APRENDIZAGEM DE MÁQUINA

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/machinelearning

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

MODELOS LINEARES DE REGRESSÃO

Machine Learning SUPERVISIONADA: estimar modelos que, embora simplificações da realidade, apresentem a melhor aderência possível entre os valores **reais (observados)** e os valores preditos

Esta é a equação geral da Regressão Linear Múltipla, com n features. Se considerarmos apenas uma, o objetivo é encontrar uma equação que apresente a relação entre uma variável dependente (target) e uma variável explicativa (feature)

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n, \ \text{RLM}$$

$$\hat{y} = \theta_0 + \theta_1 x_1, \ \text{RLS, onde } \theta_0 : \ \text{viés, intercepto, coef. lineear}$$

$$x_i : \ \text{é a i-ésima feature e } \theta_j \text{ é o j-ésimo parâmetro do modelo}$$

Cada parâmetro do modelo (com exceção do intercepto) na verdade reflete o PESO de determinada feature, então vamos adotar uma notação mais comum em aprendizado de máquina, o peso como sendo a letra W

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{W}.\mathbf{x} = \mathbf{W}^{\mathbf{T}}\mathbf{x}$$

MODELOS LINEARES DE REGRESSÃO

Notação

Medidas de desempenho comum em Regressão

Scikit-Learn.metrics – root_mean_squared_error (scikit-learn >= 1.4)

$$\text{Raiz do Erro Médio Quadrático: RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{W}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} - \mathbf{y}^{(\mathbf{i})}\right)^2}$$

Peso maior para grandes erros

$$\text{Erro Médio Absoluto: MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|\mathbf{W}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} - \mathbf{y}^{(\mathbf{i})}\right|$$

Indicado para bases com outliers...

m: número de instâncias no conjunto de dados

$\mathbf{x}^{(\mathbf{i})}$ é um vetor de todos os valores das features (excluindo o rótulo) da $\mathbf{i}-\mathbf{esima}$ instância

$y^{(i)}$ é seu rótulo, valor desejado da saída para aquela instância

$$\mathbf{x}^{(\mathbf{1})} = \begin{pmatrix} -118.20 \\ 33.91 \\ 1416 \\ 38372 \end{pmatrix} \quad y^{(1)} = 156400 \qquad \mathbf{X} = \begin{pmatrix} (\mathbf{x}^{(\mathbf{1})})^{\mathbf{T}} \\ (\mathbf{x}^{(\mathbf{2})})^{\mathbf{T}} \\ \vdots \\ (\mathbf{x}^{(\mathbf{m})})^{\mathbf{T}} \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1416 & 38372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Exemplo: longitude, latitude, n. de habitantes, renda média anual com saída valor médio da moradia

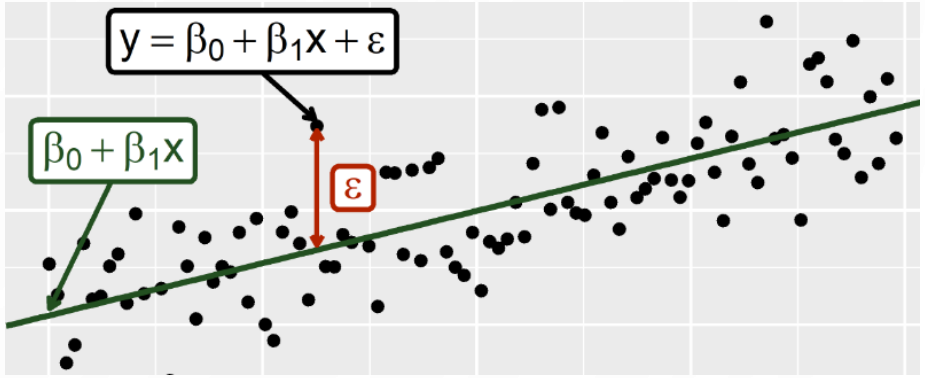
MODELOS LINEARES DE REGRESSÃO

A obtenção dos pesos do modelo (e do bias) consiste em resolver um problema de otimização de minimização da função de perda do erro quadrático médio (mean squared error – MSE) entre os valores observados e os valores preditos com a restrição de que este erro seja nulo

