Introduction to Machine Learning

EDVANCER
EDUVENTURES

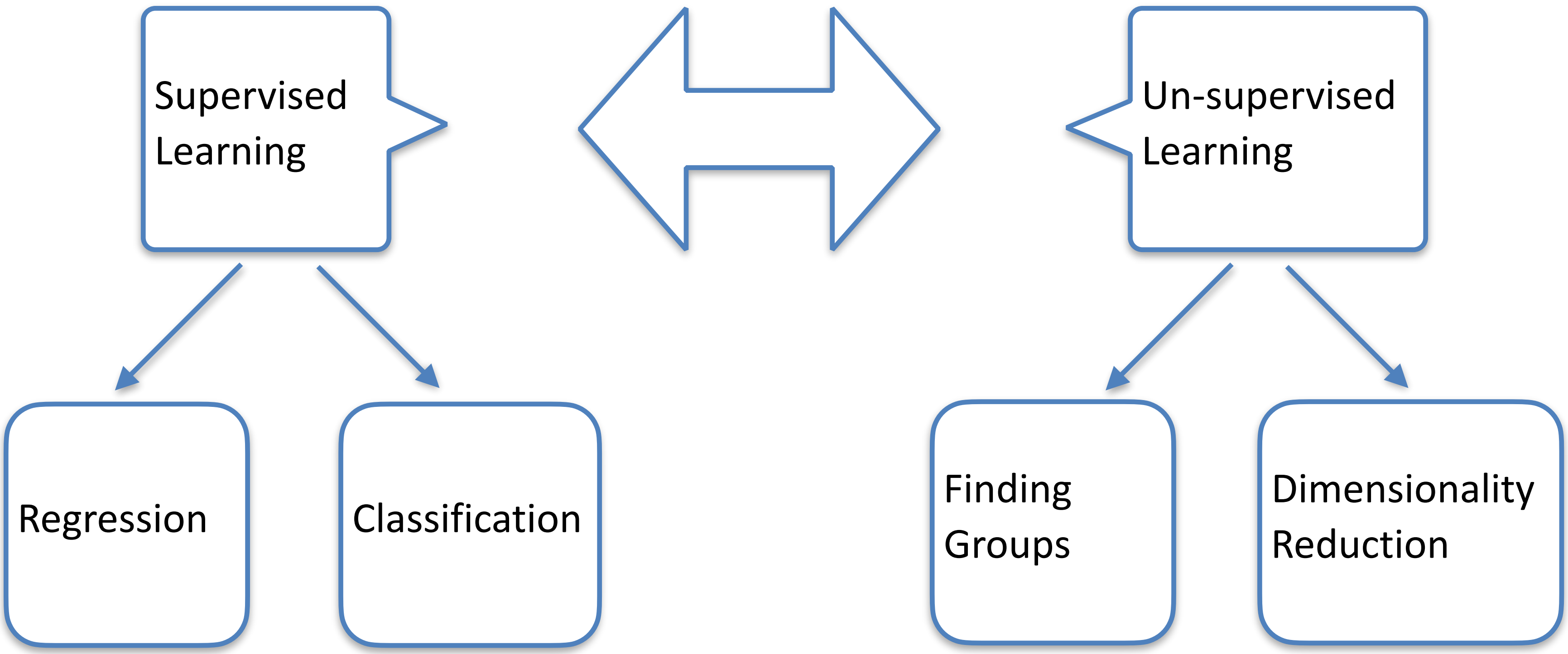
Agenda

Discussion Flow

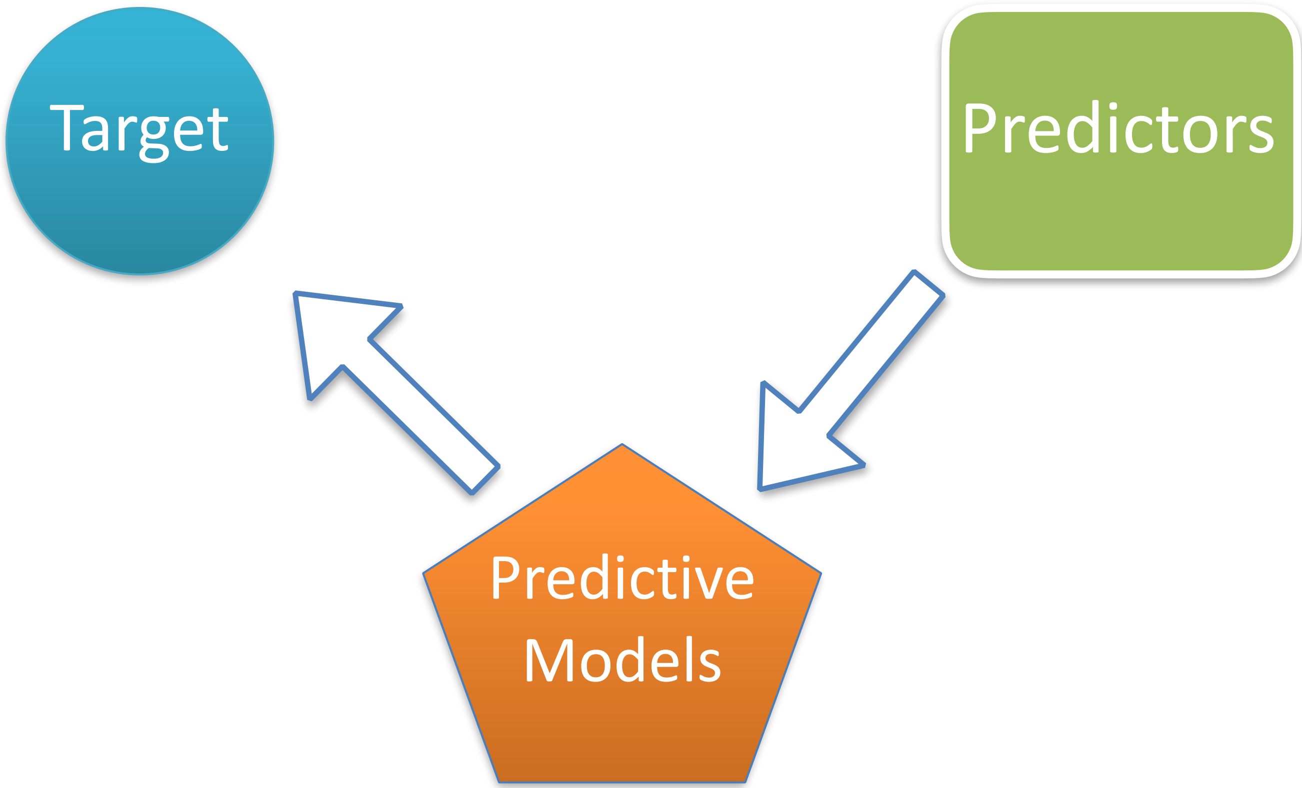
- Its all about business
- Categories of problems
- What if we make a mistake in prediction : Cost Functions
- Cost Functions to Parameters : Gradient Descent

Business Problems to ML Problem

Type of ML Problems



Components of Supervised Learning



Business Problem to ML Problems

