Engenharia do Conhecimento

Resumo

Conteúdo

Capítulo 1

Modelos

1.1 Classificação e Regressão

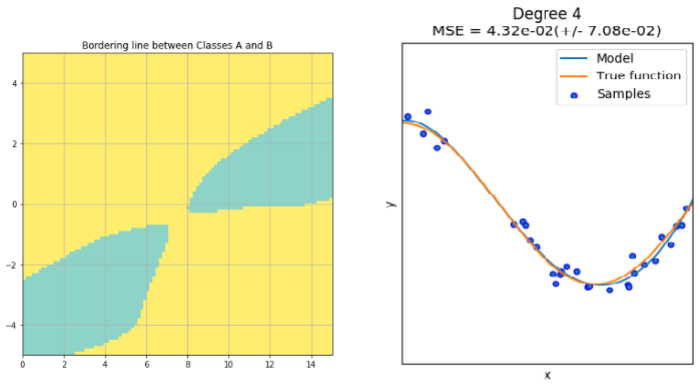
1.1.1 Classificação

O objetivo da classificação é encontrar uma separação entre duas ou mais populações. Encontrar a separação ideal (para além da linear) é um problema $\mathbb{NP}$ difícil.

Teoricamente, se as funções de distribuição das populações são conhecidas, é possível encontrar a separação perfeita entre classes. Para estimar a delineação de qualquer função, usam-se *Universal Function Approximators*.

1.1.2 Regressão

O objetivo da regressão é fazer a melhor aproximação possível da função que gerou os dados. Novamente são necessários *Universal Function Approximators*.



Classificação Regressão

1.2 Modelos em Árvore

Um modelo em árvore é uma estrutura hierárquica de condições, onde as folhas contêm o resultado.

1.2.1 Árvores de Decisão

O objetivo de uma árvore de decisão é, dado um problema multidimensional, gerar o melhor classificador possível.

Entropia

De forma a encontrar o melhor critério para a árvore usa-se entropia, a medida da informação necessária para descrever um sistema:

$$Entropia = -\sum_i p_i \cdot \log_2 p_i$$

Onde p_i é a probabilidade do evento i ocorrer, pelo que $\sum_i p_i = 1$. Por exemplo, a entropia do lançamento de uma moeda:

$$-p(Heads) \cdot \log_2 p(Heads) - p(Tails) \cdot \log_2 p(Tails)$$

$$-0.5 \cdot \log_2 0.5 - 0.5 \cdot \log_2 0.5 = 1$$

A entropia é máxima quando a confusão é máxima e entropia zero ocorre quando não há variabilidade. O objetivo de um bom classificador é dividir os dados tal que a entropia do resultado seja mínima.

ID3

Neste algoritmo, após encontrar o primeiro critério, se os nós resultantes tiverem entropia maior que zero (não puro), repete-se o processo recursivamente para cada ramo, gerando uma árvore de decisão.

CART

Utiliza o mesmo princípio que o ID3, mas tipicamente utiliza o critério gini, por oposição á entropia. Segundo o gini, o erro esperado ao classificar exemplos aleatoriamente com probabilidade p para a classe positiva e $1 - p$ para a classe negativa é:

$$2p(1 - p)$$

Critérios de Paragem

O critério de paragem usado pode ter um grande efeito no comportamento de uma árvore. Os critérios mais comuns são:

- Profundidade máxima
- Número mínimo de samples por split
- Número mínimo de samples por folha
- Número máximo de folhas
- Mínimo decremento de impureza

1.2.2 Árvores de Regressão

Preveem uma variável contínua através de passos em que a previsão é constante.

CART

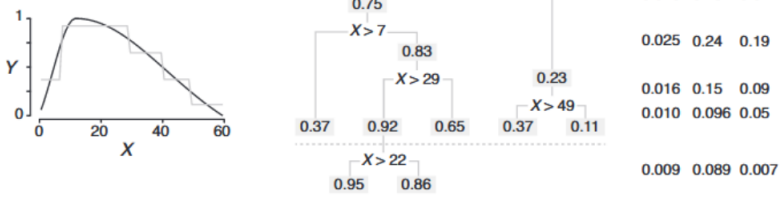
O CART pode ser usado para regressão, através do uso do Mean Squared Error como medida de impureza:

$$MSE = \frac{1}{N}\sum_i (y_i - \overline{y})^2$$

Deve usar-se o MSE á direita e esquerda do critério de divisão para encontrar o melhor ponto

$$Cost = \frac{N_{left}}{N} \cdot MSE(D_{left}) + \frac{N_{right}}{N} \cdot MSE(D_{right})$$

O ponto de divisão D é definido iterando por todos os valores possíveis da variável selecionada no ramo da árvore dado e selecionando o melhor ponto de divisão. Com CART só é possível efetuar divisão binária.



1.2.3 Resumo

Are Decision Trees...

- Simple?
  - **Mostly** – eventually they can become very branched, but typically are manageable **and are explainable**. They are suitable to overfitting, though
- Stable?
  - **YES** – No matter the data or the order, the result is always the same. They are very sensitive on the hyperparameters, which can totally change how a tree looks
- Fast to learn?
  - **YES** – Basically just counts and ratios. Very efficient even for very large data sets
- Fast to make predictions?
  - **YES** – Each decision is very simple and the depth of the tree is typically not a factor. Complexity is O(D) where D is the depth of the tree
- Updateable?
  - **NO** – New data may change the probabilities of each factor, potentially changing the tree [although counts may be stored and simply updated as needed]

1.3 Regressão Linear

A regressão linear assume uma relação linear entre os valores de input e os targets. A melhor relação linear entre x e y é:

$$y = \alpha + \beta x$$

O objetivo é encontrar os melhores valores de α e β que minimizem uma medida predefinida de erro.

