese (Doutorado em Engenharia de Teleinformática) - Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2017.

MCLACHLAN, G. J. **Discriminant analysis and statistical pattern recognition**. Hoboken, New Jersey: John Wiley & Sons, 1992.

MINKA, T. P. **A family of algorithms for approximate Bayesian inference**. 2001. 75 f. Tese (Doutorado) - Massachusetts Institute of Technology, 2001.

MINKA, T. P. **A comparison of numerical optimizers for logistic regression**. 2003. 18 f. Não publicado.

MOALA, F. A.; O'HAGAN, A. Elicitation of multivariate prior distributions: a nonparametric bayesian approach. **Journal of Statistical Planning and Inference**, Netherlands, v. 140, n. 7, p. 1635-1655, 2010.

MORAIS, E. C. **Reconhecimento de padrões e redes neurais artificiais em predição de estruturas secundárias de proteínas**. 2010. 135 f. Tese (Doutorado em Engenharia de Sistemas e Computação) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

MORIMOTO, C.; YACOOB, Y.; DAVIS, L. Recognition of head gestures using hidden markov models. *In*: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 13., 1996, Vienna, Austria. **Proceedings** [...] Los Alamitos, California: IEEE, 1996. v. 3, p. 461-465.

MULIERE, P.; WALKER, S. A bayesian non-parametric approach to survival analysis using polya trees. **Scandinavian Journal of Statistics**, United Kingdom, v. 24, n. 3, p. 331-340, 1997.

NAIR, J.; WIERMAN, A.; ZWART, B. The fundamentals of heavy-tails: properties, emergence, and identification. *In*: ACM SIGMETRICS INTERNATIONAL CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEMS, 2013, Pittsburgh. **Proceedings** [...] New York: Association for Computing Machinery, 2013. p. 387-388.

NEAL, R. M. **Bayesian learning for neural networks**. New York: Springer : New York Science & Business Media, 1996. v. 118.

NEAL, R. M. Monte carlo implementation of gaussian process models for bayesian regression and classification. **arXiv preprint physics/9701026**, 1997.

NEAL, R. M. MCMC using Hamiltonian dynamics. *In.*: BROOKS, S. *et al*. (ed.) **Handbook of markov chain Monte Carlo.** London : Chapman & Hall: CRC Press, 2011.

NIU, L.; LI, W.; XU, D. Visual recognition by learning from web data: a weakly supervised domain generalization approach. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2015, Boston. **Proceedings** [...] [*S. l.*: *s. n.*], 2015. v. 1, p. 2774-2783.

NIU, L.; XU, X.; CHEN, L.; DUAN, L.; XU, D. Action and event recognition in videos by learning from heterogeneous web sources. **IEEE transactions on neural networks and learning systems**, United States, v. 28, n. 6, p. 1290-1304, 2016.

NYDICK, S. W. The wishart and inverse wishart distributions. **Electronic Journal of Statistics**, United States, v. 6, p. 1-19, 2012.

O'HAGAN, A. Curve fitting and optimal design for prediction. **Journal of the Royal Statistical Society: Series B (Methodological)**, United Kingdom, v. 40, n. 1, p. 1-24, 1978.

O'HAGAN, A. Bayes–hermite quadrature. **Journal of Statistical Planning and Inference**, Netherlands, v. 29, n. 3, p. 245–260, 1991.

O'HAGAN, A.; KENNEDY, Marc C.; OAKLEY, J. E. Uncertainty analysis and other inference tools for complex computer codes. **Bayesian Statistics**, Oxford, p. 503-524, 1999.

OPPER, M.; WINTHER, O. Gaussian processes for classification: mean-field algorithms. **Neural computation**, United States, v. 12, n. 11, p. 2655–2684, 2000.

PAL SANKAR K E BEZDEK, J. C. **Modelos difusos para reconhecimento de padrões:** métodos que pesquisam estruturas em dados. [*S. l.*: *s. n.*], 1992.

PAO, Y. **Adaptive pattern recognition and neural networks**. Reading, Mass.: Addison-Wesley 1989.

PHADIA, E. G. **Prior processes and their applications**. [*S. l.*]: Springer, 2015.

R CORE TEAM. **R: A language and environment for statistical computing**. Vienna, Austria: R Foundation for Statistical Computing, 2020. Disponível em: https://www.R-project.org/. Acesso em: 01 Dez. 2020.

