Class 5- Machine Learning concepts
Part II

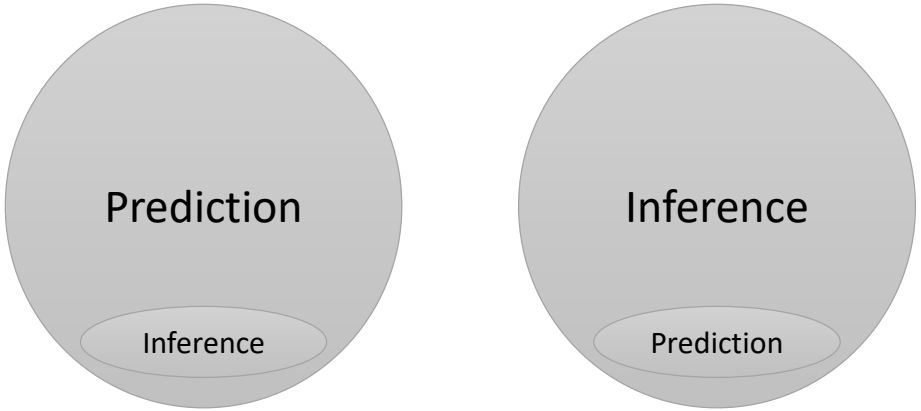
Jon M. Huntsman School of Business
UtahStateUniversity

Prof. Pedram Jahangiry

Motivation

Machine learning fundamental concepts:

- Inference and prediction
- Part I: The Model
  - Parameters and hyperparameters
  - Parametric vs nonparametric ML models
- Part II: Evaluation metrics
- Part III: Bias-Variance tradeoff
- Part IV: Resampling methods
- Part V: scaling the features
- Part VI: How do machines learn?
- Part VII: Solvers/learners (GD, SGD, Adagrad, Adam, …)

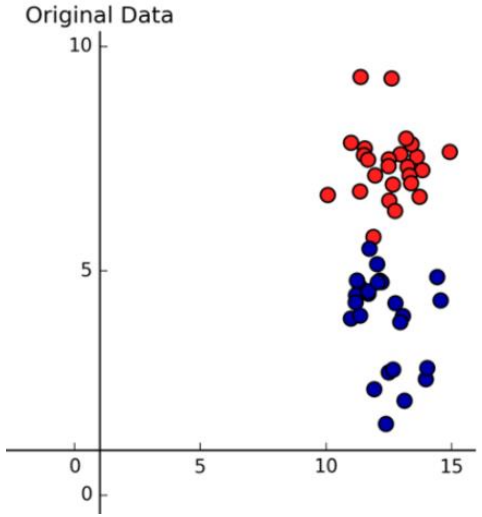


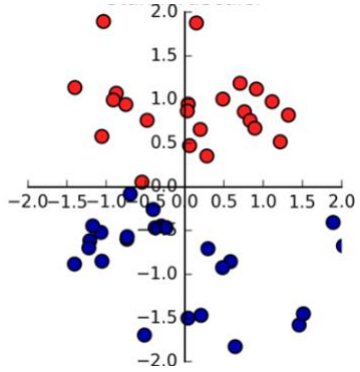
Part V
Scaling the features

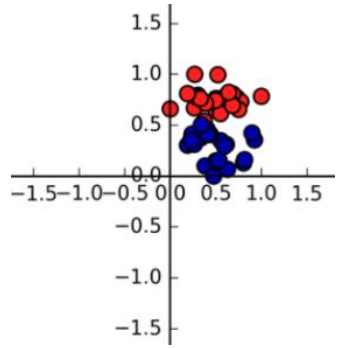
Scaling the features

Let us use x_i for raw input and $\widetilde{x_i}$ for the transformed data. Common scaling practices include:

- Standardization (Z-score): $\widetilde{x_i} = \left(\frac{x_i - \mu_x}{\sigma_x}\right)$
- Normalization:
  - Min-Max scaler over [0,1]: $\widetilde{x_i} = \left(\frac{x_i - \text{Min}(X)}{Max(X) - Min(X)}\right)$
  - Min-Max scaler over [-1,1]: $\widetilde{x_i} = 2 * \left(\frac{x_i - \text{Min}(X)}{Max(X) - Min(X)}\right) - 1$







