Weekly Oxford Worldwide

DEPARTMENT FOR CONTINUING EDUCATION

UNIVERSITY OF OXFORD

PYTHON PROGRAMMING FOR DATA SCIENCE – PART 2

MASSIMILIANO IZZO & NICHOLAS DAY

LECTURE 3

SUPERVISED LEARNING: REGRESSION WITH SCIKIT-LEARN

Data preparation

- Feature engineering, where appropriate:
  - Discretize continuous features.
  - Decompose features (e.g., categorical, date/time, etc.).
  - Add promising transformations of features
  - Aggregate features into promising new features.
- Feature scaling:
  - Standardize or normalize features.
- Dimensionality Reduction (optional)
  - Principal Component Analysis (PCA)

Regression Performance Measure

- $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- $RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
- $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
- $MedAE = median(|y_1 - \hat{y_1}|, |y_2 - \hat{y_2}|, .., |y_N - \hat{y_N}|)$
- $R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y_i})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$

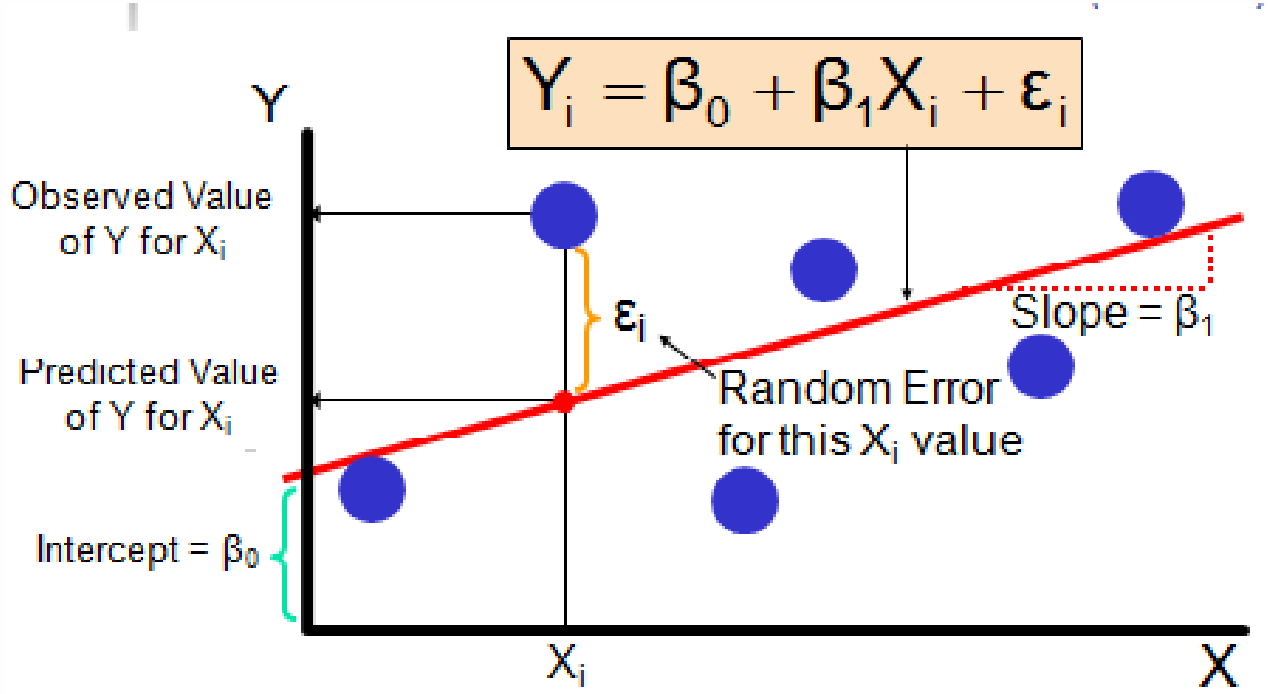
https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

Choose some Algorithms

- Linear Regression: Ordinary Least Squares
  - Closed form solution (Normal Equation)
  - Gradient Descent
- Polynomial Regression
- Regularized Models
  - Ridge Regression
  - Lasso Regression
- Decision Trees Regression
- Something else (Support Vector Machines, Neural Networks…)
- Ensemble Models, Random Forests

Linear Regression

- Find the best linear model that fits our data
- This means finding two parameters: slope (β1) and intercept (β0)



Once trained, we can use the model to make predictions => machine learning!!