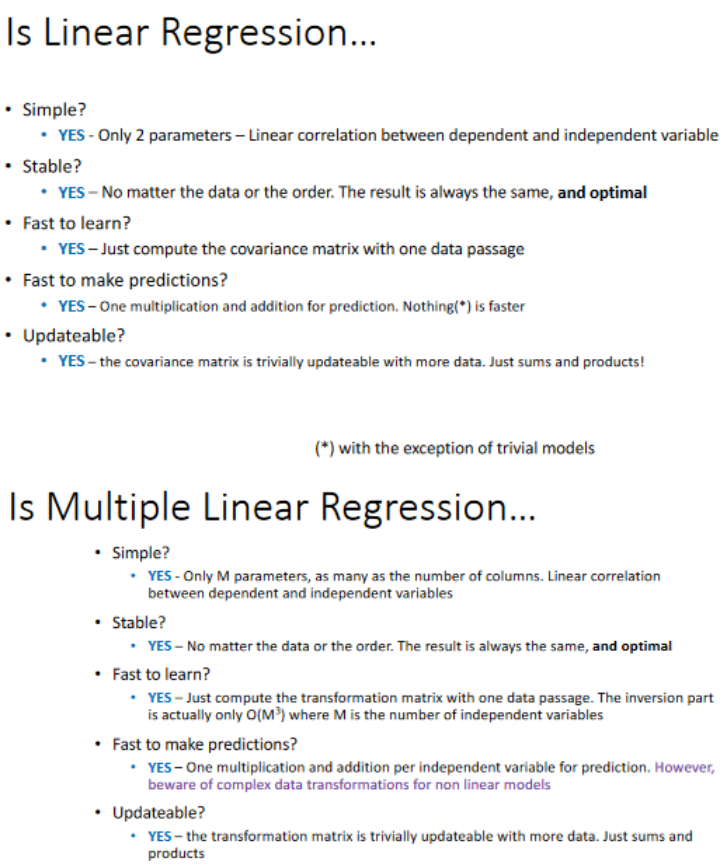
É possível usar regressão linear mesmo quando a relação entre as variáveis não é linear. Uma solução trivial consiste em transformar as variáveis independentes.

1.3.1 Regressão Linear Múltipla

A regressão linear pode ser aplicada mesmo quando existem várias variáveis independentes. Um modelo multilinear pode ser escrito como:

$$y_i \approx \sum_{j=0}^{m} \beta_j \times x_j^i = \vec{\beta} \cdot \vec{x_i}$$

1.3.2 Resumo



Algumas limitações da regressão linear são:

- Só conseguem identificar relações entre variáveis X e y (não são $universal\ function\ approximators$)
- Com um número de variáveis elevado vai sempre ocorrer overfitting

No entanto, é ótima.

1.4 Modelos Lineares Regularizados

Tratam-se de modelos em que os coeficientes são constringidos, de forma a que permitam a contribuição de outras variáveis no modelo, podendo ser vistos como uma extensão da regressão linear múltipla. Estes modelos incluem mais hiperparâmetros, que permitem reduzir a sua variância, reduzindo também a capacidade de overfitting.

1.4.1 Ridge Regression

Na função de custo habitual, em vez de usar apenas o mean squared error, inclui também a soma dos quadrados dos coeficientes (θ):

$$cost(\theta) = MSE(\theta) + \frac{\alpha}{2}\sum_{i=1}^{n}\theta_i^2$$

Em que α restringe a penalização dos coeficientes.

1.4.2 Lasso

Usa-se regularização L1, i.e., os coeficientes são penalizados em módulo:

$$cost(\theta) = MSE(\theta) + \alpha\sum_{i=1}^{n}|\theta_i|$$

1.4.3 Elastic Nets

Formam um meio termo entre as duas abordagem anteriores:

$$cost(\theta) = MSE(\theta) = \alpha_L\sum_{i=1}^{n}|\theta_i| + \alpha_R\sum_{i=1}^{n}\theta_i^2$$

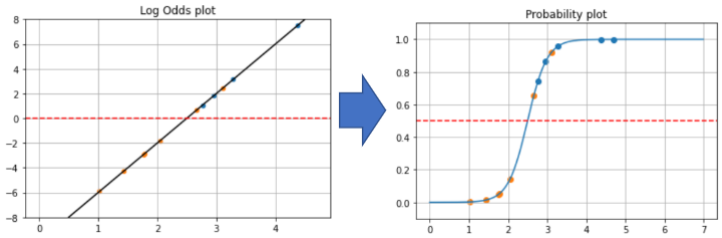
1.5 Regressão Logística

Tem como objetivo a classificação binária. Calcula a probabilidade de uma instância ser positiva. Essa probabilidade pode ser definida através de uma função logística:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot x)}}$$

Para cada valor de x podemos calcular a probabilidade de uma instância ser positiva. Esta função sigmoide pode ser transformada numa função linear:

$$\log\left(\frac{p}{1-1}\right) = \beta_0 + \beta_1 \cdot x$$



O fit de uma regressão logística consiste em fazer fit desta reta, minimizando uma loss function através de gradient descent. Esse algoritmo consiste em fazer passos repetidos na direção oposta ao gradiente de uma função no ponto atual, já que essa é a direção da descida com maior declive.

1.6 Linear Discriminant Analysis

Tem como objetivo encontrar uma combinação linear de features para separar classes, assumindo que as variáveis de todas as classes seguem uma distribuição normal.

É possível prever se um ponto pertence a uma dada classe se o logaritmo dos rácios de probabilidade $\log \frac{P(G=k|X=x)}{P(G=l|X=x)}$ for maior que um certo threshold T.