Scaling the features

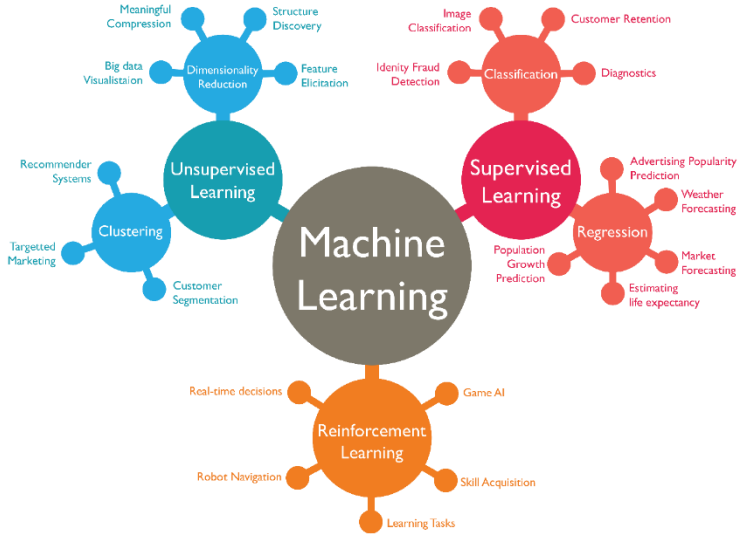
- Normalization is good to use when the distribution of the data does not follow a Normal distribution. (ideal for non-parametric algorithms like KNN)
- Standardization, can be helpful in cases where the data follows a Normal distribution. However, this does not have to be necessarily true.
- Unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.
- Be careful when scaling the time series data! Why?
- To avoid data leakage, It is a good practice to fit the scaler on the training data and then use it to transform the testing data.
- The choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using

Part VI
How do machines learn?

How do machines learn?

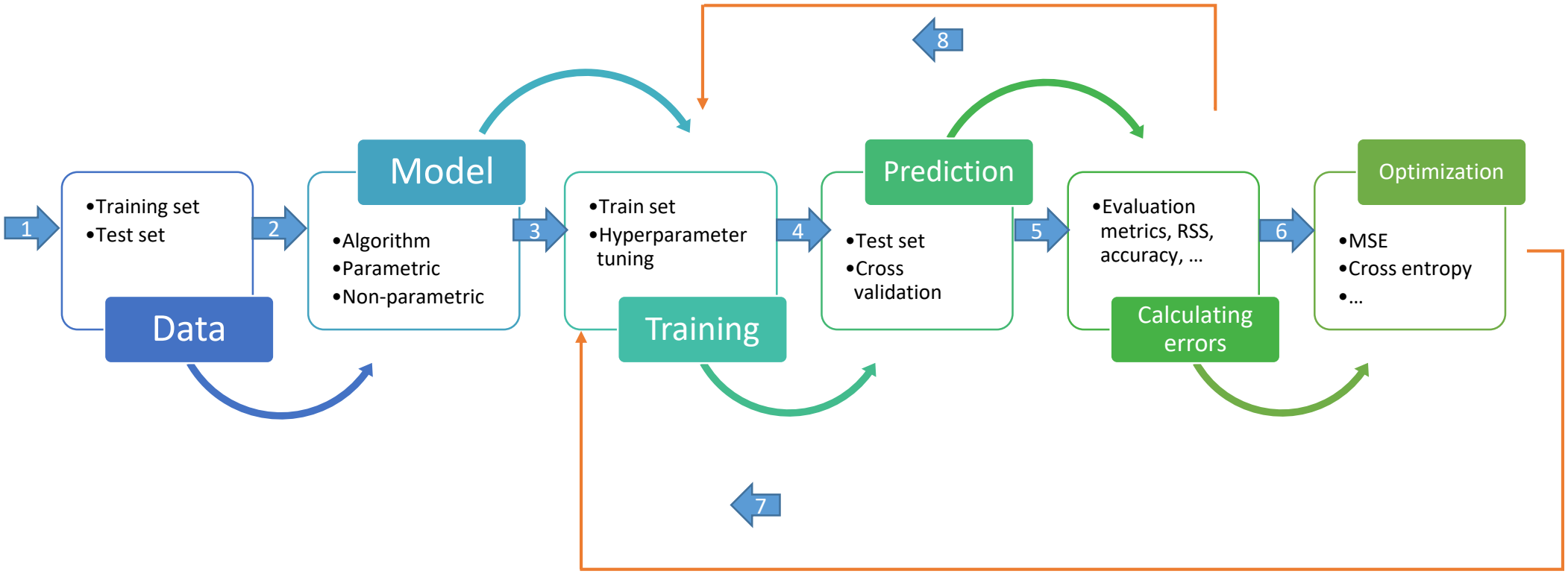
The short answer: Algorithms!

- Algorithm: a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
- Generally, the more data a machine learning algorithm is provided the more accurate it becomes.
- Different types of algorithms:
  - Supervised
  - Unsupervised
  - Reinforcement



How do machines learn?

Hyperparameter tuning using cross validation, elbow method, grid search, etc.



Updating the parameters of the model: this is the learning part

Part VII
Solvers (GD, SGD, Adagrad, Adam, …)

Solvers (learners)!

A Loss Function tells us "how good" our model is at making predictions for a given set of parameters. The cost function has its own curve and its own gradients. The slope of this curve tells us how to update our parameters to make the model more accurate.