RIPLEY, B. D. Flexible non-linear approaches to classification. *In*: CHERKASSKY, V.; FRIEDMAN, Jerome H.; WECHSLER, H. (ed.) **From statistics to neural networks**. New York: Springer, 1994. p. 105-126.

RIPLEY, B. D. **Pattern recognition and neural networks**. Cambridge; New York: Cambridge University Press, 1996.

SHAH, A.; WILSON, A.; GHAHRAMANI, Z. Student-t processes as alternatives to gaussian processes. *In*: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, 17, 2014, San Diego, California. **Proceedings** [...] [*S. l.*: *s. n.*], 2014. p. 877-885.

SOLIN, A.; SARKK, A, S. State space methods for efficient inference in student-t process regression. *In*: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, 18, 2015, San Diego, California. **Proceedings** [...] [*S.l.*: *s.n.*], 2015. p. 885–893.

STAN DEVELOPMENT TEAM. **RStan:** the R interface to Stan. R package version 2.21.2. 2020. Disponível em: http://mc-stan.org/. Acesso em: 01 Dez. 2020.

STEIN, M. L. **Interpolation of spatial data**. New York: Springer-Verlag, 1999. (Springer Series in Statistics).

TANG, Q.; NIU, L.; WANG, Y.; DAI, T.; AN, W.; CAI, J.; XIA, S.-T. Student-t process regression with student-t likelihood. *In*: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 26., 2017, Melbourne, Australia. **Proceedings** [...] [*S. l.*: *s. n.*], 2017. p. 2822-2828.

TANG, Q.; WANG, Y.; XIA, S.-T. Student-t process regression with dependent student-t noise. *In*: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, 22., 2016, The Hague, Holland. **Proceedings** [...] Amsterdam: IOS Press, 2016. p. 82–89.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 4th. ed. Burlington: Academic Press, 2009.

VANHATALO, J.; JYLANKI, P.; VEHTARI, A. Gaussian process regression with student-t likelihood. *In*: ANNUAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 23., 2009, Vancouver. **Proceedings** [...]. La Jolla, CA: Neural Information Processing Systems, 2009. Tema: Advances in neural information processing systems. p. 1910-1918.

VITERBI, A. J.; OMURA, J. K. **Principles of digital communication and coding**. New York: McGraw-Hill, 1979.

WEST, M. Outlier models and prior distributions in bayesian linear regression. **Journal of the Royal Statistical Society:** Series B (Methodological), United Kingdom, v. 46, n. 3, p. 431-439, 1984.

WIENER, N. **Extrapolation, interpolation, and smoothing of stationary time series:** with engineering applications. Cambridge: MIT Press, 1949.

WILLIAMS, C. K.; BARBER, D. Bayesian classification with gaussian processes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, United States, v. 20, n. 12, p. 1342-1351, 1998.

WILLIAMS, C. K.; RASMUSSEN, C. E. **Gaussian processes for machine learning**. Cambridge: MIT Press, 2006. v. 2.

APÊNDICE A – STAN: UMA BREVE INTRODUÇÃO

O Stan é uma plataforma/software para modelagem estatística de alta performance, escrita em C++, utilizando uma linguagem de modelagem probabilística. O Stan implementa todo o aparato de inferência estatística bayesiana completa com MCMC (NUTS,HMC), oferecendo também ferramentas para: inferência variacional (*Automatic Differentiation Variational Inference* (ADVI)) e métodos de otimização (*Limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS)).

O Stan pode ser utilizado em outros softwares por meio de interfaces, dentre as quais podemos citar: Rstan, PyStan, MatlabStan. Para utilizarmos a interface do Stan no R, necessitamos instalar um kit de ferramentas chamado Rtools[1] que permite a utilização de um compilador C++ para "traduzir" o código escrito no R para o C++. Após a instalação do Rtools, precisamos somente baixar a biblioteca 'rstan'[2] no R para ter acesso à analise de modelos Stan. Para saber mais detalhes sobre o Stan, pode-se consultar o manual do usuário (STAN DEVELOPMENT TEAM, 2020) e as referências lá inseridas.