Close form solution: Normal Equation

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$$

Find the value of $\boldsymbol{\beta}$ that minimizes the squared sum of the estimation errors $\boldsymbol{\epsilon}$

$$\underline{\boldsymbol{X}}^{\boldsymbol{T}}\,\underline{\boldsymbol{X}}\,\hat{\boldsymbol{\beta}} = \underline{\boldsymbol{X}}^{\boldsymbol{T}}\,\underline{\boldsymbol{y}}$$

The issue here is the computational complexity of the explicit solution, especially the complexity of computing with respect to the number of features $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ => $O(n^{2.4}) \div O(n^3)$

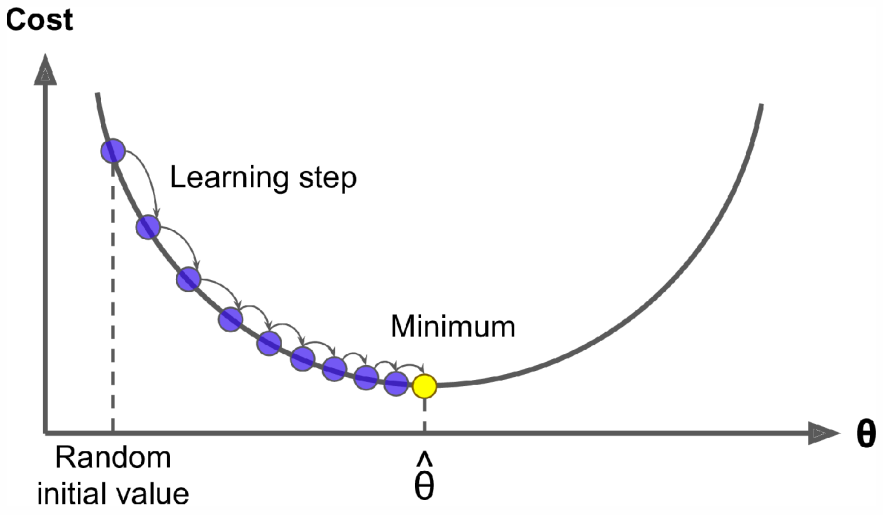
$$\hat{\boldsymbol{\beta}} = \left(\underline{\boldsymbol{X}}^{\boldsymbol{T}}\,\underline{\boldsymbol{X}}\right)^{-1}\underline{\boldsymbol{X}}^{\boldsymbol{T}}\,\underline{\boldsymbol{y}}$$

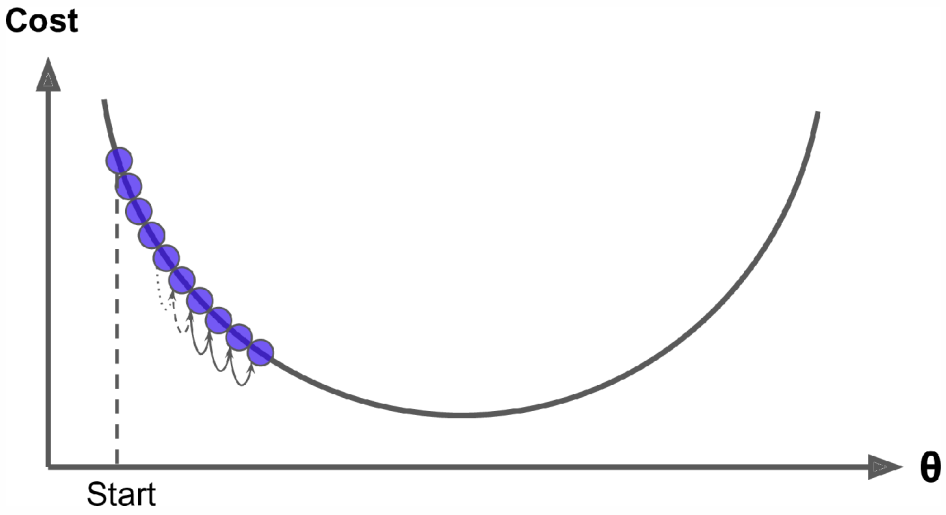
A different approach would be to use an optimisation algorithm to find the optimal solution