The two most frequently used optimization algorithms when the loss function is differentiable are:

1) Gradient Descent (GD)
2) Stochastic Gradient Descent (SGD)

Gradient Descent: is an iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one starts at some random point and takes steps proportional to the negative of the gradient of the function at the current point.

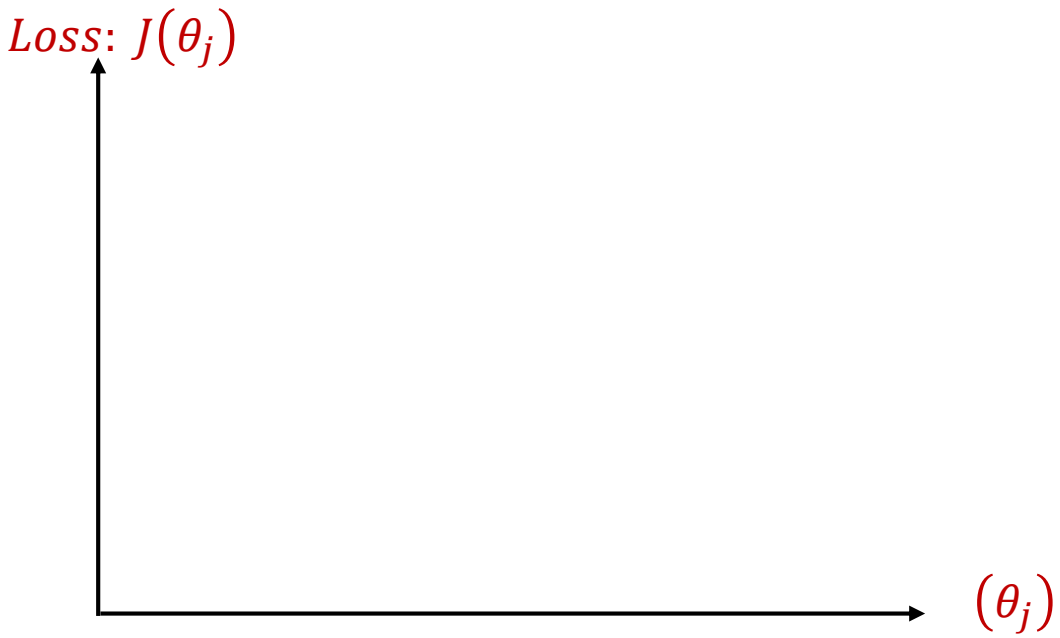
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

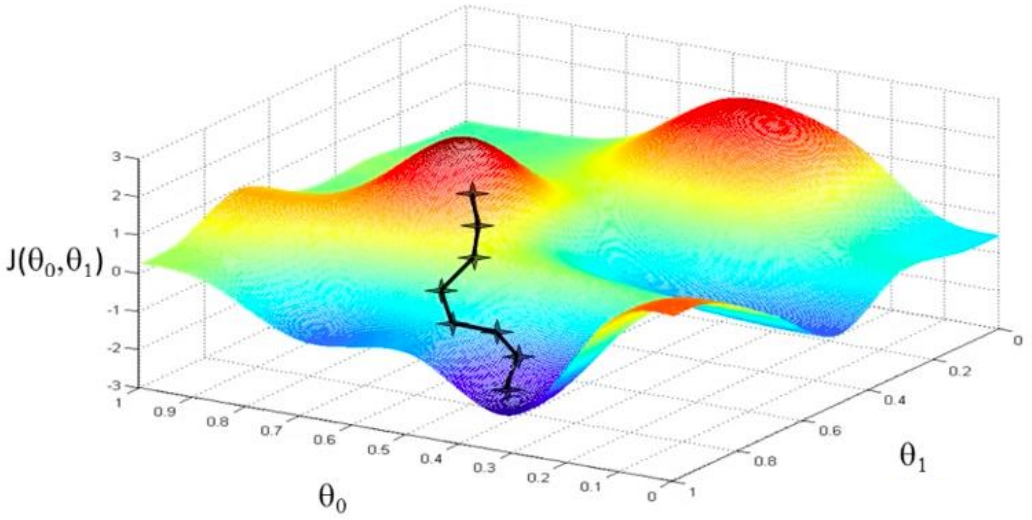
- θ_j is the model's j^{th} parameter
- α is the learning rate
- $J(\theta)$ is the loss function (which is differentiable)