Os modelos Stan são organizados em uma sequência de blocos, cujos corpos são constituídos de declarações e atribuições[3] de variáveis. O conjunto de blocos completos são refletidos no seguinte 'esqueleto':

```
functions {
// ... delcaração de funções e definições ...
}
data {
// ... declarações ...
}
transformed data {
// ... declarações ... atribuições ...
}
parameters {
// ... declarações ...
```

[1] Versão recente disponível em:<https://cran.r-project.org/bin/windows/Rtools/>.
[2] Interface do Stan no R: <https://mc-stan.org/users/interfaces/>.
[3] Pode ocorrer de alguns blocos serem somente constituídos de declarações de variáveis, seguido de blocos com as atribuições.

```
}
transformed parameters {
// ... declarações ... atribuições ...
}
model {
// ... declarações ... atribuições ...
}
generated quantities {
// ... declarações ... atribuições ...
}
```

Todos os blocos acima, exceto o bloco **model**, são de preenchimento opcional, e os mesmos devem ser inseridos na ordem como foram apresentados no esqueleto acima. Um pequeno detalhamento de cada bloco é dado a seguir: bloco **functions** contém funções definidas pelo usuário; bloco **data** contém os dados requeridos pelo modelo; bloco **trasnformed data** permite a definição de constantes e transformações que são funções dos dados; bloco **parameters** contém a declaração dos parâmetros do modelo que serão amostrados; bloco **transformed parameters** permite a definição de variáveis que sejam funções dos dados e dos parâmetros; bloco **model** é onde a função log probabilidade do modelo é definida; por fim, o bloco **generated quantities**[4] permite a definição de quantidades baseadas nos parâmetros estimados, ou a geração de números pseudo-aleatórios.

A seguir ilustraremos através de um exemplo, o funcionamento de um programa Stan.

Exemplo: Regressão de Processo Gaussiano

Os dados utilizados para este exemplo são referentes a uma tarefa de aprendizagem da dinâmica inversa de um braço de robô antropomórfico SARCOS de sete graus de liberdade. Este conjunto de dados possui vetores de entradas **x** 21-dimensional, com 48.933 pares de entrada-saída, dos quais 44.484 são para treino, e 4.449 são destinados a tarefa de teste. A variável de saída **y** são sete torques de juntas, mas, seguindo o

[4] Este bloco é executado somente após a amostra dos parâmetros ter sido realizada. Este bloco pode ser utilizado para fazer inferências a posteriori como calculo de esperanças e de estimativas de funções dos parâmetros.

trabalho de Williams e Rasmussen (2006), apresentamos resultados para apenas um dos sete mapeamentos, das 21 variáveis de entrada ao primeiro dos sete torques. A seguir apresentamos um possível código Stan para este problema.

```
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha, vector rho, real sigma){
int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + sigma;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+sigma;
return K;
}
}
data {
  int<lower=1> N1;
  int<lower=1> D;
  vector[D] x1[N1];
  vector[N1] y1;
  int<lower=1> N2;
  vector[D] x2[N2];
}
transformed data {
  int<lower=1> N = N1 + N2;
  vector[N] mu;
```

```
  vector[D] x[N];
  for (n1 in 1:N1) x[n1] = x1[n1];
  for (n2 in 1:N2) x[N1 + n2] = x2[n2];

  for(i in 1:N) mu[i]<-0; // vetor de médias nulo
}
parameters {
  vector<lower=0>[D] rho;
  real<lower=0> alpha;
  real sigma;
  vector[N] eta;
}
transformed parameters {
vector[N] f;
{
matrix[N,N] L;
matrix[N,N] L_K;

L=L_cov_exp_quad_ARD(x,alpha,rho,sigma);
L_K = cholesky_decompose(L);
f=mu+(L_K*eta); // amostrando de uma Gaussiana multivariada
}
}
model {
  rho ~ inv_gamma(5, 5);
  alpha ~ normal(0, 1);
  sigma ~ normal(0, 1);
  eta ~ normal(0, 1);

  y1~ normal(f[1:N1], sigma);

}
```

```
generated quantities {
  vector[N2] y2;
  for (n2 in 1:N2)
    y2[n2] = normal_rng(f[N1 + n2],sigma);
}"
```

Dentro do bloco **functions** temos a função L_cov_exp_quad_ARD que implementa a função de covariância como dada em (4.1). Cada linha de código no programa Stan acima é encerrada com um (;). Toda forma de declaração dentro dos blocos em Stan é precedida por um termo especificando o tipo de dado/variável declarada, que nesse caso podem ser: *int*, *real*, *vector*, *row_vector* e *matrix*. O Stan também permite especificar os limites (superior e inferior) das variáveis declaradas, usando a sintaxe <lower,upper>, por exemplo: int<lower=1> N, real<lower=0> sigma. ,vector<lower=1,upper=3> rho[N] etc.