Ou seja, encontrar o vetor de pesos tal que $\min \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{W}^{\mathbf{T}} \mathbf{x}^{(\mathbf{i})} - \mathbf{y}^{(\mathbf{i})} \right)^2$, s.a. $\epsilon = 0$ onde $y = w_0 + w_1 x_1 + \epsilon$

Uma solução é a equação normal $\hat{\mathbf{W}} = \left(\mathbf{X}^{\mathbf{T}} \mathbf{X} \right)^{-1} \mathbf{X} \mathbf{y}$

Mas há uma inversa de produto de matrizes a resolver (n+1) x (n+1). A classe LinearRegression do scikit-learn é baseada na função scipy.linalg.lstsq() – mínimos quadrados, que usa uma versão mais efetiva com decomposição de valores singulares (SVD).

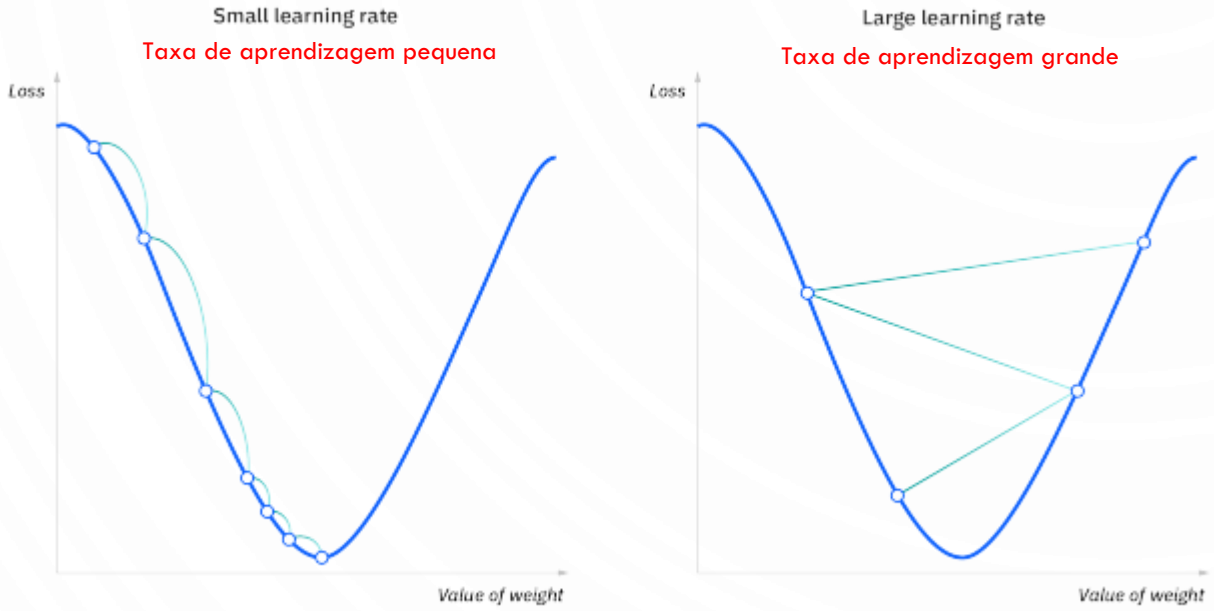


Complexidade eq. normal: $O(n^{2.4})$ a $O(n^3)$
Complexidade SVD scikit-learn: $O(n^2)$