Gradient Descent Visualization

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Gradient descent proceeds in epochs. An epoch consists of using the training set entirely to update each parameter. The learning rate α controls the size of an update





repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

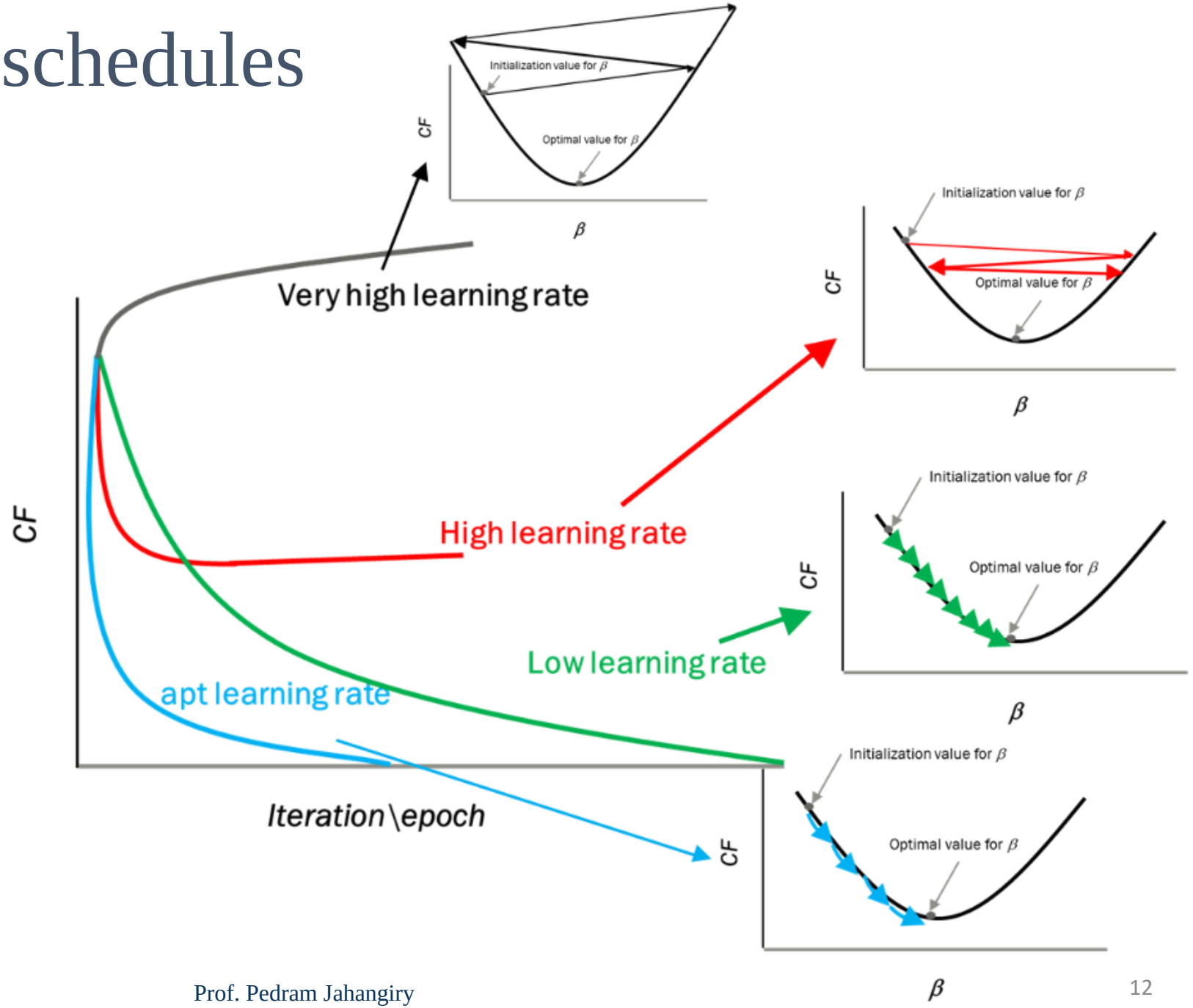
}

Learning rate schedules

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- If α is too small, gradient descent can be slow

- If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even **diverge**.



Beyond Gradient Descent?

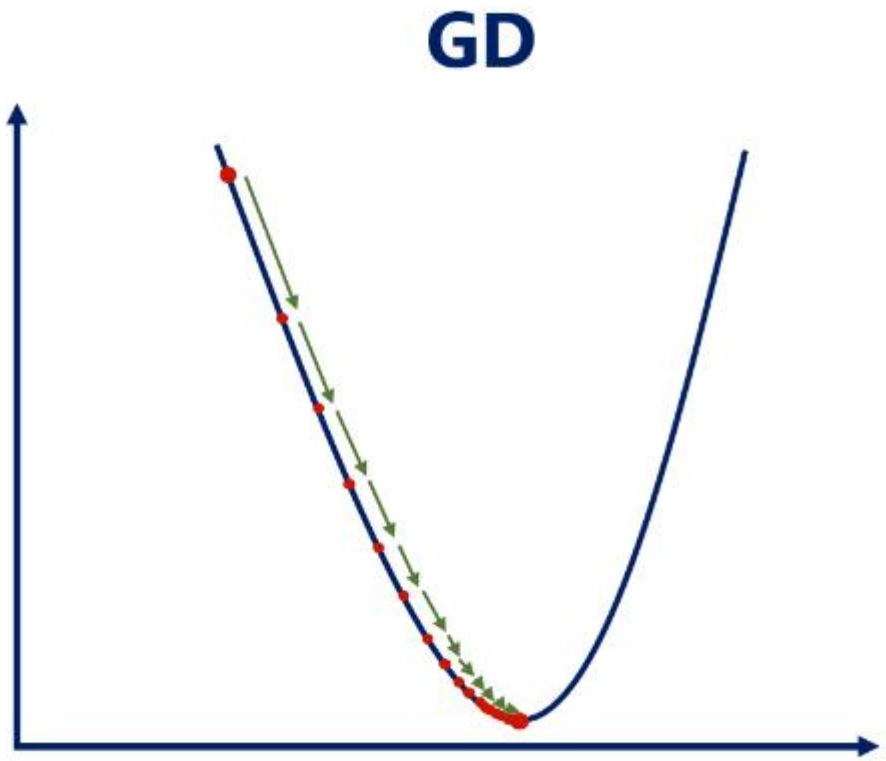
Disadvantages of gradient descent:

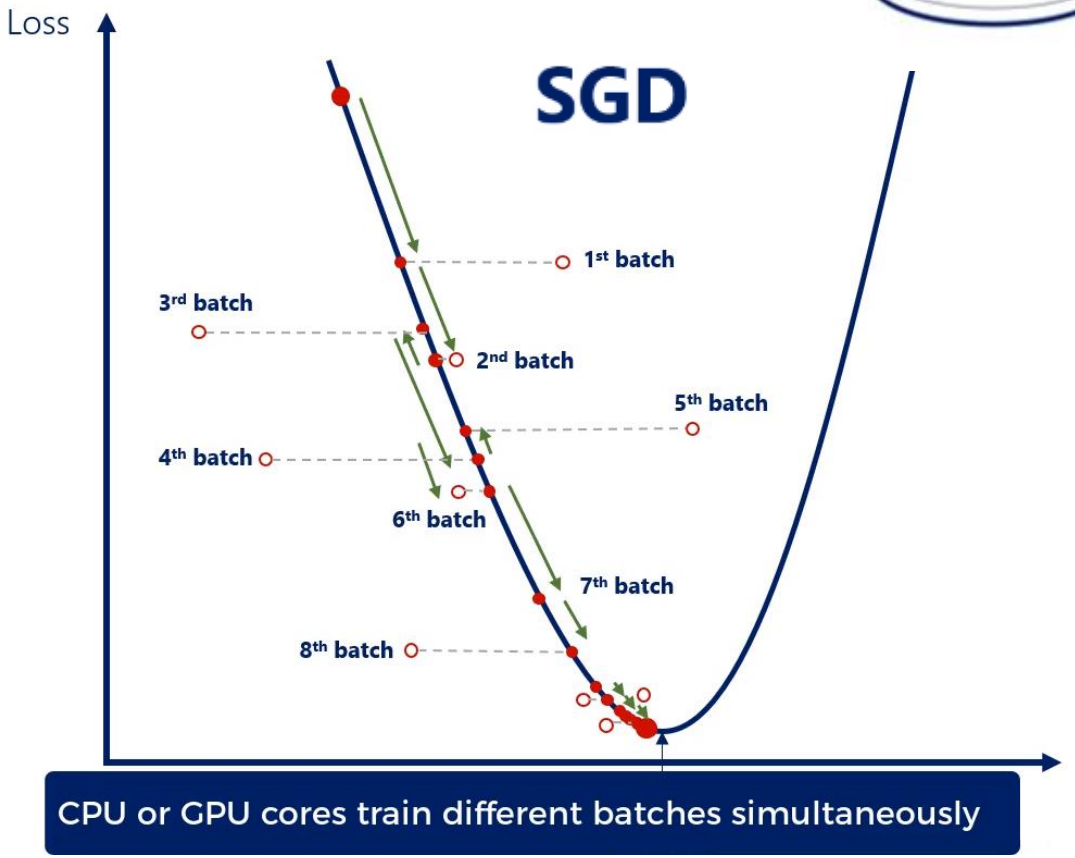
- Single batch: use the entire training set to update a parameter!
- Sensitive to the choice of the learning rate
- Slow for large datasets

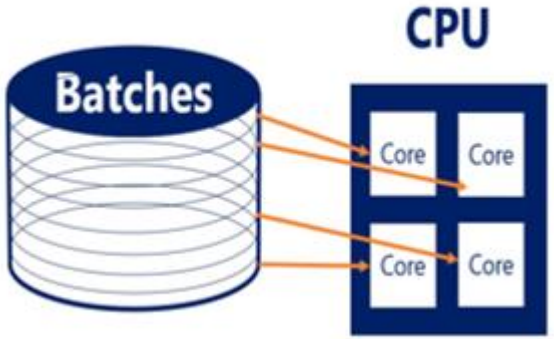
(Minibatch) Stochastic Gradient Descent: is a version of the algorithm that speeds up the computation by approximating the gradient using smaller batches (subsets) of the training data. SGD itself has various "upgrades".