Dentro do bloco **data** temos a declaração de dados constantes: N1 e D que são respectivamente a quantidade de pontos de dados para treino e dimensão do vetor de entradas $\mathbf{x}$. O restante das declarações são: x1[N1], y1 e X2[N2]. Estas variáveis referem-se respectivamente ao vetor de entradas $\mathbf{x} \in \mathbb{R}^D$ para treino, ao vetor de saídas de treino, e ao vetor de entradas $\mathbf{x}_* \in \mathbb{R}^D$ de testes.

Dentro do bloco **transformed data** constam respectivamente a definição de um termo constante N, quantidade total de dados (N1+N2), o vetor de médias nulo $\boldsymbol{\mu}$ e a declaração de uma variável x[N], que será a concatenação dos dados de treino e teste. Dentro desse bloco há um comentário após a atribuição de zeros ao vetor $\boldsymbol{\mu}$, após duas barras invertidas (//). Outra forma de comentar em códigos Stan seria utilizando (#).

Dentro do bloco **parameter** temos as variáveis declaradas rho, alpha, sigma e eta, que são respectivamente o parâmetro comprimento-escala ρ_d, o sinal de variância α, a variância do termo ruidoso do modelo $\varepsilon \sim N(0, \sigma^2)$ e o vetor de variáveis iid com distribuição Normal de média zero e variância 1 correspondente ao vetor $\mathbf{z}$ na amostragem de $\mathbf{f} = \boldsymbol{\mu} + \mathbf{Lz}$.

No bloco **transformed parameters**, constam a definição do vetor latente $\mathbf{f}$ e a definição da matriz de covariância $\mathbf{K}$. Ainda nesse bloco, temos a respectiva amostragem de $\mathbf{f}$ pela fórmula $\mathbf{f} = \boldsymbol{\mu} + \mathbf{Lz}$. Dentro do bloco **model**, temos a definição das distribuições a priori aos respectivos hiperparâmetros definidos no bloco **parameters**, e em seguida, temos a definição da função de log probabilidade do modelo $\mathbf{y} = f(\mathbf{x}) + \varepsilon$. Por útlimo, no

bloco **generated quantities** temos a declaração da variável y2 para receber os valores preditos das entradas de teste.