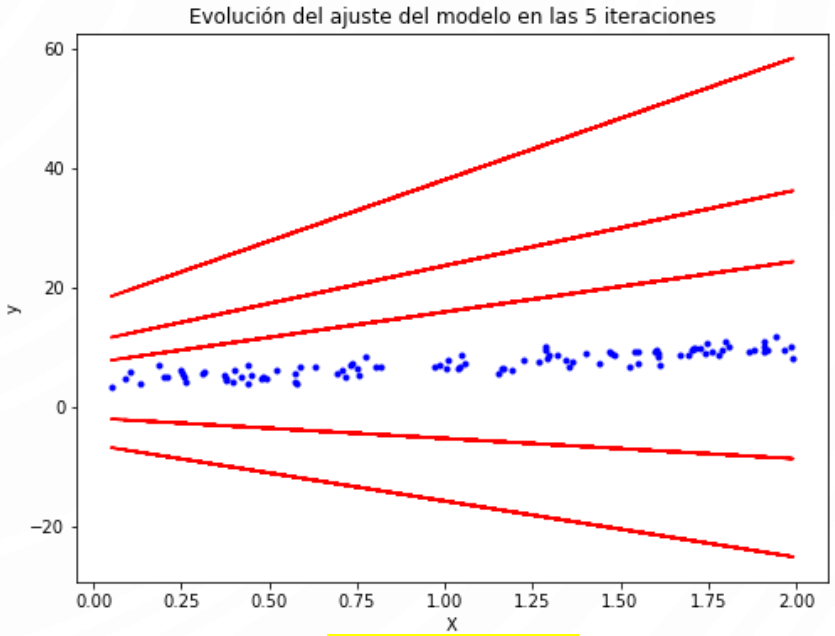
São contudo eficientes para lidar com grandes conjuntos de treinamento: $O(m)$
Para predição são lineares para número de instâncias e de características

GRADIENTE DESCENDENTE - GD

O gradiente descendente é um algoritmo de otimização que costuma ser usado para treinar modelos de aprendizado de máquina e redes neurais. Ele treina modelos de aprendizado de máquina minimizando os erros entre os resultados previstos e os reais, ou seja minimizando a função de perda associada.



