IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 5: Background on Machine Learning

Sourangshu Bhattacharya

Computer Science and Engineering

IIT KHARAGPUR

In this review

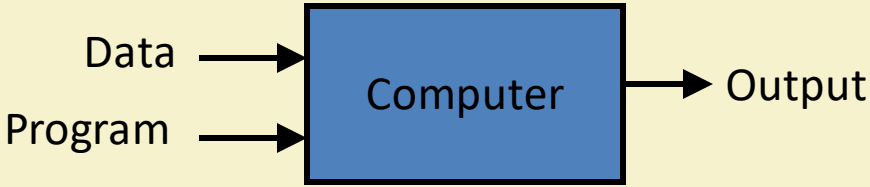
- Outline:
  - What is Machine Learning ?
  - Supervised learning
  - Linear Regression
  - Generalization
  - Classification
  - Clustering

What Is Machine Learning?

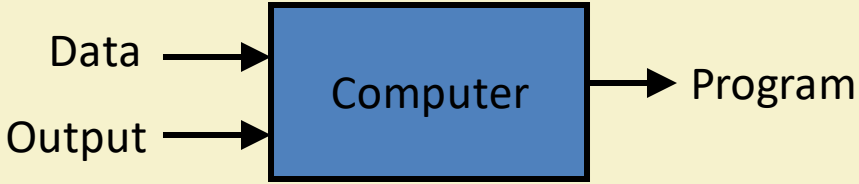
- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

What Is Machine Learning?

Traditional Programming



Machine Learning



Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
    - Representation / Model
    - Evaluation / Metric / Loss
    - Optimization / Estimation

Representation / Model

The function or set of equations, describing how input and outputs of the problem are related. Generally, equations have parameters.

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation / Metric

Describes the way of measuring the quality of output given all the inputs, (including the true output labels).

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization / Estimation

Provides a method for finding the values of parameters which achieve the best performance on the supplied dataset.

- Closed form equations – e.g.: linear regression
- Sampling based techniques – e.g. collapsed Gibbs sampling for LDA.
- Combinatorial optimization - E.g.: Grid search for hyperparameters
- Convex optimization - E.g.: Stochastic Gradient descent
- Constrained optimization - E.g.: Linear programming

Types of Learning

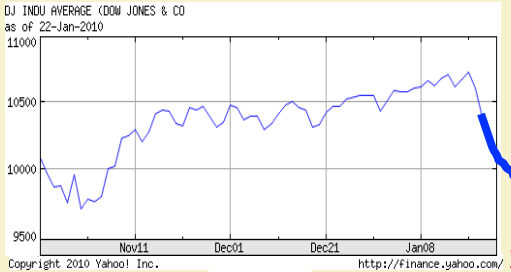
- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Supervised Learning

Supervised Learning

Goal: Construct a **predictor** $f : X \to Y$ to minimize loss function (performance measure)

Sports
Science
News



Y = ?

X = Feb01

Classification:

$$P(f(X) \neq Y)$$

Probability of Error

Regression:

$$\mathbb{E}[(f(X) - Y)^2]$$

Mean Squared Error

Regression algorithms

Training data $\{(X_i, Y_i)\}_{i=1}^{n}$ → **Learning algorithm** → Prediction rule $\widehat{f}_n$

Linear Regression

Replace Expectation with Empirical Mean

Optimal predictor: $$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

Empirical Minimizer: $$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Empirical mean

Law of Large Numbers:

$$\frac{1}{n} \sum_{i=1}^{n} [\text{loss}(Y_i, f(X_i))] \xrightarrow{n \to \infty} \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$$

Restrict class of predictors

Optimal predictor: $$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

Empirical Minimizer: $$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

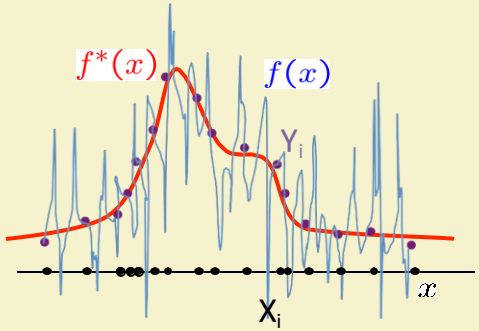
Class of predictors

Why?