- A bank is making too many losses because of defaulters on retail loans.
  - Target : Customer will default on their loan or not
  - Predictors : Loan Application and Customer Characteristics (Loan Amt, Age, Credit History etc)

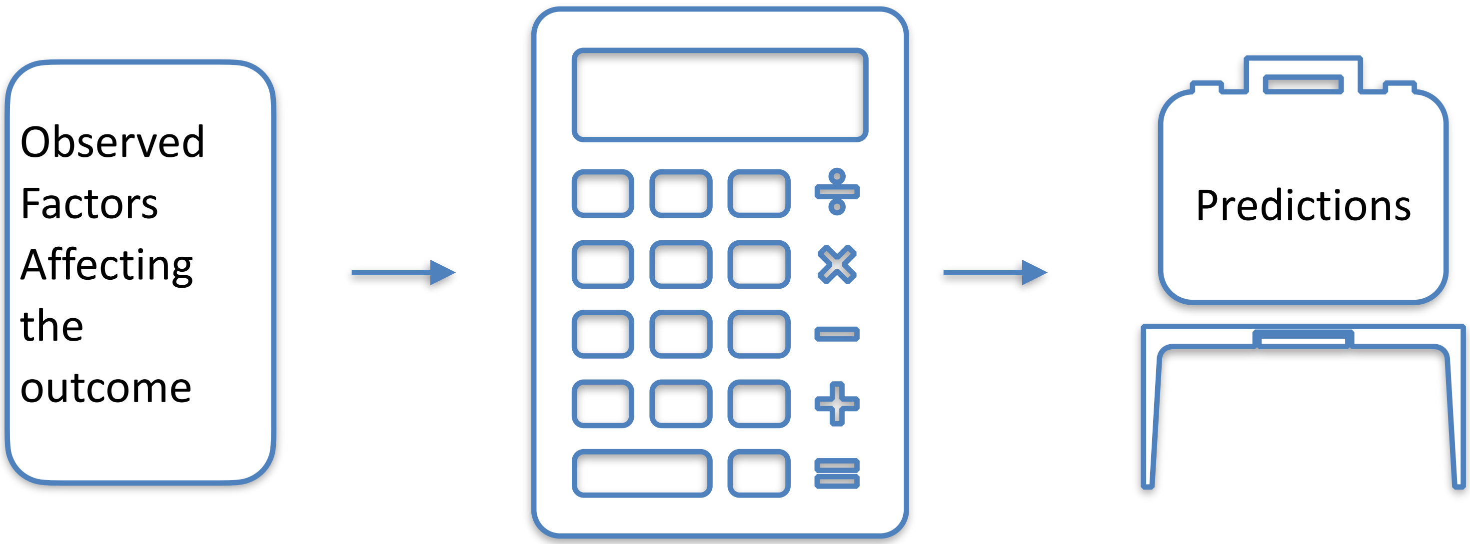
- An e-commerce company wants to know how it should plan for the budget on cloud servers
  - Target : Server Load
  - Predictors : Day of month , Number of products , sale , season etc

