Módulos do Python para Machine Learning

Scikit-learn é o módulo mais usado do python para executar tarefas de machine learning. No notebook temos um exemplo de seu uso para o caso de uma regressão linear, multilinear e não linear (polinomial).

Exemplos note book

Teoria de aprendizagem estatística

Existem diferenças entre o que podemos chamar de estatística clássica e os processos de machine learning. Para exemplificar, vamos considerar o problema das bolas numa urna. Existem dois tipos de bola (azul e vermelha)

O problema mais típico dentro da estatística clássica seria: dada uma amostra de bolas da urna, o que podemos dizer sobre a composição da urna. Do ponto de vista de M.L., queremos encontrar uma função f que defina a cor da bola a partir da observação da amostra (training set). De modo mais específico, vamos supor que exista essa função f e vamos procurar por uma estimativa $\hat{f}$. Essa $\hat{f}$ poderá ser aplicada a um conjunto nunca observado de bolas (generalização) e espera-se que ela possa estimar a cor dessas bolas.

Matematicamente, vamos definir uma função

$$f: X \mapsto Y$$

e o conjunto de treinamento (training set)

$$\{(x, y)\}$$

Vale notar que f não é conhecida, ou seja, temos apenas o input e output da função. O conjunto das hipóteses $\mathcal{H}$ vai representar todas as possíveis representações para essa função, a partir do qual vamos estimar $\hat{f}$. Um problema de machine learning consiste em obter $\hat{f}$ a partir do training set $\{(x, y)\}$. Suponha, por exemplo, que o domínio de f consista de todas os vetores de 3-bits do tipo

$$X = \{000, 001, \cdots, 111\}$$

e que f seja uma função que verifica se o número de zeros em x é maior que o número de uns, ou seja,

$$Y = \{0, 1\} = \{\text{"não"}, \text{"sim"}\}$$

Nesse caso, o conjunto das hipóteses compreende a todas as possível funções de X. Esse conjunto tem 2^{16} elementos pois temos 2^4 possíveis valore em X e e existem duas possibilidades de valores de Y para cada um deles, logo: $(2^2)^4 = 2^{16} = 2^{(2^4)} = 65536$

Supondo que tenhamos um conjunto com 8 observações de f, nosso objetivo será o de encontrar $\hat{f}$ que minimize o erro $E_{in}(\hat{f})$ no training set. Novamente, sabemos que existem 2^8 elementos do conjunto de

do conjunto de hipóteses que consistem de possibilidades para o conjunto de treinamento. A questão é como escolher entre esse elementos. É preciso assumir que o training set representa uma amostragem aleatória que seja representativa da população. Costuma-se chamar essa suposição de uma estrutura estável de probabilidade para os dados in-sample e out-of-sample. Trata-se de uma hipótese fundamental que implica que o bom acordo dos métodos de M.L. está condicionada a um data environment não perturbado, a partir do qual o conjunto de treinamento foi extraído. Além disso, quanto maior o training set menor de ser o erro associado a exposição de $\hat{f}$ a um novo conjunto de dados (de modo geral).

Exemplo notebook

De modo geral, estamos interessados em saber como nosso modelo ($\hat{f}$) vai se comportar quando exposto a novos dados, ou seja, dado que minimizamos $E_{in}(\hat{f})$ como fica o $E_{out}(\hat{f})$? Isso é o que chama generalização e pode ser expresso via

$$\mathbb{P}\left(\left| E_{out}(\hat{f}) - E_{in}(\hat{f}) \right| > \epsilon \right) < \delta$$

para um dado valor de ϵ e δ. Assim, a probabilidade da diferença absoluta entre os erros superar ϵ é inferior a δ. Note que essa expressão não

se refere ao fato de $E_{in}(f)$ ser pequeno ou grande, mas apenas ao fato de não ser muito diferente de E_{out}. Existe um resultado matemático que estabelece que