Overfitting!

Empiricial loss minimized by any function of the form

$$f(x) = \begin{cases} Y_i, & x = X_i \text{ for } i = 1, \ldots, n \\ \text{any value}, & \text{otherwise} \end{cases}$$



Restrict class of predictors

Optimal predictor: $$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

Empirical Minimizer: $$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Class of predictors

- Class of Linear functions
- Class of Polynomial functions
- Class of nonlinear functions

Linear Regression

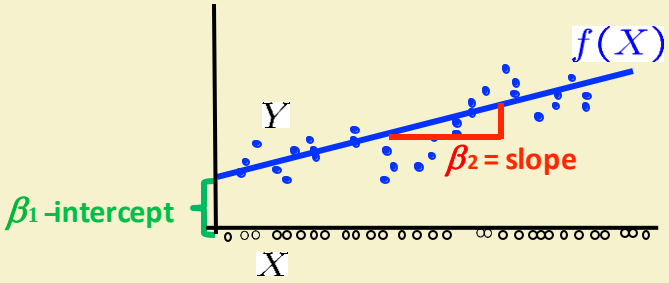
$$\hat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Least Squares Estimator

$\mathcal{F}_L$ –Class of Linear functions

Uni–variate case:

$$f(X) = \beta_1 + \beta_2 X$$



Multi–variate case:

$$f(X) = f(X^{(1)}, \ldots, X^{(p)}) = \beta_1 X^{(1)} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)}$$

$$= X\beta \qquad \text{where} \quad X = [X^{(1)} \ldots X^{(p)}], \quad \beta = [\beta_1 \ldots \beta_p]^T$$

Least Squares Estimator

$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

$$f(X_i) = X_i \beta$$

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (X_i \beta - Y_i)^2$$

$$\widehat{f}_n^L(X) = X\widehat{\beta}$$

$$= \arg\min_{\beta} \frac{1}{n} (\mathbf{A}\beta - \mathbf{Y})^T (\mathbf{A}\beta - \mathbf{Y})$$

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix}$$

Least Squares Estimator

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0$$

Normal Equations

$$\underset{p \times p}{(\mathbf{A}^T\mathbf{A})}\underset{p \times 1}{\widehat{\beta}} = \underset{p \times 1}{\mathbf{A}^T\mathbf{Y}}$$

If $(\mathbf{A}^T\mathbf{A})$ is invertible,

$$\boxed{\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \widehat{f}_n^L(X) = X\widehat{\beta}}$$

When is $(\mathbf{A}^T\mathbf{A})$ invertible ?
Recall: Full rank matrices are invertible.

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?

Non-linear basis functions

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for j=0 … n

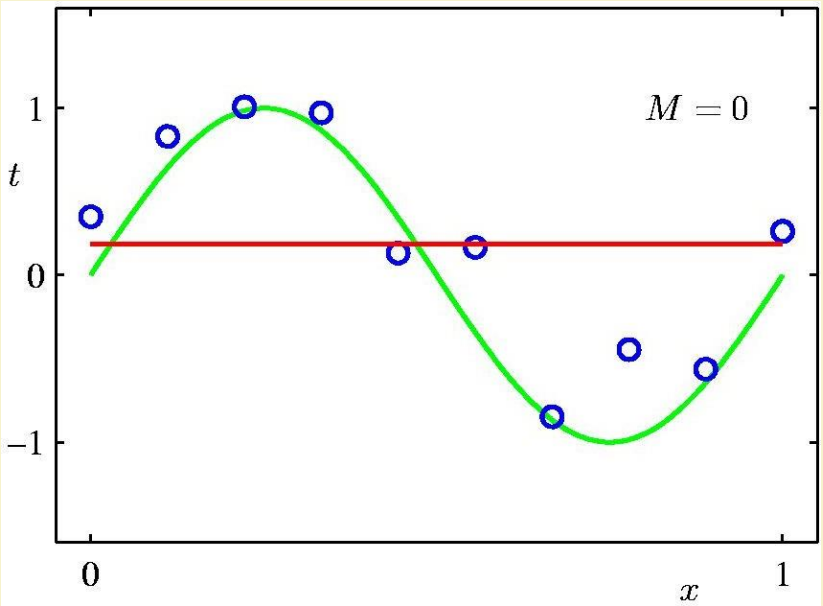
- Gaussian: $\phi_j(x) = \frac{(x-\mu_j)}{2\sigma_j^2}$