1.7 Classificador de Bayes

Segundo o teorema de bayes tem-se que:

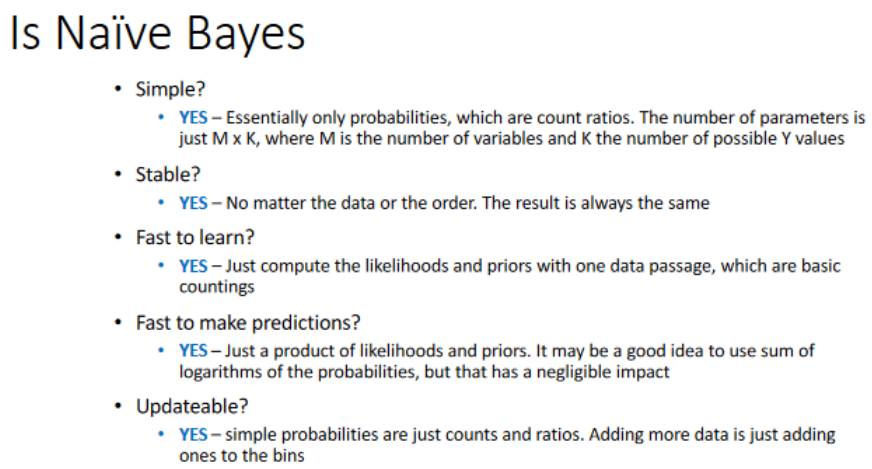
$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$$

1.7.1 Naive Bayes

Segundo o algoritmo de naive bayes, a probabilidade de uma instância x pertencer á classe C_k é:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

1.7.2 Resumo



1.8 K-NN

1.8.1 Medidas de Distância

Existem várias formas de medir a distância entre dois pontos X e Y.

Distância Euclideana

$$D(X,Y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Distância de Manhattan

$$D(X,Y) = \sum_i |x_i - y_i|$$

Distância de Jaccard

Baseia-se na representação dos items como valores binários. Representa o rácio de elementos partilhados entre todos os elementos existentes.

1.8.2 K-NN para Classificação

Escolhido um K, um elemento é classificado como os K elementos mais próximos. Pode levar a empates mesmo com K ímpar.

Com $K = 1$ tipicamente tem-se alta variância, com K muito elevado tem-se bias alto.

1.8.3 K-NN para Regressão

Geralmente usa-se a mediana de todos os vizinhos para fazer previsões, o que pode levar a aumentos em passos, mudando apenas quando a vizinhança muda.

1.8.4 K-NN Distance Weighting

Por vezes pode ser útil atribuir pesos a cada distância.

- Distância inversa:
$$w_i = \frac{1}{d_i}$$
- Gaussian kernels:
$$w_i = e^{-\frac{d_i^2}{kw}}$$

1.8.5 Resumo

Is K nearest Neighbours

- Simple?
  - **Not really** – Even though the principles are simple, the problem of inference grows linearly with the size of the dataset. The model is the dataset
- Stable?
  - **YES (kind of)** –The result is most of the times the same, Yet in classification when there are ties, the result may depend on the training set order
- Fast to learn?
  - **YES** – No training. The model is the data [eventually some processing for space partitioning]
- Fast to make predictions?
  - **NO** – Typically for large datasets it is a slow algorithm
- Updateable?
  - **YES** –more data is trivially added to the model without any problem or refitting. Yet, eventually it may be required to make a new data partition for efficient searching.

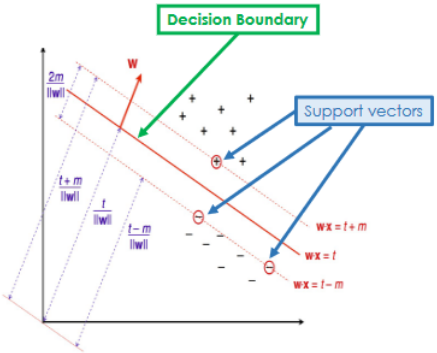
1.9 Support Vector Machines

1.9.1 Dados Linearmente Separáveis

Admitem infinitas decision boundaries que separam as diferentes classes, sendo algumas intuitivamente melhores que outras.

1.9.2 SVM

Os exemplos de treino mais próximos da decision boundary são chamados de support vectors.



Onde t é a decision threshold, x é o valor da instância e w é o peso.
A decision boundary de uma SVM é definida como a combinação linear dos support vectors e a margem é $m/||w||$, onde m é a distância entre a decision boundary e as instâncias de treino mais próximas.

1.9.3 Maximizar a Margem

Multiplicadores de Lagrange

É uma estratégia para encontrar os máximos e mínimos locais de uma função, sujeita a restrições de igualdade. Por exemplo, para encontrar o máximo ou mínimo de uma função $f(x)$ sujeita á restrição $g(x) = 0$ usa-se a função Lagrangiana:

$$\Lambda(x, \lambda) = f(x) + \lambda \cdot g(x)$$

Adicionar as restrições com multiplicadores α_i para cada exemplo de treino dá a função de Lagrange:

$$\Lambda(w, t, \alpha_1, ..., \alpha_n) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i(y_i(w \cdot x_i - t) - 1)$$

Que pode ser simplificado para um problema de dupla otimização (restrições de positividade e uma restrição de igualdade).

Esta forma dupla de otimização ilustra que procurar pela decision boundary com margem de decisão máxima é equivalente a procurar pelos support vectors e que o problema de otimização é inteiramente definido pelo produto dos pares entre as instâncias de treino.

1.9.4 Soft Margin SVM

É possível estender o conceito de SVM para classes que não são estritamente separáveis, através da introdução de slack variables, ξ. É adicionada uma para cada exemplo, o que permite algumas instâncias estarem dentro da margem ou até do lado errado da decision boundary (margin errors).