Agora, a fins de exemplificação, considere uma amostra de valores $\mathbf{x} \in \mathbb{R}^{21}$ de N1=100, para treino, e $\mathbf{x}_* \in \mathbb{R}^{21}$ de N2=5 para teste. Utilizando o modelo Stan acima, basta inserirmos as seguintes linhas de código no R para obtermos os resultados

```
library('rstan')

dadostan<-list(N1=N1,N2=N2,sigma=0.2,D=D,x1=X_trn,x2=X_tst,y1=Y_trn)

ajuste<-stan(model_code=model_GP,data=dadostan,iter=200,warmup=100,chains=1)
 print(ajuste,c('rho','alpha','sigma','y2'))
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
rho[1]	1.13	0.19	0.78	0.47	0.68	0.89	1.26	3.32	16	1.17
rho[2]	1.35	0.12	0.81	0.43	0.85	1.07	1.52	3.61	45	1.06
rho[3]	1.16	0.09	0.56	0.52	0.82	1.04	1.38	2.65	42	1.08
rho[4]	1.24	0.17	1.12	0.32	0.68	0.84	1.33	4.83	44	1.11
rho[5]	1.31	0.09	0.55	0.58	0.91	1.19	1.61	2.11	37	1.14
rho[6]	1.07	0.08	0.51	0.49	0.73	0.99	1.18	2.36	37	1.13
rho[7]	1.24	0.09	0.45	0.52	0.91	1.19	1.51	2.09	23	1.05
rho[8]	1.06	0.04	0.41	0.59	0.82	0.97	1.18	2.18	105	1.01
rho[9]	1.15	0.10	0.64	0.62	0.80	0.96	1.16	3.32	42	1.00
rho[10]	1.05	0.07	0.51	0.51	0.70	0.87	1.27	2.46	59	1.06
rho[11]	1.25	0.15	0.71	0.44	0.66	1.08	1.60	2.98	23	1.06
rho[12]	1.67	0.37	0.86	0.56	1.01	1.31	2.32	3.10	5	1.26
rho[13]	1.29	0.09	0.66	0.51	0.85	1.11	1.51	3.20	58	0.99
rho[14]	1.13	0.05	0.42	0.56	0.85	1.04	1.46	1.91	75	0.99
rho[15]	2.12	0.38	1.15	0.59	1.15	1.96	2.61	4.64	9	1.25
rho[16]	1.32	0.11	0.52	0.69	0.98	1.24	1.43	2.40	25	1.14
rho[17]	2.00	0.15	1.06	0.50	1.15	1.89	2.61	4.42	47	1.02
rho[18]	2.22	0.63	1.35	0.58	1.04	1.92	3.31	5.38	5	1.50
rho[19]	2.23	0.71	1.40	0.65	1.03	1.88	3.09	4.98	4	1.47
rho[20]	1.56	0.13	0.78	0.56	0.96	1.56	1.89	2.91	35	1.03
rho[21]	1.83	0.39	1.11	0.61	0.98	1.65	2.38	5.03	8	1.34
alpha	6.85	1.71	3.51	0.35	4.52	7.01	9.97	11.57	4	1.70
sigma	14.56	0.92	1.66	12.34	12.59	14.89	15.99	16.97	3	1.87
y2[1]	7.65	2.03	14.46	-18.87	-1.15	6.95	16.70	38.27	51	1.08
y2[2]	4.54	1.82	13.84	-21.40	-4.62	4.19	15.70	29.91	58	1.07
y2[3]	3.02	2.19	16.66	-32.36	-7.81	4.53	14.52	29.90	58	1.02
y2[4]	5.35	3.27	17.20	-28.60	-5.94	6.36	18.12	35.59	28	1.08
y2[5]	2.84	2.02	16.84	-25.29	-8.32	2.45	14.31	33.27	69	1.02

dentro da função stan(), os argumentos iter=200 refere-se à quantidade de iterações que o modelo fez, warmup=100 refere-se ao *burn-in*, e chains=1 diz respeito à quantidade de cadeias que estão sendo simuladas.

APÊNDICE B – CÓDIGOS STAN DOS CLASSIFICADORES GPC E TPC

Abaixo seguem os códigos em Stan/R dos modelos de classificação utilizados neste trabalho.

Modelo GPC

Modelo binário

```
model_logistic.GP="

// função para implementar a f.c.  (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){
int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}
}
data{
int<lower=1> N; //quantidade de dados de treino
int<lower=1> D; //dimensao do vetor de dados x
vector[D] x[N]; //dados de entrada
int<lower=0,upper=1> z[N]; //vetor de saida observ.
}
transformed data{
real delta=1e-9; // aqui seria o jitter
real w=1;
real a=1;
}
parameters{
vector<lower=0>[D] rho;  //comprimento-escala
real<lower=0> alpha; //sinal de variancia
vector[N] eta; //vetor z de variaveis iid normal
}
model{
vector[N] f;
{
matrix[N,N] L_K=L_cov_exp_quad_ARD(x,alpha,rho,delta);
f=L_K*eta; //amostrando de uma Gaussiana Multivariada
}

// distribuições a priori

rho ~ gamma((a/2),(a/(2*w)));
alpha ~ normal(0,1);
eta ~ normal(0,1);

z ~ bernoulli_logit(f);//verossimilhanca
}
"
```

Modelo Multiclasse

```
model_multiclasse.GP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){

int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);
```

```
for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}
}
data{
int<lower=1> N; // quantidade de pontos de treino
int<lower=1> D; // dimensão do vetor de entrada
int<lower=1> C; // quantidade de classes
vector[D] x[N]; // dados de entrada para treino
int<lower=1> CN; // dimensão do vetor extendito f
int<lower=0,upper=1> y[N,C]; // matrix de rótulos
}
transformed data{
real delta=1e-9; // aqui seria o jitter
real w=1;
real a=1;
}
parameters{
vector<lower=0>[D] rho[C]; // comprimento escala
real<lower=0> alpha[C]; // sinal de variancia
vector[CN] eta;
}
model{
vector[CN] f;
matrix[N,C] fextent;
{
matrix[CN,CN] L_K;  // matrix em bloco diagonal
for(i in 1:CN)
for(j in 1:CN)
 L_K[i,j]<-0;