$$E_{out}(\hat{f}) \leq E_{in}(\hat{f}) + \sqrt{\frac{8}{n} \ln\left(\frac{(2n)^{d_{vc}} + 1}{\delta/4}\right)}$$

com probabilidade $1-\delta$, mostrando que E_{out} não pode ser pior que E_{in} adicionada a um termo de penalidade. Nessa expressão, n representa o tamanho do conjunto de dados $F = \{x_1, x_2, \dots, x_n\}$ e d_{vc} é a chamada dimensão de Vapnik-Chervonenkis. Essa quantidade é uma medida da complexidade do espaço de funções que pode ser "aprendido" pelo procedimento de machine learning. Pode ser difinida como a cardinalidade do maior conjunto de pontos que o algoritmo de aprendizagem pode "quebrar" (shatter) os dados. Para uma difinição mais precisa, considere $\tilde{A}$ como uma classe de conjuntos e deixe

$$N_{\tilde{A}}(F) = \#\{F \cap A : A \in \tilde{A}\}$$

representar o número de conjuntos de F que podem ser extraídos pelos conjuntos de $\tilde{A}$. Seja ainda

$$s(\tilde{A}, n) = \max_{F \in F_n} N_{\tilde{A}}(F)$$

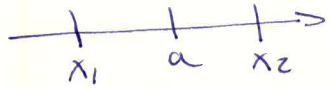
com F_n todos os conjunto de F de tamanho n.

A dimensão VC é definida como o maior K tal que

$$S(\tilde{A}, n) = 2^K$$

a menos que $S(\tilde{A}, n) = 2^n$, quando $d_{vc} \to \infty$. Vejamos um exemplo. Suponha $F = \{x_1, x_2\}$ e $\tilde{A} = \{(x \leq a)\}$, ou seja, $\tilde{A}$ representa todos intervalos fechados a direita por um parâmetro a. Nesse caso, todos os conjunto que $\tilde{A}$ pode extrair de F são:

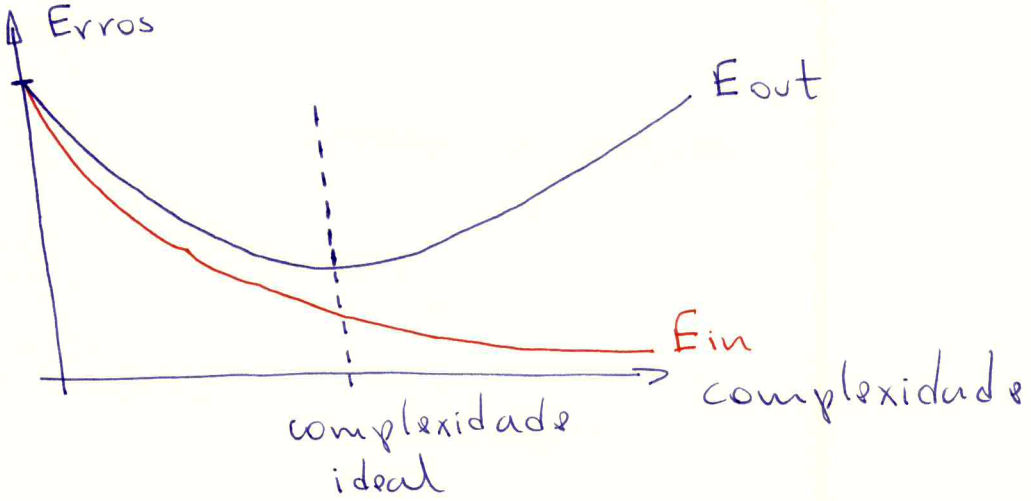
$$\{\varnothing, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$$

Para o primeiro caso, basta escolher $a < x_1 \wedge a < x_2$.
Para o segundo caso, assumindo $x_1 < x_2$, podemos escolher a tal que $x_1 < a < x_2$, ou seja, 

. Para último caso, basta tomar $a > x_2$. Note que não é possível extrair o terceiro conjunto $\{x_2\}$ usando $\tilde{A}$. Assim, $\tilde{A}$ pode shatter qualquer conjunto com $n = 1$, mas não pode fazer o mesmo com $n = 2$. Logo, d_{vc} de $\tilde{A}$ é igual a 1. ~~Note que~~

O resultado da relação entre Eout e Ein, portanto, afirma que a limitação de Eout torna-se menos restritiva a medida que a complexidade do modelo aumenta. Modelos muito complexos podem levar a um erro muito pequeno no training set, mas a erros muito grandes em outros dados.