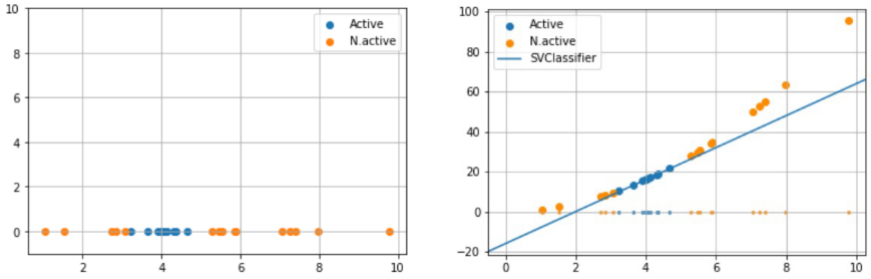
$$w^*, t^*, \xi_i^* = argmin \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$

Sujeito a $y_i(w \cdot x_i - t) \geq 1 - \xi_i$ e $\xi_i \geq 0, 1 \leq i \leq n$. Onde C é um parâmetro definido previamente, que balança a maximização com a slack variable. Pode ser pensado como uma constante de regularização, que deve ser diminuída para noisy data. Um valor baixo de C vai relaxar o modelo e permitir alguns exemplos estarem dentro da margem.

1.9.5 Kernels

Uma função kernel mede a semelhança entre qualquer par de inputs, permitindo operar num espaço com mais dimensões, sem ter de calcular as coordenadas nesse espaço. Representa apenas a relação entre instâncias, não é necessário mudar o problema de otimização nem precisam de ser explicitamente calculados (kernel trick).

No seguinte exemplo foi usado um kernel de transformação polinomial para separar dois tipos de instâncias



Radial Basis Function Kernel

É uma das mais poderosas adições ao SVC:

$$G(X_i, X_j) = e^{-\gamma \cdot (X_i - X_j)^2}$$

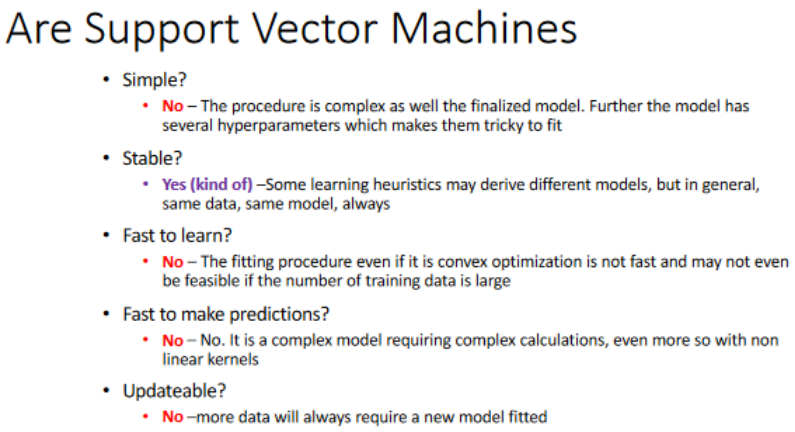
Funciona de forma semelhante aos Gaussian weights no K-NN, mas mapeia o espaço das amostras em dimensões infinitas. O parâmetro γ define a influencia de um único parâmetro de treino, valores baixos indicam muito e valores altos indicam pouco.

1.9.6 Support Vector Regression

Procura o "hypertube"mais pequeno capaz de alojar a maioria dos pontos, minimizando as distâncias das instâncias que não conformam dentro do "tube".

São ignoradas as instâncias dentro do "hypertube", sendo apenas contados os pontos fora. Pode ter resultados inesperados por se tratar de um modelo muito sensível ás distribuições dos dados e pode ser usado com kernels que dobram a forma do espaço dimensional, podendo ser usado para todo o tipo de curvas.

1.9.7 Resumo



1.10 Ensemble Models

A ideia principal é que vários modelos simples têm melhor comportamento que um modelo sofisticado.

1.10.1 Bagging

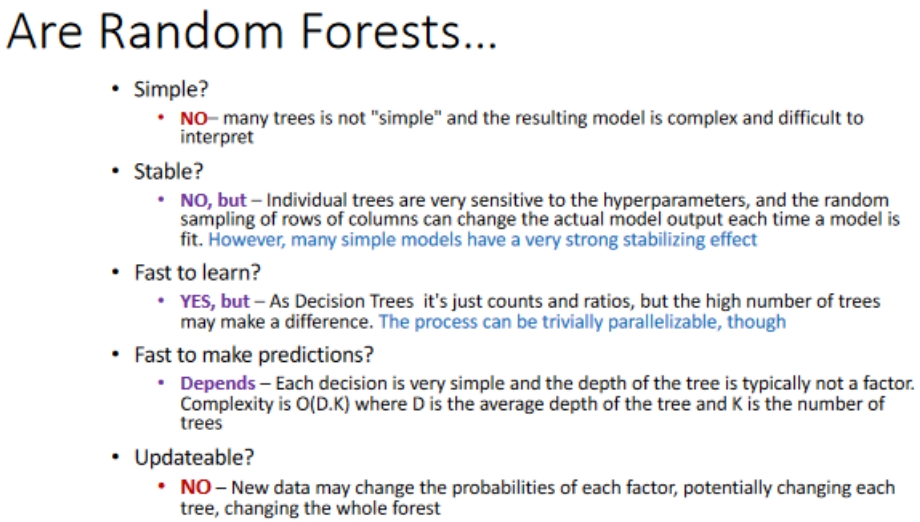
Consiste em treinar uma ensemble de modelos em bootstrapped datasets, criados a partir do conjunto de treino inicial.

No Monte Carlo bootstrapping, dado um conjunto de treino D, são gerados k novos conjuntos de treino D_i fazendo resampling dos dados uniformemente com substituição.

Random Forests

É o algoritmo de bagging mais comum, procedendo da seguinte forma:

- Faz-se bootstrap dos dados
- Seleciona-se aleatoriamente um subconjunto de todas as colunas
- Faz-se fit a uma árvore de decisão
- Repete-se para K árvores



1.10.2 Pasting

Idêntico a bagging, mas ignorando o processo de bootstrapping. São gerados vários modelos, cada um exposto a diferentes partições de treino e teste.