for(i in 1:C)
L_K[((i-1)*N+1):(i*N),((i-1)*N+1):(i*N)]<-L_cov_exp_quad_ARD(x,
alpha[i], rho[i],delta);

f=L_K*eta; //amostrando de uma Gaussiana Multivariada

for(i in 1:N)
for(j in 1:C)
   fextent[i,j]<-f[(((j-1)*N+1)+(i-1))];
}

// distribuições a priori

for(i in 1:C)
rho[i] ~ gamma((a/2),(a/(2*w)));
alpha ~ normal(0,1);
eta ~ normal(0,1);

for(i in 1:N)
 y[i]~multinomial(softmax(to_vector(fextent[i])));
}
"
```

Modelo preditivo

Modelo binário

```
model_logistic_pred.GP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){
int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}
}
data {
  int<lower=1> D; //dimensão do vetor de entrada
  int<lower=1> N1; // quantidade de pontos de dados de treino
```

```
  vector[D] x1[N1]; // vetor de dimensão N1 x D
  int<lower=0, upper=1> z1[N1]; //vetor de saida
  int<lower=1> N2; //quantidade de pontos de teste
  vector[D] x2[N2]; // vetor de dimensão N2 x D
  vector<lower=0>[D] rho; //comprimento-escala
  real<lower=0> alpha; //sinal de variancia
}
transformed data {
  real delta = 1e-9;
  int<lower=1> N = N1 + N2;
  vector[D] x[N];
  matrix[N,N] L_K;

  for (n1 in 1:N1) x[n1] = x1[n1];
  for (n2 in 1:N2) x[N1 + n2] = x2[n2];

  L_K=L_cov_exp_quad_ARD(x,alpha,rho,delta);
}
parameters {
 vector[N] eta;
}
transformed parameters {
  vector[N] f;

{
f=L_K*eta; //amostrando de uma Gaussiana Multivariada
}
}
model {

  // priori para o modelo
eta ~ normal(0, 1);

  z1 ~ bernoulli_logit(f[1:N1]);
}
"
```

OBS: Dentro do bloco **transformed parameters**, definimos o vetor **f** de dimensão N como um parâmetro. Este vetor, que será amostrado, é a concatenação dos valores latentes observados e de teste. Os valores latentes entram como argumento da função logística, tal que $\mathbf{f}[i]$ é a probabilidade de $z1[i] = 1$. Apenas os valores **z1** foram observados e por isso são amostrados. Não há um vetor **z2** porque o Stan não suporta amostragem discreta ainda; ao invés disso, as predições são feitas guardando os valores latentes **f**[N1+1:N] para uso posterior como argumento da função logística para possibilitar uma predição de classe $\hat{\pi}_*$. Este mesmo processo se aplica a todos os modelos utilizados neste trabalho.

Modelo multiclasse

```
    model_multiclasse_pred.GP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){
int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
//return K;
}
}
data{
int<lower=1> N1; // quantidade de pontos de treino
int<lower=1> D; // dimensão do vetor de entrada
int<lower=1> C; // quantidade de classes
vector[D] x1[N1]; // dados de treinamento
int<lower=1> CN1; // dimensão do vetor extendido f
int<lower=1> CN2; // dimensão do vetor extendido f
int<lower=0,upper=1> y1[N1,C]; // matrix de rótulos
int<lower=1> N2; // quantidade de pontos de teste
vector[D] x2[N2]; // dados de entrada dos pontos de teste
vector<lower=0>[D] rho[C]; //comprimento-escala
real<lower=0> alpha[C]; //sinal de variancia
}
transformed data{
real delta=1e-9; // aqui seria o jitter
int<lower=1> N = N1 + N2;
int<lower=1> CN= CN1+CN2;
   vector[D] x[N];
matrix[CN,CN] L_K;
//matrix[CN,CN] L;
```