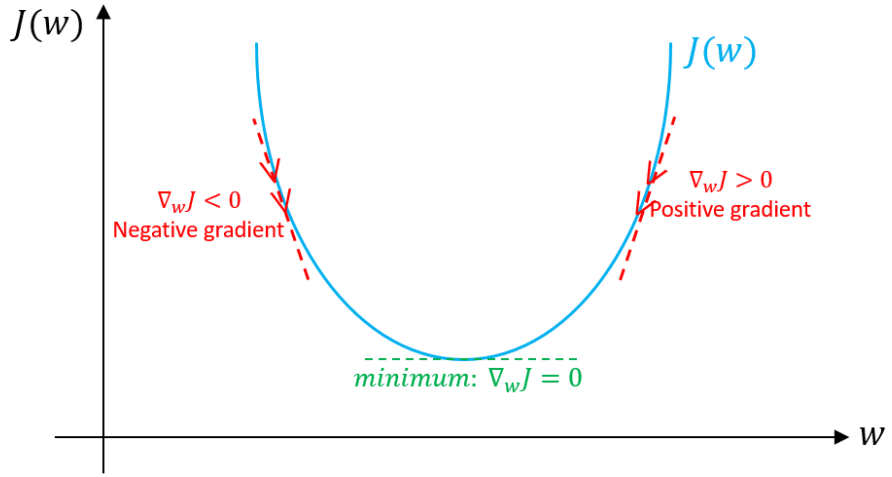
Fonte: IBM

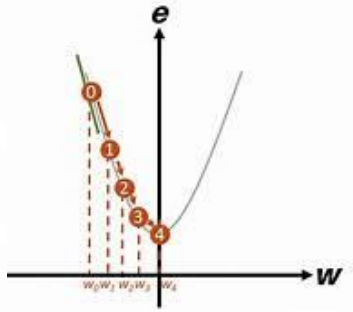


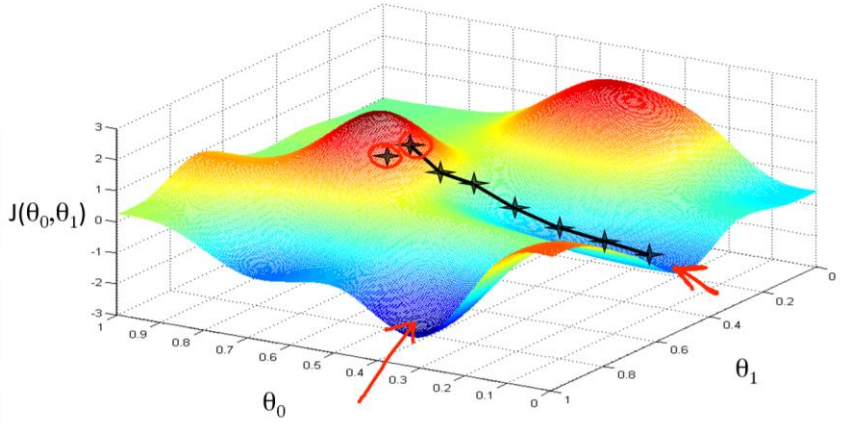
Aprendizado

Parâmetros do modelo inicializados aleatoriamente. A cada passo calcula o gradiente (ou seja, a derivada parcial da função de perda em relação ao vetor de pesos W e segue em direção ao gradiente descendente, até o zero (convergir para o mínimo). O tamanho da etapa de aprendizado é proporcional à inclinação da função de perda, logo as etapas ficam gradualmente menores à medida que se os pesos se aproximam do mínimo

GRADIENTE DESCENDENTE EM LOTE (BATCH) - GD







É necessário calcular o gradiente da função de perda em relação à cada peso do modelo, ou seja, quanto a função mudará caso se modifique um pouco o peso. A cada etapa TODO o conjunto (lote de dados) é usado, sendo lento para conjunto de treinamento muito grandes. Contudo, escalona bem o número de características, sendo muito mais rápido do que a equação normal ou SVD.

$$\frac{\partial}{\partial w_j} MSE(\mathbf{W}) = \frac{\mathbf{2}}{\mathbf{m}} \sum_{\mathbf{i=1}}^{\mathbf{m}} \left(\mathbf{W^T x^{(i)}} - \mathbf{y^{(i)}} \right) \mathbf{x_j^{(i)}}$$

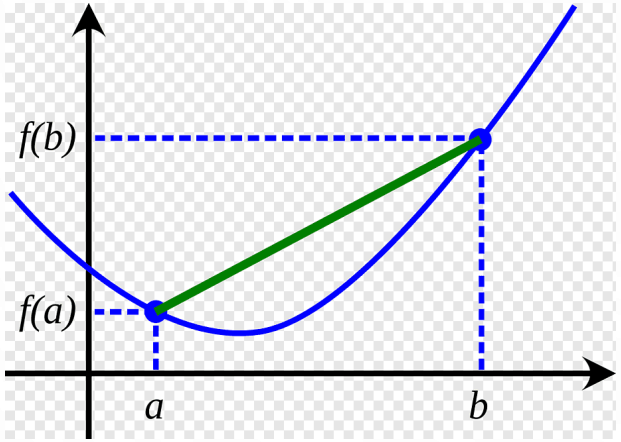
$$\nabla_w MSE(\mathbf{W}) = \begin{pmatrix} \frac{\partial}{\partial w_0} MSE(\mathbf{W}) \\ \frac{\partial}{\partial w_1} MSE(\mathbf{W}) \\ \vdots \\ \frac{\partial}{\partial w_n} MSE(\mathbf{W}) \end{pmatrix} = \frac{\mathbf{2}}{\mathbf{m}} \mathbf{X^T} (\mathbf{XW} - \mathbf{y})$$