1.10.3 Boosting

Consiste em iterativamente fazer fit de classificadores fracos. Após adicionar cada modelo fraco, é aumentado o peso dos dados mal classificados e possivelmente diminuído o peso dos dados classificados corretamente.

AdaBoost

Este algoritmo repete os seguintes passos:

- É atribuído um peso a cada instância do conjunto de treino para cada classificador (inicialmente $1/N$).
- Treina um modelo fraco
- Adapta os pesos de cada sample de acordo com o erro da previsão

A importância de cada classificador pode ser definida como:

$$\alpha_t = \frac{1}{2} \ln \frac{(1 - TotalError)}{TotalError}$$

Sendo que cada peso é ajustado após ser calculada a sua importância.

Is AdaBoost...

- Simple?
  - NO– as Bagging models, many models is not "simple" and the resulting model is complex and difficult to interpret
- Stable?
  - Many times – But in general yes. Stable estimators will produce stable boosting models. Yet, there are weird fluctuations with the number of models resulting from the learning rate hyperparameter
- Fast to learn?
  - Not necessarily – AdaBoost requires weak models which are simple and fast. However it may require many such simple models for convergence. Yet ultimately as the Boosting procedure is sequential it is not parallelizable
- Fast to make predictions?
  - Depends – Similar to Random Forests. Each model is simple, complexity is a linear factor of O(C.K) where C is average model complexity and K is the number of models. Also AdaBoost models many times should be simpler than RFs
- Updateable?
  - NO – New data may change the probabilities of each factor, potentially changing each model, changing the whole procedure

Gradient Boosting

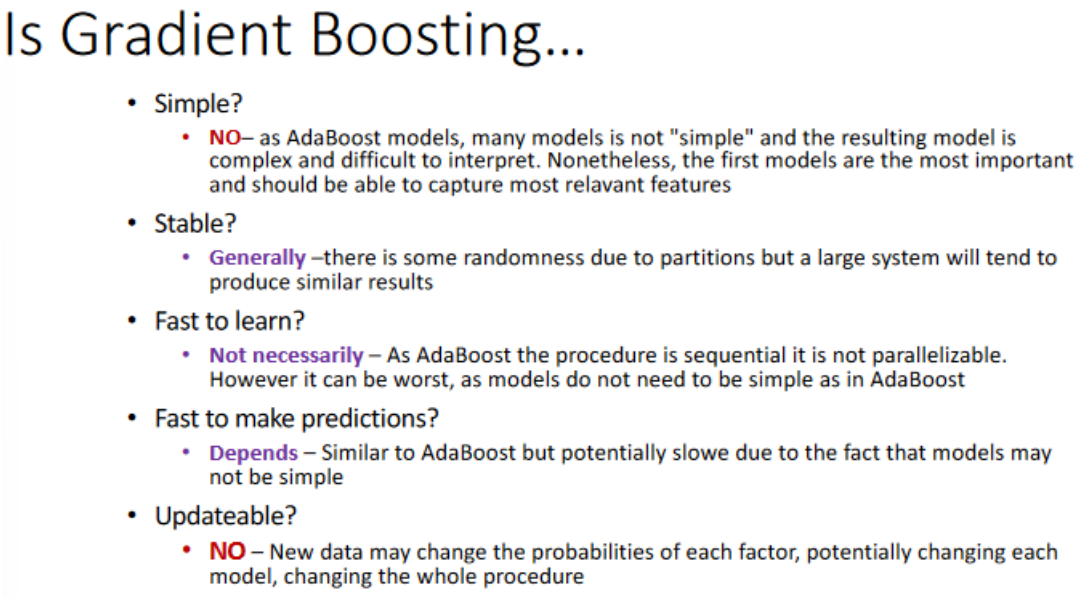
Consiste na ideia de fazer um modelo simples, avalia os seus erros e faz um modelo novo que tenta prever esses erros. Este processo é repetido até ser atingido um critério de paragem.

Um parâmetro de learning rate (r) escala (para baixo) a importância dos classificadores subsequentes.

$$Result = R_1 + (1-r) \cdot R_2 + (1-r)^2 \cdot R_3 + ... + (1-r)^{(k-1)} \cdot R_k$$

Pode ser usado para classificação ou regressão.

Uma outra versão do gradient boosting, o XGBoost utiliza também regularização, treinando iterativamente árvores de decisão, com cada iteração usando os erros do modelo anterior para fazer fit do próximo modelo.



1.10.4 Stacking

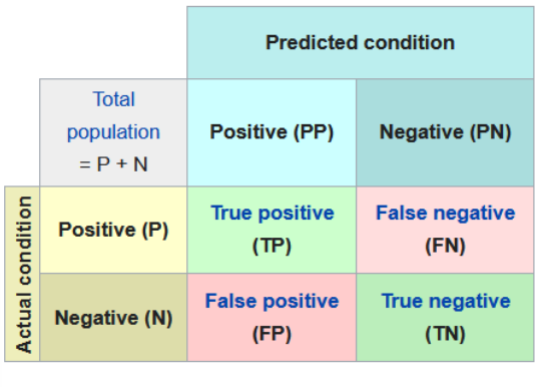
Modelos de stacking não requerem votações ou avaliação OOB. São treinados vários modelos e os resultados são usados como variáveis dependentes para previsão.

Capítulo 2

Avaliação de Modelos

A avaliação de modelos é essencial ao sucesso em aprendizagem supervisionada. Diferentes tipos de modelos requerem diferentes formas de avaliação, mas todos eles devem ser comparados á performance no conjunto de teste.

2.1 Matriz de Confusão



2.2 Modelos de classificação

2.2.1 Modelos de classificação binários

Accuracy

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN - FP + FN}$$

Compara o número de previsões corretas ao total de previsões.