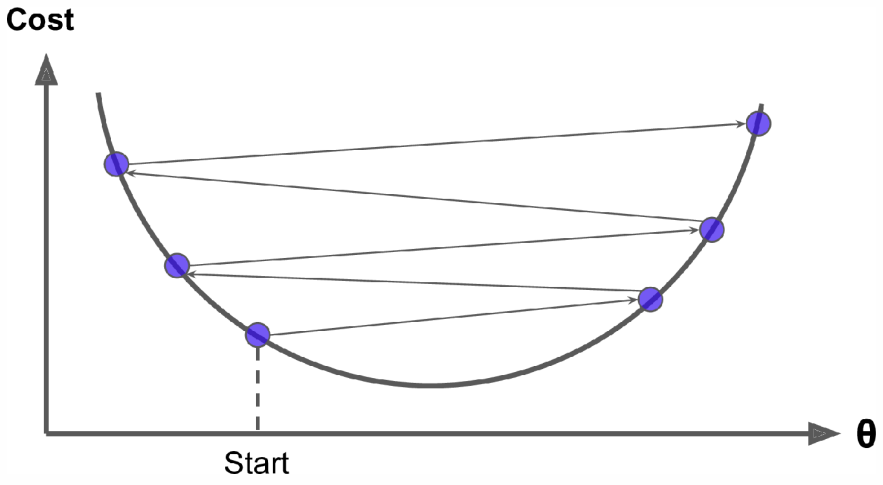
Gradient Descent

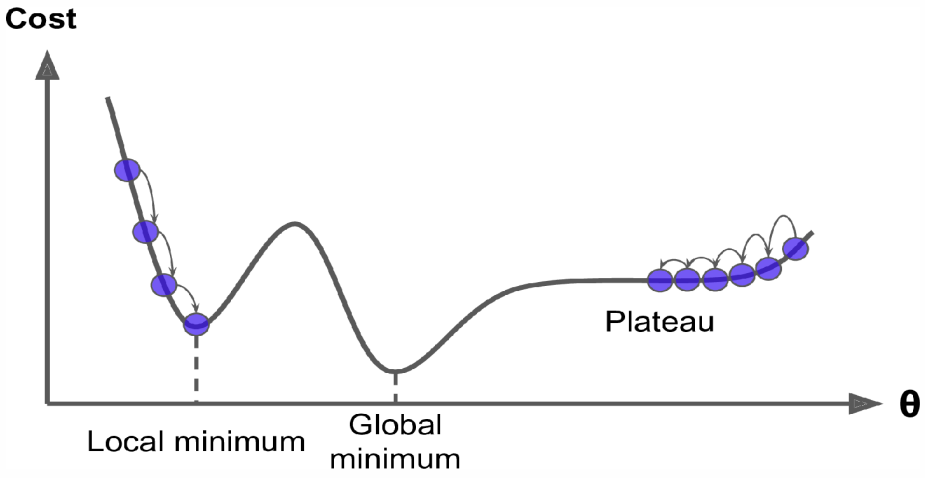
- Tweak the weights β iteratively in order to minimize a cost function.
- Measure the local gradient of the error function with respect to the weights β, and tweak β in the direction of descending gradient.
- Once the gradient equals zero, you have reached a minimum.

Gradient Descent









Batch Gradient Descent

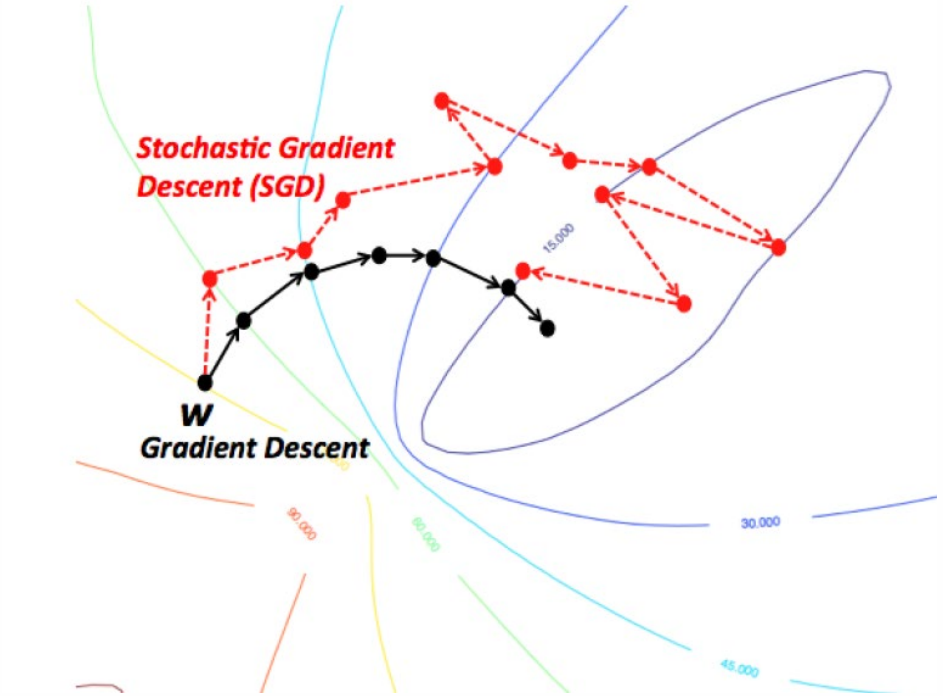
Let's consider the Mean Squared Error as our performance metric. The Gradient Formula for MSE is:

$$\nabla_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} MSE(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} MSE(\boldsymbol{\theta}) \\ \dots \\ \dots \\ \frac{\partial}{\partial \theta_n} MSE(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \boldsymbol{X}^T (\boldsymbol{X\theta} - y)$$