Are we going to ignore unsupervised Learning then?!!

- There is no target to predict here
- Unsupervised Learning concerns itself with finding structures in the data
- Finding groups and profiling them to launch targeted campaigns, products
- Anomaly detection in a data collection
- Dimensionality reduction to reduce data size for easy experimentation
- Finding latent factors from many observed factors

Cost Function

What is a predictive model



$$\hat{y}_i = f(X_i)$$

$$y_i = \hat{y}_i + \epsilon_i$$

What do we expect from a predictive model

- Its accurate
- Accuracy is not limited to a single observation
- We want an overall prediction framework which generalises very well for all the data points
- How do we quantify this collective accuracy/ performance measure ?

Cost Function Example : Linear Regression

$$f(X_i) \;=\; \beta_0 + \; \beta_1 X_{1i} + \; \beta_2 X_{2i} \ldots\ldots$$

Error in prediction for a single observation

$$\epsilon_i \;=\; y_i \;-\; \hat{y}_i$$

$$= \; y_i - \; (\beta_0 + \; \beta_1 X_{1i} + \; \beta_2 X_{2i})$$

This error can be both positive or negative

Contd..

A bad idea for collective error → $\sum_{i=1}^{n} \epsilon_i$

Simple sum of errors with different signs , means that sum can be small despite errors being large

Alternatives

- $\sum_{i=1}^{n} {\epsilon_i}^2$
- $\sum_{i=1}^{n} |\epsilon_i|$

Popular Cost Function : Sum of Squares

$$\sum_{i=1}^{n} {\epsilon_i}^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i})^2$$

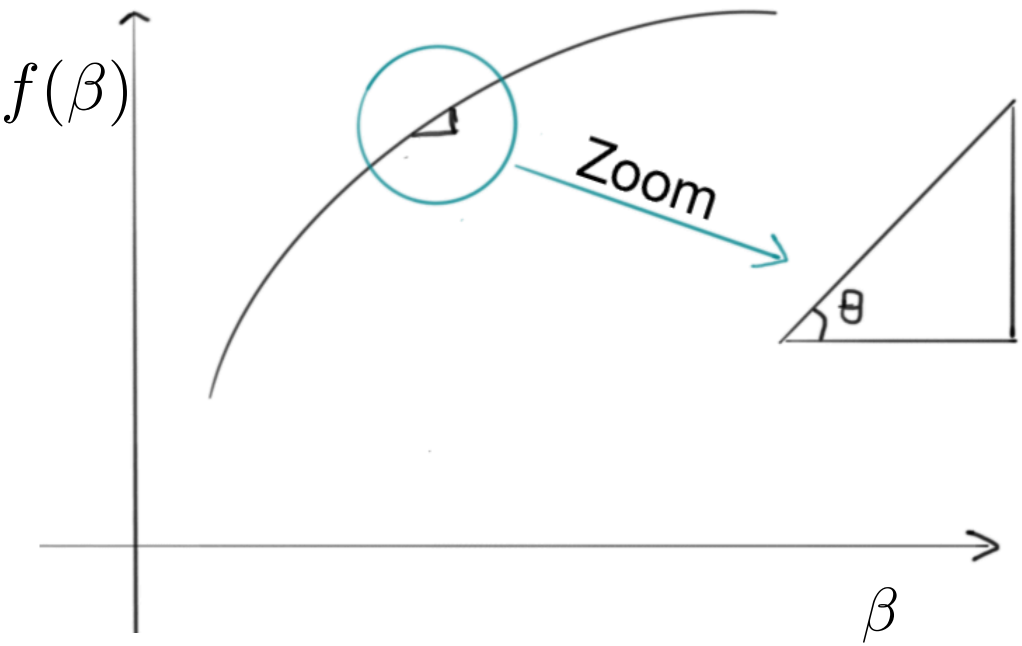
Here parameters of the cost function are betas , we'd like to determine values of betas for which cost function is minimum