Existe, portanto, um trade-off entre a complexidade do modelo e o erro, conduzindo a uma complexidade ideal para o modelo, conforme ilustra a figura abaixo.

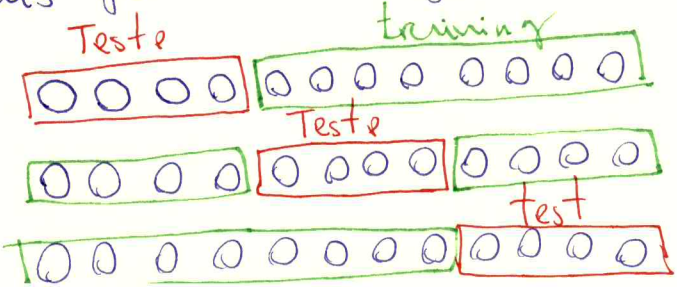


Exemplo notebook

Validação cruzada (Cross-validation)

Além da estimativa dos erros, muitas vezes é necessário encontrar desvios desse erros. No caso no exemplo anterior, simplesmente geramos mais dados. Porém, essa possibilidade de não existe quando estamos lidando com dados reais, ou seja, não temos o processo gerador dos dados. Uma possibilidade é usar o procedimento de validação cruzada. Um dos mais simples é o chamado K-fold validation, que basicamente consiste em dividir o dado em K partes, usar cada uma delas para teste, e as demais para ~~validação~~ treinar. Num diagrama, teríamos

ou validação



Em cada passo, podemos estimar o erro no training e validation e, posteriormente, podemos tomar a media e o desvio padrão sobre todos os K passos.

Exemplo notebook

Bias e Variance

Até agora temos analisado os erros em termos das in-samples e out-samples. Naturalmente essas quantidades dependem do dado usado para treinar o modelo. Para obter uma estimativa desses erros que sejam independentes do training set, devemos estender essas quantidades para todos os possíveis training sets. Por exemplo, se nosso estimador é obtido a partir do training set D, então devemos denotá-lo por $\hat{f}_D$. Sendo assim, teremos $E_{out}(\hat{f}_D)$. Para eliminar a dependência em D, devemos calcular a media sobre todos os possíveis training sets

$$E_D\left[E_{out}(\hat{f}_D)\right] = \text{bias} + \text{var}$$

com

$$\text{bias} = \left[\bar{\hat{f}}(x) - f(x)\right]^2$$

$$\text{var} = E_D\left(\hat{f}_D(x) - \bar{\hat{f}}(x)\right)^2$$

sendo $\bar{\hat{f}}$ a média dos estimadores sobre todos os D.

O bias indica que mesmo o método sendo exposto a todos os possíveis training sets, ele ainda pode diferir por algum valor da função alvo. A var mostra como o estimador varia com o training set

Vejamos um exemplo. Suponha que nosso conjunto de hipóteses seja todas a regressões lineares do tipo

$$h(x) = ax$$

sendo o training set dado por

$$\{(x_i, \text{sen}(\pi x_i))\}_{i=1,2}$$

com $x_i \rightsquigarrow \text{Unif}[-1,1]$. Nesse caso, podemos encontrar o coeficiente a via

$$a = \frac{\bar{x}^T \bar{y}}{\bar{x}^T \bar{x}}$$

Sendo assim,

$$\hat{\bar{f}}(\bar{x}) = \bar{a}x$$

com $\bar{a}$ o valor médio de a sobre todos os possíveis trainings sets, o qual fica dado por