Recall

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

Representa o rácio entre verdadeiros positivos e o número total de positivos.

Precision

$$PPV = \frac{TP}{TP + FP}$$

Representa o rácio entre verdadeiros positivos e o número de exemplos classificados como positivos

F1 Score

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

Tem em conta a precision e o recall.

Matthews Correlation

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Retorna um valor entre 0 e 1, onde:

- 1 representa uma previsão perfeita
- 0 representa uma previsão equivalente a aleatória
- -1 indica um desacordo total entre previsão e observação

Não é um bom indicador de quão similar é a um modelo aleatório porque é dependente do dataset.

2.2.2 Modelos de Classificação N-ários

Accuracy

É possível usar accuracy neste tipo de modelos, tendo em conta o rácio de elementos corretamente classificados em todas as amostras. Só é confiável se o dataset for balanceado, i.e., tiver números semelhantes de amostras em todas as classes.

Matthews Correlation

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{\left(s^2 - \sum_k^K p_k^2\right) \times \left(s^2 - \sum_k^K t_k^2\right)}}$$

Onde:

- $t_k = \sum_i C_{ik}$ número de ocorrências da classe k
- $p_k = \sum_i C_{ki}$ número de previsões da classe k
- $c_k = \sum_k C_{kk}$ número de previsões corretas
- $s = \sum_i \sum_j C_{ij}$ número de amostras total

Esta métrica funciona bem mesmo com classes não balanceadas. 1 é a melhor pontuação e o mínimo é um valor entre -1 e o, dependendo das distribuições y.

2.3 Modelos de Regressão

Geralmente existem duas considerações ao avaliar modelos de regressão:

- Quanta variância da variável dependente está a ser capturada
- Qual o erro médio de uma previsão

2.3.1 Ratio of the Variance Explained

$$RVE = 1 - \frac{\sum(y_i - \hat{y})^2}{Var(y)}$$

Dá uma medida relativa da qualidade geral do modelo. 1 indica um regressor perfeito, 0 é um regressor trivial (prevê a mediana).

2.3.2 Root Mean Squared Error

$$rmse = \sqrt{\frac{\sum(y_i - \hat{y})^2}{N}}$$

Avalia o erro médio do modelo. É expresso nas unidades da variável dependente.

2.3.3 Pearson Correlation

$$\rho_{y,\hat{y}} = \frac{cov(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}$$

Verifica a correlação entre as previsões e a verdade

2.4 Validação de Modelos

É necessário validar um modelo para identificar a presença de overfitting ou underfitting. Overfitting representa a bias do modelo para o conjunto de treino. Aumenta com a complexidade do modelo e diminui com o tamanho do dataset.

Underfitting representa a incapacidade do modelo para capturar os verdadeiros padrões dos dados. É causado por modelos demasiado simples ou datasets com poucas do inadequadas variáveis independentes.

2.4.1 Simple Cross Validation

Na simple cross validation o dataset é dividido aleatoriamente em conjunto de treino, usado para treinar o modelo e conjunto de teste, usado para testar o modelo.

Os resultados são altamente dependentes na partição aleatória, pelo que ao correr várias vezes vão ser gerados resultados diferentes.

2.4.2 K-Fold Cross Validation

O k-fold cross validation segue uma estratégia semelhante. escolhido um k, o dataset é dividido em k partições, usando $k-1$ para treino e 1 para teste. O teste é efetuado k vezes, uma com cada partição e o modelo é avaliado com base na média destes resultados.

2.4.3 Leave-one-out Cross Validation

É identico ao k-Fold cross validation, mas processa-se em iterações, começando com $k = 1$, até $k = N$ (número de elementos no dataset).

Capítulo 3

Processamento de Dados

3.1 Escalamento de Dados

Vários modelos requerem uma uniformização das variáveis independentes, especialmente aqueles baseados em distâncias.

3.1.1 Range Scaling

É útil quando o modelo requer uma gama específica de valores, por exemplo, $[0, 1]$ ou $[-1, 1]$. A presença de outliers pode comprimir os dados.

3.1.2 Standardization Scaling

É uma das técnicas mais populares, subtrai a mediana e divide pelo desvio padrão:

$$X_t = \frac{X - \mu}{\sigma}$$

Torna todas as variáveis diretamente comparáveis mas não lida com outliers.

3.1.3 Power Transform

É uma forma de lidar com outliers, transformando os dados numa distribuição normal. O algoritmo procura vários valores de lambda e para cada feature seleciona o que melhor representa uma distribuição normal. Os outliers são comprimidos.

3.1.4 Unit Norm Scaling

Ignora features separadas e faz com que cada instância tenha a sua norma igual a 1. A norma default é L2: a raiz da soma dos quadrados de cada valor.

Apesar de resultar em valores entre 0 e 1, não é o mesmo que range scaling. Lida com outliers, ainda que estes possam dominar algumas amostras.

3.2 Imputação

Quando existem dados em falta, há 4 formas de proceder: não fazer nada, eliminar linhas, eliminar features (colunas) ou imputar dados.

3.2.1 Univariate Imputation

Funciona para uma única variável. Cada valor em falta é preenchido com uma estatística ou assunção em comum:

- Mediana
- Média
- Moda
- Valor fixo constante

É uma abordagem simples mas que funciona bem desde que não existam muitos valores em falta.

3.2.2 K-NN Imputation

Usa as instâncias mais próximas com features preenchidas para inferir sobre os valores em falta. Pode lidar com múltiplas variáveis, mas pode ser lenta e comprometer a performance do modelo.

3.3 Feature Selection

Tem como objetivo determinar qual o conjunto de features relevantes a um dado problema. Ajuda a simplificar modelos, diminuir tempos de treino e evitar a curse of dimensionality.