```
   for (n1 in 1:N1) x[n1] = x1[n1];
   for (n2 in 1:N2) x[N1 + n2] = x2[n2];

for(i in 1:CN)
for(j in 1:CN)
 L_K[i,j]<-0;

for(i in 1:C)
 L_K[((i-1)*N+1):(i*N),((i-1)*N+1):(i*N)]<-
 L_cov_exp_quad_ARD(x,alpha[i],rho[i],delta);

}
parameters{
vector[CN]eta;
}
transformed parameters{
vector[CN] f;
matrix[N,C] fextent;

f=L_K*eta; //amostrando de uma Gaussiana Multivariada
for(i in 1:N)
 for(j in 1:C)
  fextent[i,j]<-f[(((j-1)*N+1)+(i-1))];
}
model{

// priori para o modelo
eta ~ normal(0, 1);

for(i in 1:N1)
 y1[i]~multinomial(softmax(to_vector(fextent[i])));
}
"
```

Modelo TPC

Modelo binário

```
    model_logistic.TP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){

int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}
}

data{

int<lower=1> N;
int<lower=1> D;
vector[D] x[N];
int<lower=0,upper=1> z[N];
real<lower=0> v; // graus de liberdade
}

transformed data{
real delta=1e-9;
real w=1;
real a=1;

vector[N] mu;
for(i in 1:N) mu[i]<-0; // vetor de médias nulo
}

parameters{
vector<lower=0>[D] rho;
real<lower=0> alpha;
vector[N] eta;
real<lower=0> u;
}

model{
vector[N] f;
```

```
{
matrix[N,N] L_K=L_cov_exp_quad_ARD(x,alpha,rho,delta);
f=mu+sqrt(v/u)*(L_K*eta); // amostrando de uma t
}
// distribuições a priori

rho ~ gamma((a/2),(a/(2*w)));
alpha ~ normal(0,1);
eta ~ normal(0,1);
u ~ chi_square(v);

z ~ bernoulli_logit(f);
}
"
```

OBS: O código do modelo TPC acima é para graus de liberdade fixos. Para podermos implementar o TPC atribuindo distribuição a priori aos graus de liberdade v, procedemos da seguinte forma:

1. Retiramos a declaração de variável "real<lower=0> v;" do bloco **data**;
2. inserimos a declaração de variável: "real<lower=0> v;" no bloco **parameters**;
3. dentro do bloco **model** acima, inserimos a seguinte linha de código na parte de atribuição de distribuição a priori: v $\sim$ *gamma*(2, 0.1);.

Modelo multiclasse

```
model_multiclasse.TP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){

int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}

K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}

}

data{
int<lower=1> N; // quantidade de pontos de treino
int<lower=1> D; // dimensão do vetor de entrada
int<lower=1> C; // quantidade de classes
vector[D] x[N]; // dados de entrada para treino
int<lower=1> CN; // dimensão do vetor extendito f
int<lower=0,upper=1> y[N,C]; // matrix de rótulos
real<lower=0> v;  // graus de liberdade
}

transformed data{
real delta=1e-9;
real w=1;
real a=1;

vector[CN] mu;
for(i in 1:CN) mu[i]<-0; // vetor de médias nulo

}

parameters{

vector<lower=0>[D] rho[C]; // comprimento escala
real<lower=0> alpha[C]; // sinal de variancia
vector[CN] eta;
real<lower=0> u;
}

model{

vector[CN] f;
matrix[N,C] fextent;
{
 matrix[CN,CN] L_K;  // matrix em bloco diagonal

 for(i in 1:CN)
 for(j in 1:CN)
```

```
  L_K[i,j]<-0;

for(i in 1:C)
L_K[((i-1)*N+1):(i*N),((i-1)*N+1):(i*N)]<-
L_cov_exp_quad_ARD(x,alpha[i],rho[i],delta);

f=mu+sqrt(v/u)*(L_K*eta); // amostrando de uma t

for(i in 1:N)
   for(j in 1:C)
   fextent[i,j]<-f[(((j-1)*N+1)+(i-1))];
}

// distribuições a priori

for(i in 1:C)
rho[i] ~ gamma((a/2),(a/(2*w)));
alpha ~ normal(0,1);
eta ~ normal(0,1);
u ~ chi_square(v);

for(i in 1:N)
 y[i]~multinomial(softmax(to_vector(fextent[i])));
}
"
```

OBS: Assim como o código para TPC binário anterior, o código para TPC multiclasse acima também é para graus de liberdade fixos. Para podermos implementar o TPC multiclasse atribuindo distribuição a priori aos graus de liberdade v, podemos seguir os mesmos passos anteriormente mencionados.

Modelo preditivo

Modelo binário