Hence, the gradient descent at each iteration is: $\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n - \eta \, \nabla_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta})$

Stochastic Gradient Descent

- Note that this formula involves calculation over the full training set ***X*** at each Gradient Descent step. This is why the algorithm is called *Batch Gradient Descent*.
- The solution is to use *Stochastic Gradient Descent (SGD)*: pick a random instance in the training set at every step and computes the gradients based only on that single instance
- SGD: faster algorithm but slower to converge



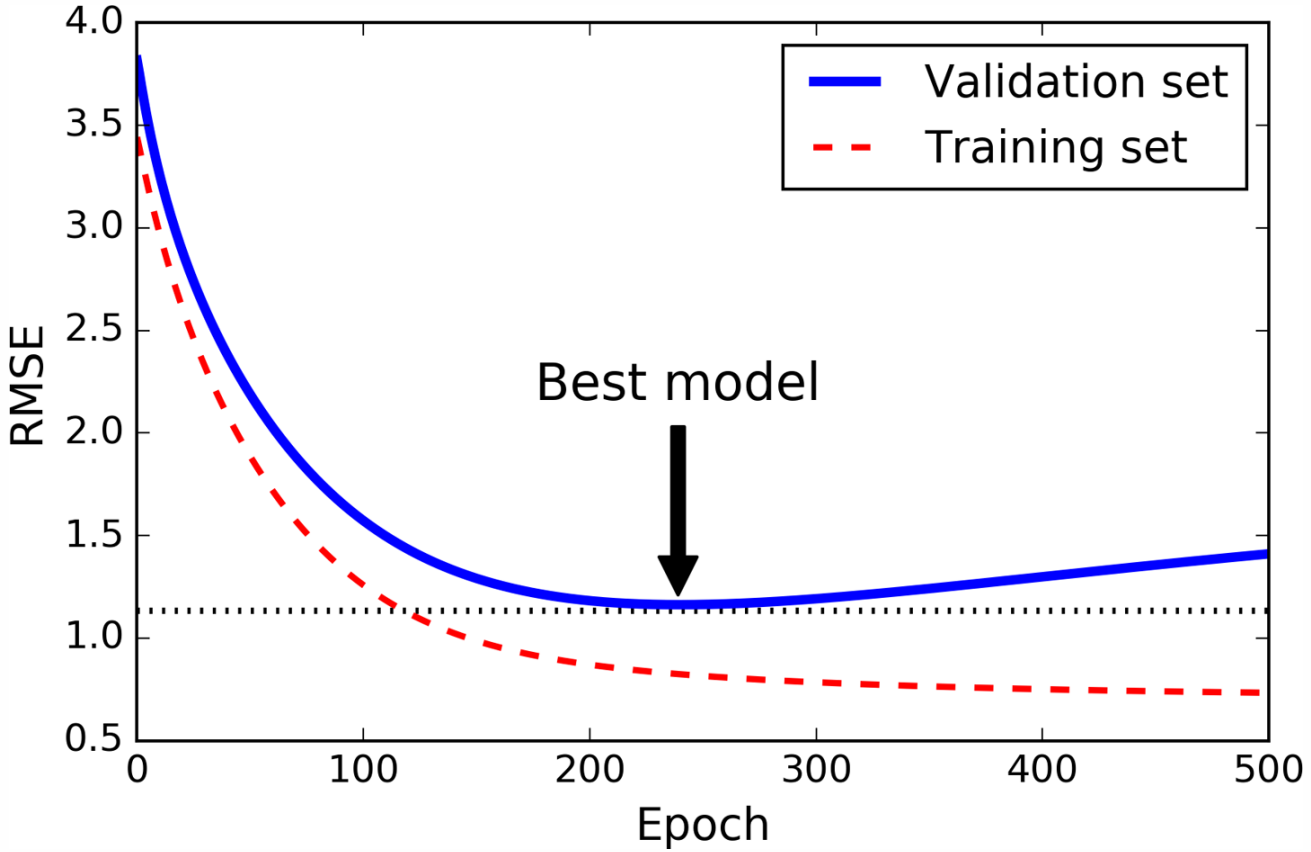
source: https://wikidocs.net/3413

Polynomial Regression and Regularisaton

- My data is often more complex than a straight line (or a hyperplane) => need to add Polynomial Features to our fitting model
- P.R. is prone to overfitting => Regularization techniques are often needed
- Ridge Regression
- Lasso Regression
  - tends to eliminate the weights of the least important features
- ElasticNet (Ridge + Lasso)

Early Stopping

A very different way to regularize iterative learning algorithms such as Gradient Descent is to stop training as soon as the validation error reaches a minimum.



Cross-validation