1) Adagrad
2) Adam

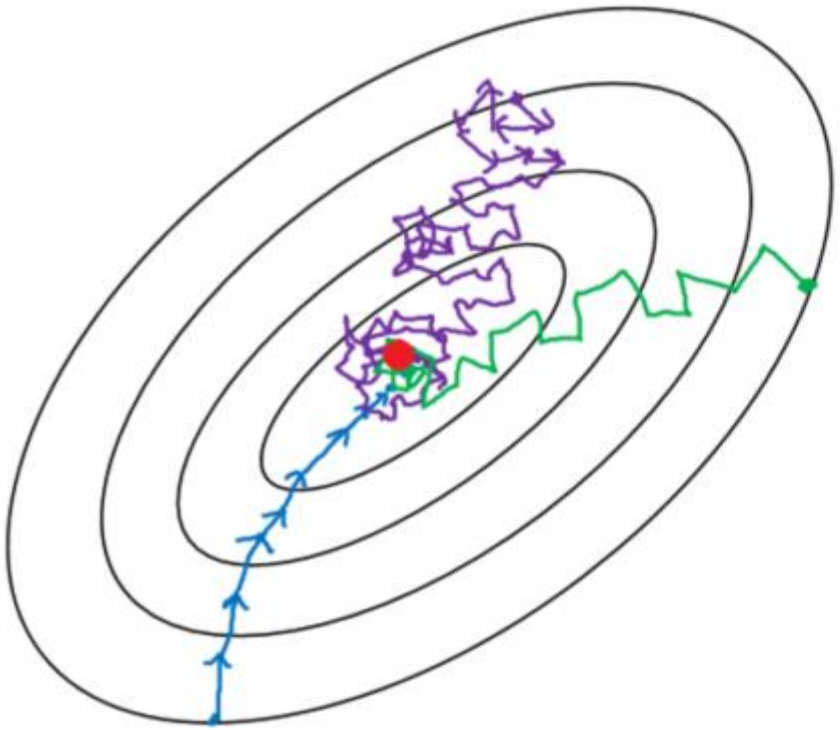
Why SGD?


GD

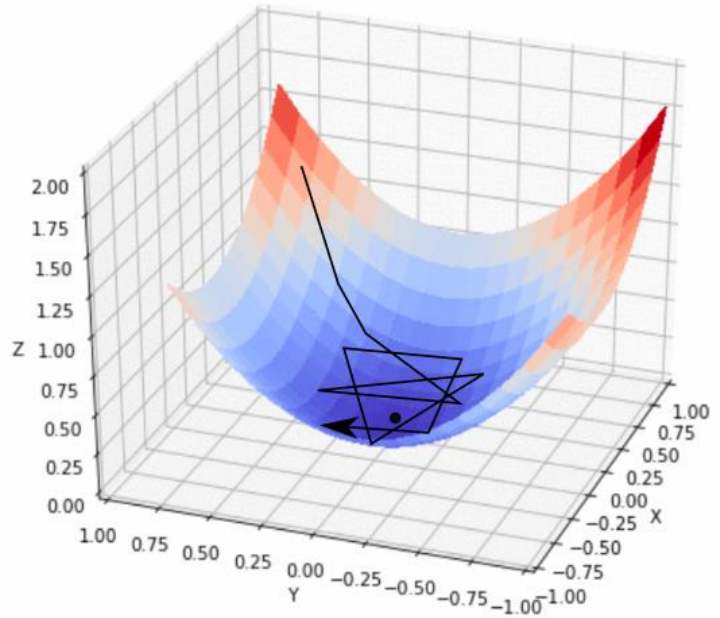

Loss
SGD
1st batch
2nd batch
3rd batch
4th batch
5th batch
6th batch
7th batch
8th batch
CPU or GPU cores train different batches simultaneously