```
    model_logistic_pred.TP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){

int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}
K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}
}

data {
  int<lower=1> D; //dimensão do vetor de entrada
  int<lower=1> N1;
  vector[D] x1[N1]; // vetor de dimensão N1 x D
  int<lower=0, upper=1> z1[N1];
  int<lower=1> N2;
  vector[D] x2[N2]; // vetor de dimensão N2 x D
  real<lower=0> v;
  vector<lower=0>[D] rho;
  real<lower=0> alpha;
}
transformed data {
  real delta = 1e-9;
  int<lower=1> N = N1 + N2;
  vector[D] x[N]; // original era isso: real x[N]
  matrix[N,N] L_K;
  vector[N] mu;

  for (n1 in 1:N1) x[n1] = x1[n1];
  for (n2 in 1:N2) x[N1 + n2] = x2[n2];

  for(i in 1:N) mu[i]<-0; // vetor de médias nulo

  L_K=L_cov_exp_quad_ARD(x,alpha,rho,delta);
}

parameters {
  vector[N] eta;
  real<lower=0> u;
```

```
}

transformed parameters {

vector[N] f;
{
f=mu+sqrt(v/u)*(L_K*eta); // amostrando de uma t
}
}
model {
eta ~ normal(0,1);
u ~ chi_square(v);

  z1 ~ bernoulli_logit(f[1:N1]);
}
"
```

Modelo multiclasse

```
model_multiclasse_pred.TP="

// função para implementar a f.c. (ARD)
functions{
matrix L_cov_exp_quad_ARD(vector[] x, real alpha,
vector rho, real delta){

int N=size(x);
matrix[N, N] K;
real neg_half=-0.5;
real sq_alpha=square(alpha);

for(i in 1:(N-1)){
K[i,i]=sq_alpha + delta;
for(j in (i+1):N){
K[i,j]=sq_alpha*exp(neg_half*dot_self((x[i]-
x[j])./rho));
K[j,i]=K[i,j];
}
}

K[N,N]=sq_alpha+delta;
return cholesky_decompose(K);
}

}

data{
int<lower=1> N1; // quantidade de pontos de treino
int<lower=1> D; // dimensão do vetor de entrada
int<lower=1> C; // quantidade de classes
vector[D] x1[N1]; // dados de treinamento
int<lower=1> CN1; // dimensão do vetor latente treino
int<lower=1> CN2; // dimensão do vetor latente  teste
int<lower=0,upper=1> y1[N1,C]; // matrix de rótulos
int<lower=1> N2; // quantidade de pontos de teste
vector[D] x2[N2]; // dados de entrada de teste
vector<lower=0>[D] rho[C];
real<lower=0> alpha[C];
real<lower=0> v;
}

transformed data{
real delta=1e-9;
int<lower=1> N = N1 + N2;
int<lower=1> CN= CN1+CN2;
    vector[D] x[N];
matrix[CN,CN] L_K;
vector[CN] mu;

for(i in 1:CN) mu[i]<-0; // vetor de médias nulo

   for (n1 in 1:N1) x[n1] = x1[n1];
   for (n2 in 1:N2) x[N1 + n2] = x2[n2];

for(i in 1:CN)
for(j in 1:CN)
 L_K[i,j]<-0;

for(i in 1:C)
  L_K[((i-1)*N+1):(i*N),((i-1)*N+1):(i*N)]=L_cov_exp_quad_ARD(x,
  alpha[i],rho[i],delta);
}

parameters{
vector[CN] eta;
   real<lower=0> u;
}

transformed parameters{

matrix[N,C] fextent;
vector[CN] f;
{
f=mu+sqrt(v/u)*(L_K*eta); // amostrando de uma t-student
```

```
for(i in 1:N)
 for(j in 1:C)
  fextent[i,j]<-f[(((j-1)*N+1)+(i-1))];
}
}
model{
eta ~ normal(0,1);
u ~ chi_square(v);

for(i in 1:N1)
 y1[i]~multinomial(softmax(to_vector(fextent[i])));
}
"
```