An inefficient idea for finding best betas

- Start generating random values of betas
- Retain new random values every time cost function goes down

Gradient Descent

Gradient and small changes



$$\tan(\theta) = \frac{\Delta f}{\Delta \beta}$$

But tangent of a function is equal to the gradient at that point

$$\frac{\delta f}{\delta \beta} = \frac{\Delta f}{\Delta \beta} \longrightarrow \Delta f = \frac{\delta f}{\delta \beta} * \Delta \beta$$

Extrapolating the idea in higher dimension

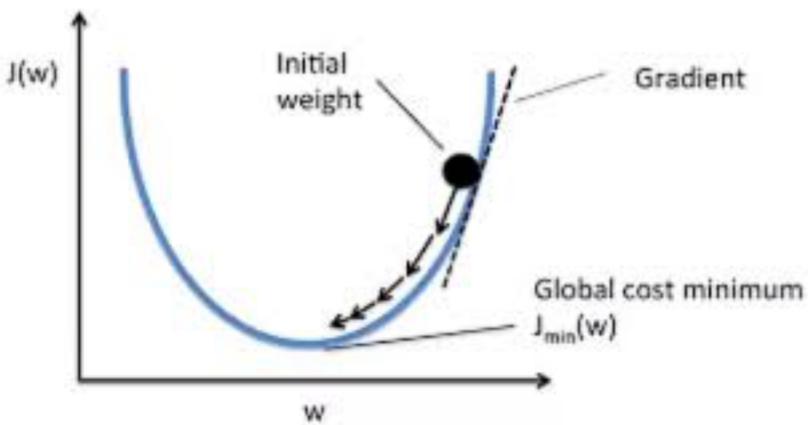
$$\frac{\delta f}{\delta \beta} \longrightarrow \left(\frac{\delta f}{\delta \beta_1}, \frac{\delta f}{\delta \beta_2}, \frac{\delta f}{\delta \beta_3}, \ldots\right) = \nabla f$$

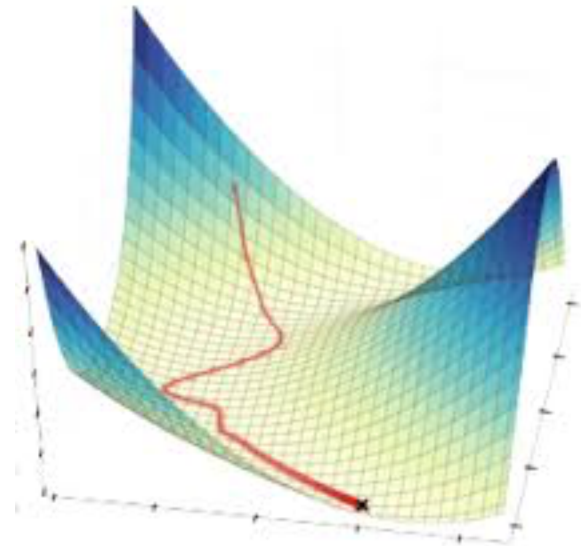
$$\Delta \beta = (\Delta \beta_1, \Delta \beta_2, \Delta \beta_3, \ldots)$$

$$\Delta f = \frac{\delta f}{\delta \beta} * \Delta \beta \longrightarrow \Delta f = \nabla f * \Delta \beta$$

A better way for determining best betas

$$\Delta\beta = -\eta * \nabla f$$





Gradient for Linear Regression

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & x_{31} & \dots & x_{p1} \\ 1 & x_{12} & x_{22} & x_{32} & \dots & x_{p2} \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots & \\ 1 & x_{1n} & x_{2n} & x_{3n} & \dots & x_{pn} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ .. \\ y_n \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_p \end{bmatrix}$$

Contd..

$$\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i})^2$$

$$f = (Y - X\beta)^T (Y - X\beta)$$

$$\nabla f = -2X^T(Y - X\beta)$$

Lets see it in action in Python