CPU
Batches
Core
Core
Core
Core

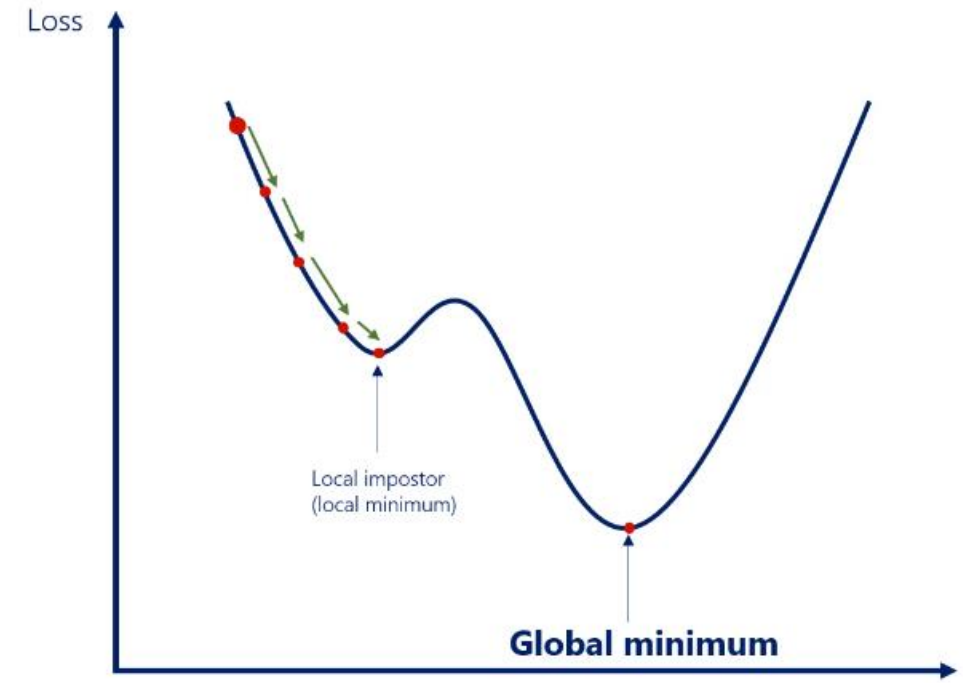
SGD vs GD

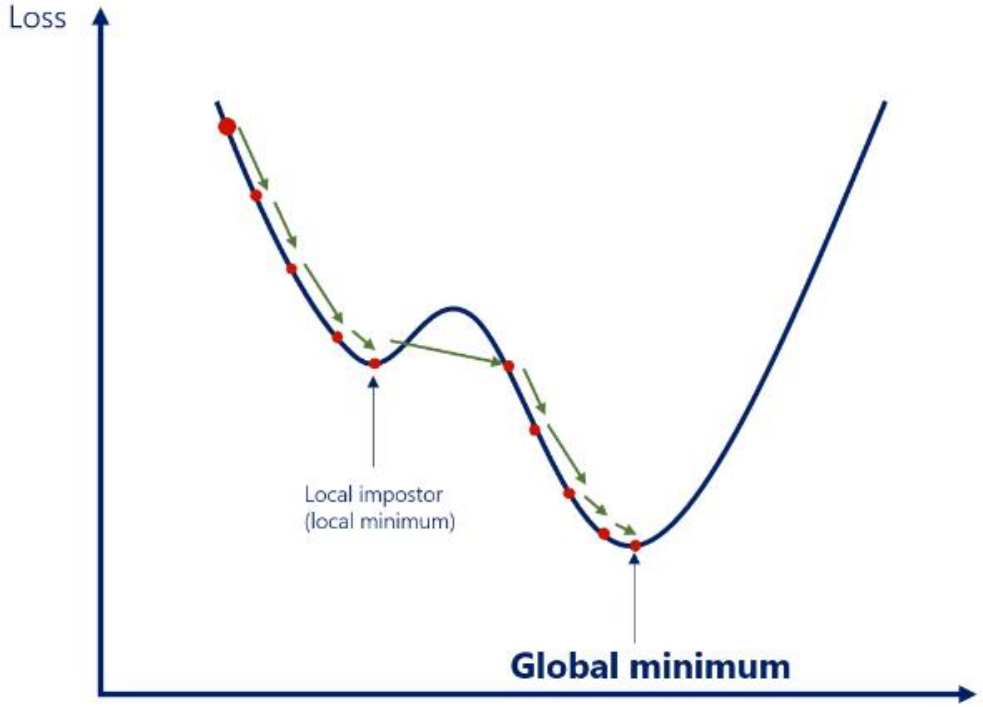


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent



Why upgrade SGD?