3.3.1 Modelos Simples

Variância

Usa-se a variância de cada variável individualmente. A ideia é remover variáveis cuja variância esteja abaixo de um certo limite.

Covariância e correlação

É útil para identificar relações binárias. Pode-se utilizar a correlação de Pearson, que mede a correlação linear, ou a correlação de Spearman, que a rank correlation entre dois conjuntos de dados.

Informação mútua

Mede a dependência mútua entre duas variáveis. Quantifica a quantidade de informação obtida por uma variável ao absorver a outra.

3.3.2 Força Bruta

Consiste em tentar todas as combinações de features possíveis. Apenas concebível se o número de features por pequeno ou para abordagens que não requerem percorrer todos os dados (modelos lineares, naive bayes).

3.3.3 Stepwise

Consiste em proceder iterativamente, mudando os modelos base e selecionando a melhor alternativa (algoritmo greedy). Duas abordagens essenciais:

- Forward selection
  - Tentar cada feature individualmente e adicionar ao modelo a que tem melhor performance
  - Repetir até ter o número de features desejado
- Backward Selection
  - Começar com o modelo completo, removendo cada feature e verificar se o modelo tem performance melhor
  - Repetir até ter o número de features desejado

3.4 Dimensionality Reduction

Tem como objetivo transformar o dataset original numa representação com menos dimensões.

3.4.1 Principal Components Analysis

Transforma linearmente os dados para um novo sistema de coordenadas onde a maioria das variações pode ser descrita com menos dimensões que inicialmente.

- Seja X uma matrix $(n \times p)$, onde n corresponde aos número de exemplos e p o número de features ou dimensões
- A transformação é definida pelo conjunto de vetores de pesos w_j, de tamanho l, que mapeiam cada linha de $X(x_i)$ a um novo vetor de scores PC $(x_i \cdot w_j)$, que maximiza a variância.
- A decomposição PCA de X é dada por $T = XW$

PCA funciona pois a decomposição em eigenvectors ordena naturalmente os eigenvalues por ordem decrescente, apenas é necessário definir que parte da variância total se deve sacrificar.

3.5 Model Tuning

É o processo de melhorar a qualidade de um modelo. O objetivo é encontrar o conjunto de parâmetros capazes de produzir os melhores modelos (hyperparameter optimization). Pode ser um processo computacionalmente exigente,

podendo requerer grid search.

Apesar de tudo, pode não ser sempre a melhor escolha, podendo levar a overfitting.

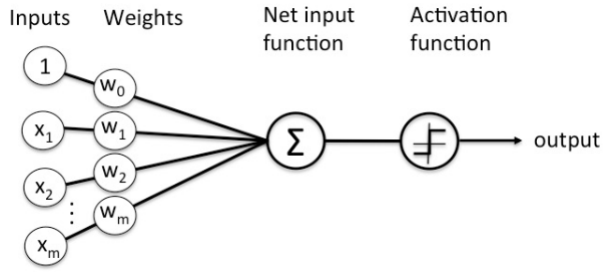
Capítulo 4

Outros Tópicos

4.1 Redes Neurais

4.1.1 Percetrão de Rosenblatt

Para encontrar o output de um neurónio tem-se em conta as somas ponderadas de todos os inputs. De seguida este resultado é passado por uma função de ativação e o output final é produzido.



A convergência é garantida se as classes forem separáveis por uma linha reta.

Aprendizagem com o Percetrão

É possível criar um learning model, sendo que aprender é a determinação do peso de cada input. O processo é o seguinte:

- Definir o learning rate (η)
- Atribuir valores aleatórios aos pesos (tipicamente $[-1, 1]$)
- Para cada amostra x:
  - Calcular o valor de output $y = g(x)$
  - Comparar ao valor esperado e calcular o erro, para definir mudanças no gradiente do peso

$$\Delta w_0 = \eta(truth^{(i)} - y^{(i)})$$

$$\Delta w_j = \eta(truth^{(i)} - y^{(i)})x_j^{(i)}$$

– Ajustar os pesos em Δw

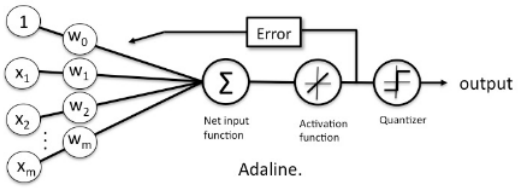
$$w_j := w_j + \Delta w_j$$

- Repetir o processo enquanto houver instâncias mal classificadas

Batch Learning com ADALINE

A ideia consiste em substituir a aprendizagem instância a instância por aprendizagem em conjuntos (batches) de dados.

Neste modelo é usada uma função linear de ativação. Os pesos são atualizados com base no algoritmo gradient descent e pode acontecer mesmo que todos os exemplos estejam classificados corretamente. Isto porque em vez de olhar para y, agora é usada uma loss function, que deve ser minimizada.



A loss function mede o erro entre a previsão e o valor real. Os pesos são atualizados como uma função entre o learning rate e a loss:

$$w \leftarrow w + \eta(o - y)x$$

Para encontrar o mínimo local de uma função diferenciável usa-se o stochastic gradient descent. A ideia é atualizar os pesos a cada sample, mas fazê-lo usando o gradient descent.

Percetrão com Sigmoide

É um caso interessante, pois corresponde exatamente á regressão logística. Apesar do processo de aprendizagem ser diferente, os resultados deverão ser iguais.

Resumo

Is the Perceptron...