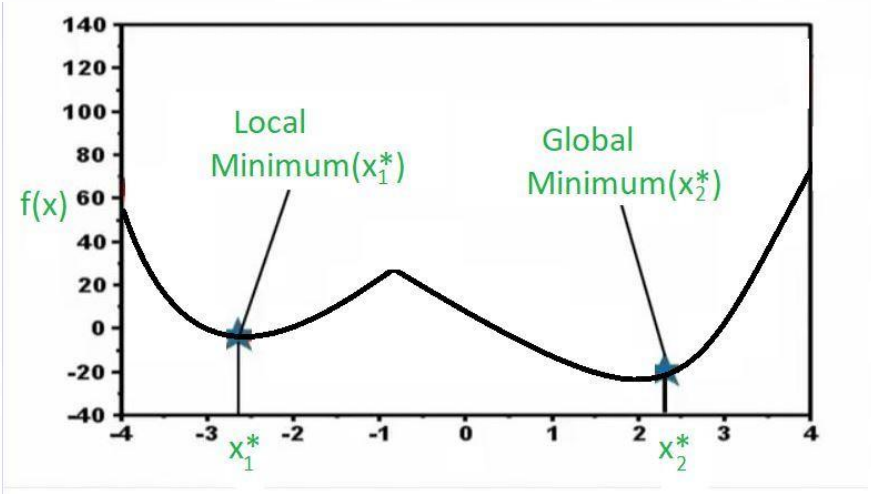
GRADIENTE DESCENDENTE EM LOTE (BATCH) - GD

Expressão de aprendizado – etapa do GD, supondo iterador k:

$$\mathbf{W}(\mathbf{k}+\mathbf{1}) = \mathbf{W}(\mathbf{k}) - \eta \nabla_{\mathbf{w}} \mathbf{MSE}(\mathbf{W}), \text{ onde } \eta\text{: taxa de aprendizagem}$$

Taxa de aprendizagem baixa: muitas iterações para convergir, lento
Taxa de aprendizagem alta: pode atravessar o vale e ir para o outro lado em lugar ainda mais alto do que onde estava, divergindo
Com funções convexas (como MSE) se pressupõe que não há mínimos locais, apenas um mínimo global e não muda abruptamente. A BUSCA é feita no espaço de pesos do modelo. Quanto mais pesos, mas complexa a busca.
Contudo é muito importante que as features do conjunto de treinamento tenham a mesma escala!





Dados dois pontos, o segmento de reta que os une nunca cruza a curva

GRADIENTE DESCENDENTE ESTOCÁSTICO (SGD)

Seleciona uma instância aleatória no conjunto de treinamento a cada etapa e calcula os gradientes com base apenas nesta instância única

Mais rápido, menos dados a cada iteração

Treinamento em grandes conjuntos de dados, pois só uma precisa estar na memória a cada iteração

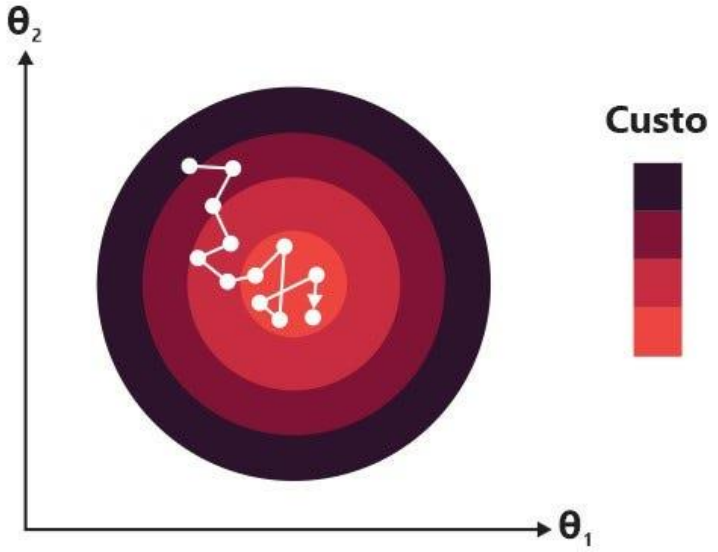
Contudo, é aleatório, menos regular que o em batch. Não desce suave, mas fica oscilando, pode gerar solução subótima

Para funções de perda irregulares (diferente da MSE) pode ajudar a fugir de mínimos locais e chegar ao global

Mas pode nunca se estabilizar no mínimo, e existem soluções de reduzir gradualmente a taxa de aprendizado
Implementar CRONOGRAMA DE APRENDIZADO