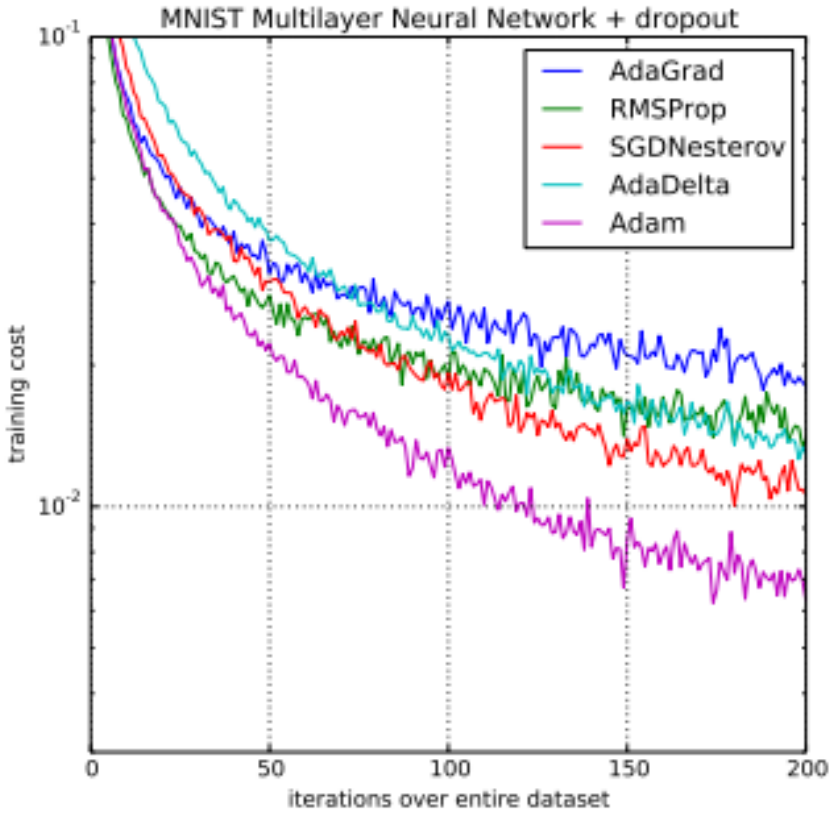




SGD upgrades

- Adagrad (Adaptive Gradient Algorithm): is a version of SGD that scales α for each parameter according to the history of gradients. As a result, is reduced for very large gradients and vice-versa.

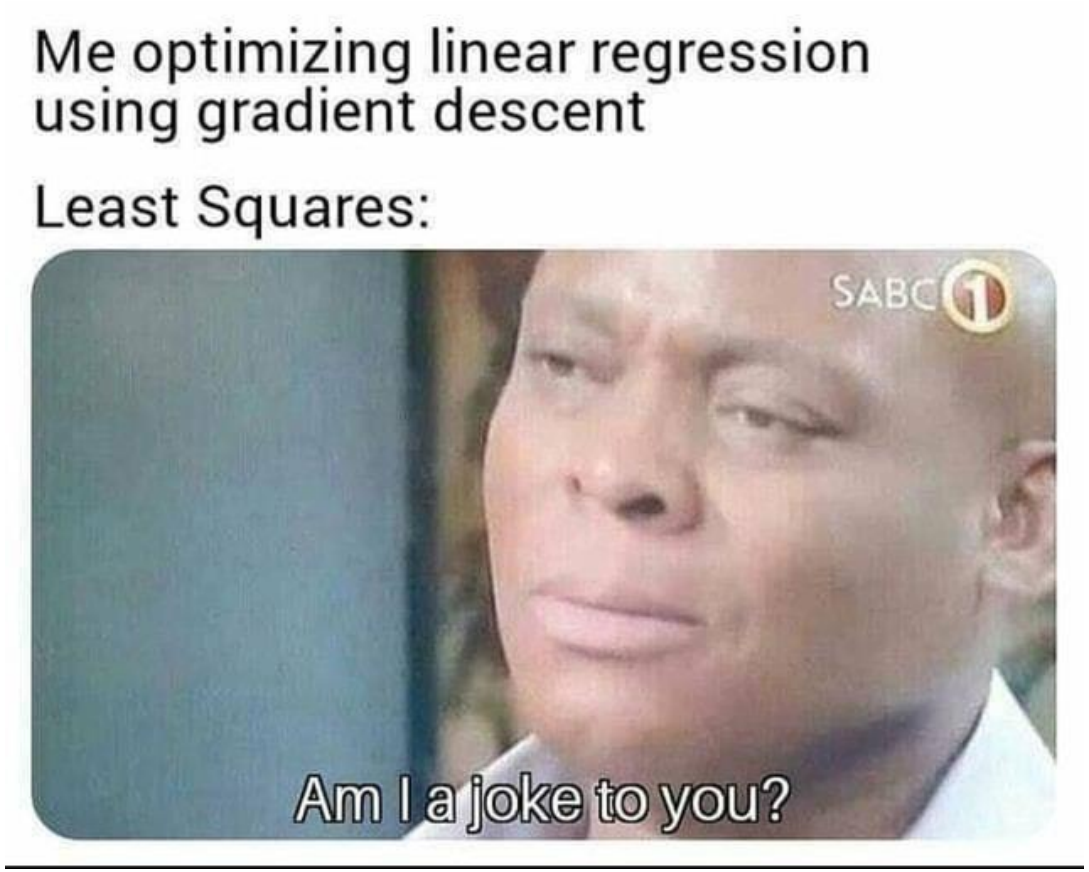
- Adam (Adaptive Moment Estimation): is a method that helps accelerate SGD by orienting the gradient descent in the relevant direction and reducing oscillations.



Final message!

Notice that gradient descent and its variants are not machine learning algorithms. They are solvers of minimization problems in which the function to minimize has a gradient (in most points of its domain).

Question of the day!



Students' questions

1) TBA


thank you all
CLASS DISMISSED