- Simple?
  - YES - Only M+1 parameters, as many as the number of features. Yet the learning rate hyperparameter can complicate model fitting
- Stable?
  - Generally NO – If the data is linearly separable, then yes, it guarantees convergence. Otherwise, no. Stochastic gradient descent can even produce different results with different orderings of the data
- Fast to learn?
  - Generally YES – Both the perceptron and gradient descent are generally very fast, even for large datasets (providing a decent learning rate is used)
- Fast to make predictions?
  - YES – One multiplication and addition per independent variable for prediction.
- Updateable?
  - YES – the perceptron can easily accommodate further epochs for training with more and more data

4.1.2 Redes Neurais Artificiais

O percetrão é um modelo simples capaz de efetuar aprendizagem supervisionada, mas torna-se mais poderoso quando combinado numa rede com outros percetrões, uma rede neural artificial.

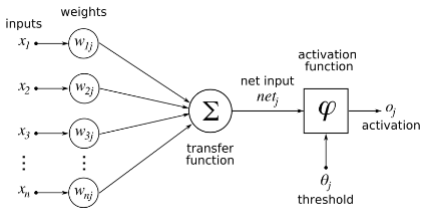
Podem ser combinados independentemente numa camada simples, onde cada output é independente e os pesos de cada não interferem. Ou, no caso de estarem combinados e dos seus pesos estarem ligados, temos uma rede naural multilayer.

A aprendizagem com multi layer perceptrons é mais difícil que percetrões simples. Existem muitos mais pesos que são interdependentes:

- Pesos do hidden layer para o output layer
- Bias do hidden layer
- Pesos do input layer para o hidden layer
- Bias do input layer

Backpropagation

É o algoritmo de aprendizagem usado em multi layer perceptrons. É semelhante ao gradient descent, mas com a regra derivative chain.



O processo de treino consiste em:

- Iniciar os pesos com valores aleatórios $[-1, 1]$
- Enviar um batch de dados pela rede
- Atualizar os pesos através de backpropagation
- Repetir estes passos um certo número de épocas ou até os pesos convergirem e deixarem de mudar

Resumo

Is the MLP...

- Simple?
  - NO – typically MLPs are really complex. Some models have millions of parameters
- Stable?
  - Generally NO – Random initialization and many local minima mean that a network may produce a very different model for the same data
- Fast to learn?
  - Typically NO – They are very demanding on CPU but the fitting process can be parallelized even on GPUs
- Fast to make predictions?
  - Depends on the size of the network – Bigger models are slower, but GPUs can make the process very fast
- Updateable?
  - YES – similar to the perceptron it can easily accommodate further epochs for training with more and more data

4.2 Aprendizagem Não Supervisionada

Trata-se de uma forma de observar a estrutura dos dados, de forma a que padrões possam emergir.

4.2.1 Clustering

É um problema $\mathbb{NP}$- difícil. Consiste em agrupar elementos de forma a minimizar as distâncias entre elementos e maximizar a separação entre grupos.

Os resultados dependem dos dados e do algoritmo usado.

Espaços métricos e vetoriais

Um espaço vetorial é um espaço de features onde cada instância pode ser descrita como um conjunto de features medidas. Por outro lado, num espaço métrico as instâncias apenas podem ser descritas relativamente a outras instâncias, através de uma medida de distância ou função de semelhança.

Os métodos de clustering dependem da natureza dos dados ser, ou não, vetorial

K-Means Partitioning

É usado em espaços vetoriais. Dado um dataset D, com n objetos e k, o número de clusters a formar, um algoritmo de partitioning agrupa os objetos em k grupos (clusters).

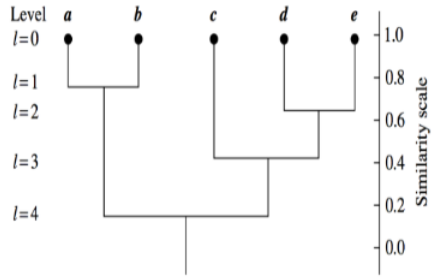
Para formar os clusters, o algoritmo tenta otimizar um critério de partitioning objetivo. Processa-se segundo os seguintes passos:

- Arbitrariamente escolhem-se k objetos como centros iniciais de clusters.
- Cada objeto é associado ao cluster cujo centro esteja mais próximo
- Na próxima iteração os centros são atualizados, i.e., a mediana dos valores de cada cluster é recalculada e os seus objetos também
- Este processo repete-se um certo número de vezes ou até deixar de haver mudanças

Métodos Hierárquicos

Estes métodos são usados em espaços métricos. A organização formada difere, sendo organizada numa hierarquia, ou árvore de clusters. Os algoritmos hierárquicos podem ser:

- Aglomerativos: começam com clusters apenas com um objeto, que são fundidos para formarem clusters maiores
- Divisivos: começam com um único cluster com todos os objetos, que é iterativamente dividido em clusters menores



Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.

Distância Entre Clusters

Existem vários métodos para determinar a distância entre um par de clusters. Segundo o critério de linkage, dados clusters C_i e C_j a distância entre eles, $\delta(C_i, C_j)$ pode ser definida por:

- Single linkage: a distância mínima entre um ponto em C_i e C_j

$$\delta(C_i, C_j) = min\{\delta(x, y) | x \in C_i, y \in C_j\}$$

- Complete linkage: a distância máxima entre um ponto em C_i e C_j

$$\delta(C_i, C_j) = max\{\delta(x, y) | x \in C_i, y \in C_j\}$$

- Average linkage: a distância média entre pares de pontos em C_i e C_j

$$\delta(C_i, C_j) = \frac{\sum_{x \in C_i} \sum_{y \in C_j} \delta(x, y)}{n_i \cdot n_j}$$

- Mean distance: a distância entre as medianas ou centroides dos dois clusters

$$\delta(C_i, C_j) = \delta(\mu_i, \mu_j)$$

  Onde $\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x$.

Avaliar Clustering

Existem várias formas de avaliar um cluster:

- Avaliação interna envolve a atribuição de uma única pontuação ao cluster. São priorizados clusters com alta semelhança dentro do cluster e baixa semelhança entre clusters
- Avaliação externa compara o cluster a uma "ground truth". Medem o quão perto o clustering está das classes conhecidas