$$\bar{a} = E\left[\frac{x_1 \text{sen}(\pi x_1) + x_2 \text{sen}(\pi x_2)}{x_1^2 + x_2^2}\right]$$

onde usamos

$$\bar{x} = [x_1, x_2]$$

$$\bar{y} = [\text{sen}(\pi x_1), \text{sen}(\pi x_2)]$$

Calcular esse valor esperado analiticamente é muito complicado, mas usando uma simulação, temos que

$$\bar{a} \simeq 1{,}43 \qquad \text{(Exemplo notebook)}$$

Sendo assim, ficamos com

$$\text{var}(x) = \mathbb{E}\left[(a-\bar{a})x\right]^2 = x^2\,\mathbb{E}(a-\bar{a})^2$$

$$\approx 0{,}71\,x^2$$

e

$$\text{bias}(x) = \left[\text{sen}(\pi x) - \bar{a}x\right]^2$$

Exemplo notebook

Learning noise

Até agora não consideramos o efeito de ruídos em nossas análises de machine learning. Para tratar desse problema, considere a seguinte função alvo

$$y(\bar{x}) = \bar{w}_0^T \bar{x} + \eta$$

sendo $\eta \rightsquigarrow \text{Normal}(0, \sigma^2)$, $\bar{w}_0$ e $\bar{x} \in \mathbb{R}^d$. Suponha que tenhamos n medidas de y, isto é,

$$\{(\bar{x}_i, y_i)\}_{i=1,2,\dots,n}$$

é o training set. colocando numa notação vetorial, temos

$$\bar{y} = \bar{X}\bar{w}_0 + \bar{n}$$

sendo $\bar{y} \in \mathbb{R}^n$, $\bar{w}_0 \in \mathbb{R}^d$ e $\bar{X} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n]$. Nosso conjunto de hipóteses são os modelos lineares

$$h(\bar{w}, \bar{x}) = \bar{w}^T \bar{x}$$

Nosso objetivo é "aprender" $\bar{w}$ a partir do training set.

O problema nesse caso é que o ruído η pode fazer com que valores idênticos de $\bar{x}_i$ sejam associados a valores diferentes de y_i.

Ainda assim, para cada training set, sabemos que

$$\bar{w} = (\bar{X}^T \bar{X})^{-1} \bar{y}$$

e que erro in-sample fica

$$E_{in} = \|\bar{y}\|^2 - \|\bar{X}\bar{w}\|^2$$

com $\bar{h} = \bar{X}\bar{w}$ nossa melhor hipótese para o training set em questão. Podemos agora tomar a média sobre a distribuição de η, ou seja,

$$E[\|y\|^2] \quad \text{e} \quad E[\|\bar{X}\bar{w}\|^2]$$

resultando em

$$E[E_{in}] = \sigma^2\left(1 - \frac{d}{n}\right)$$

Aqui podemos observar a relação entre a intesidade do ruído σ e a complexidade do modelo (d), além do tamanho do training set. Em particular, a razão d/n representa o trade-off entre a complexidade do modelo e o tamanho do training set. Notamos, ainda, que para $n \to \infty$ $E[E_{in}] \to \sigma^2$, ou seja, o método não pode aprender o ruído, de modo que o erro fica limitado pelo erro do ruído.

De modo similar, podemos encontrar que

$$E(E_{out}) = \sigma^2\left(1 + \frac{d}{n}\right)$$

Notamos novamente que $n \to \infty$ conduz a $E(E_{out}) = \sigma^2$. Por outro lado, se $E(E_{in}) = 0$, então $\frac{d}{n} = 1$ e

$$E(E_{out}) = 2\sigma^2$$

ou seja, aprendendo por completo na in-sample leva a um erro maior na out-sample. Vale notar ainda que

$$E(E_{out}) - E(E_{in}) = \frac{2\sigma^2 d}{n}$$

ou seja, para n finito vai haver sempre uma diferença entre o out-sample e in-sample errors.

Exemplo notebook