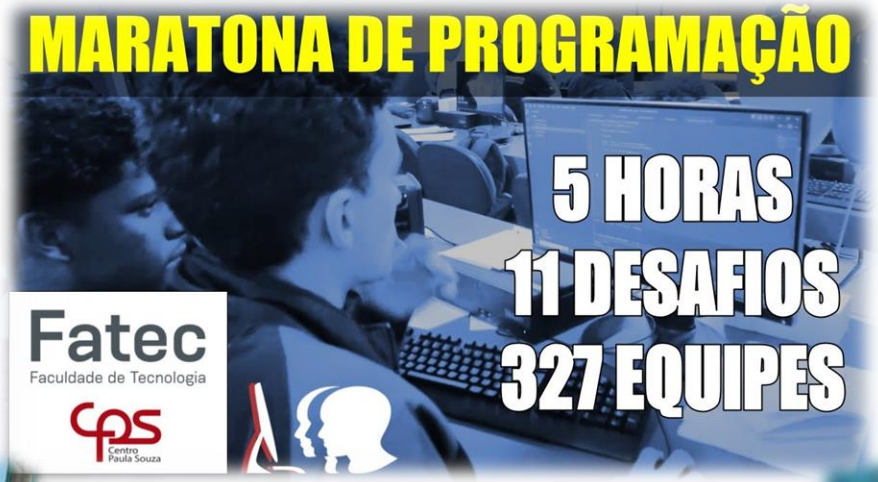
No Scikit-Learn, SGDRegressor

Embaralhar o conjunto de treinamento junto com os rótulos, e as instâncias precisam ser independentes



GRADIENTE DESCENDENTE ESTOCÁSTICO (SGD)

ÉPOCA DE TREINAMENTO: processo de FORWARD PASS (calcular as saídas) e ao processo de BACKWARD PASS (atualizar os pesos) em todo o conjunto de treinamento



Bob Burnquist - Skatista

Final Feminina – Carabina – Tiro esportivo

GRADIENTE DESCENDENTE MINI-BATCH

Calcula os gradientes com base em pequenos conjuntos aleatórios de instâncias

Permite obter ganho de desempenho com a otimização de hardware nas operações da matriz, especialmente se usar GPU

O progresso do algoritmo é menos irregular que o GD estocástico, principalmente com grandes mini-batches

Pode ser mais difícil escapar de mínimos locais

- O caminho do batch para no mínimo, SGD e mini-batch prosseguem, perto do mínimo.
- GD batch leva muito tempo para cada época, os outros também alcançariam com um bom cronograma de aprendizado

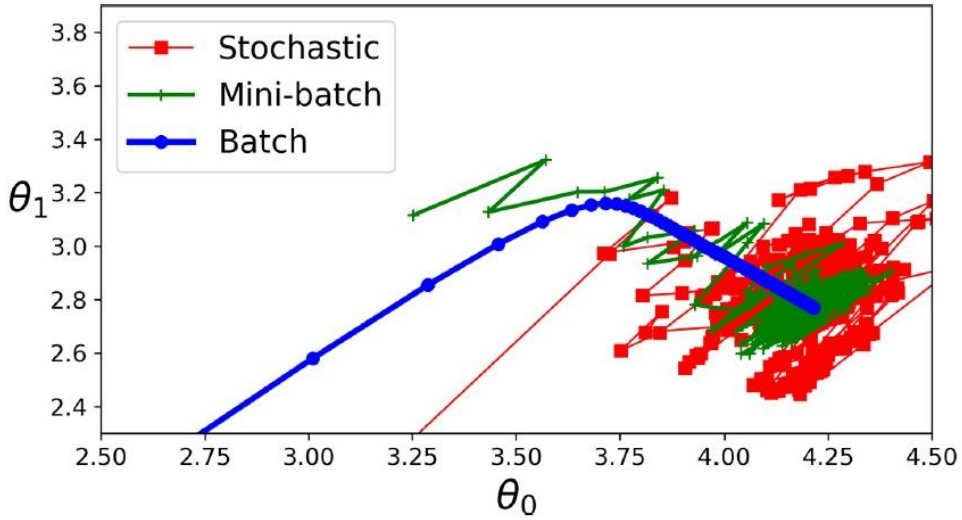


Figure 4-11. Gradient Descent paths in parameter space

(Géron, 2019)