- Sigmoid: $\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$

- Logs: $\phi_j(x) = \log(x+1)$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

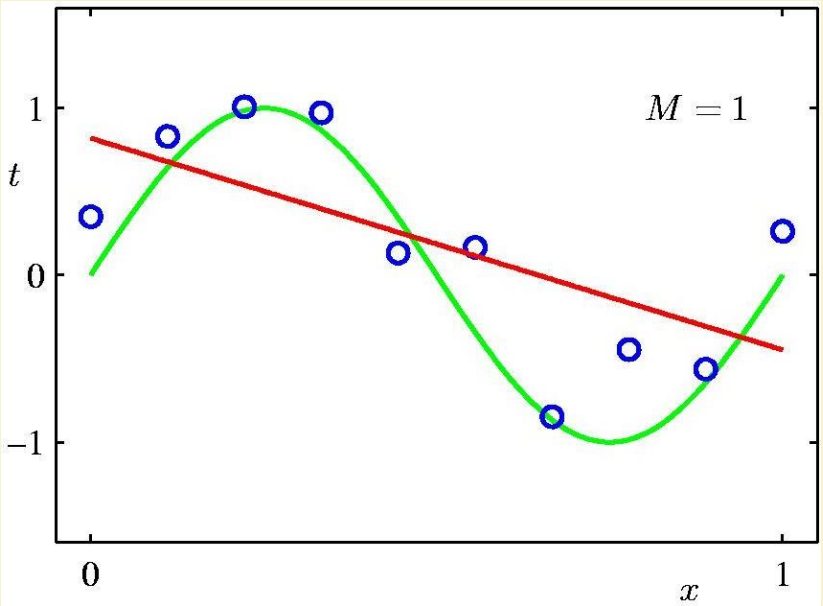
0th Order Polynomial



n=10

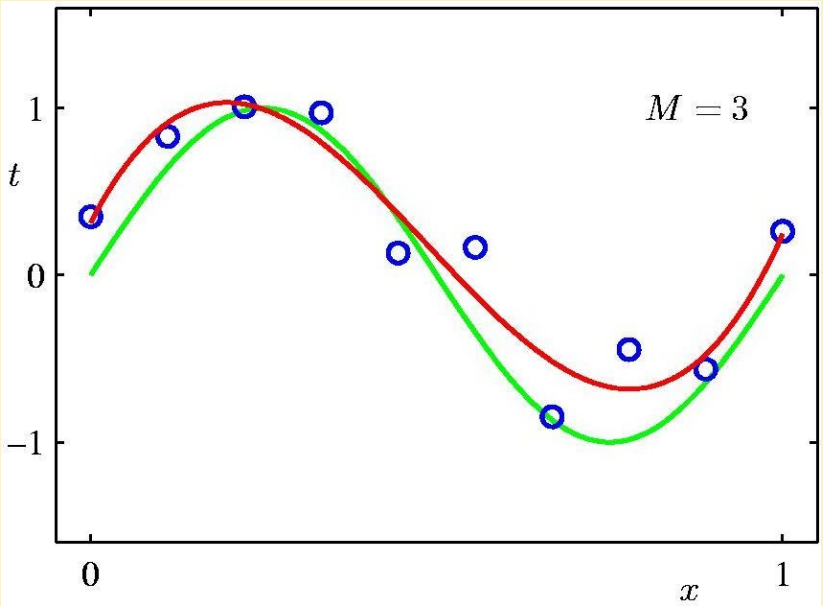
Slide courtesy of William Cohen

1st Order Polynomial



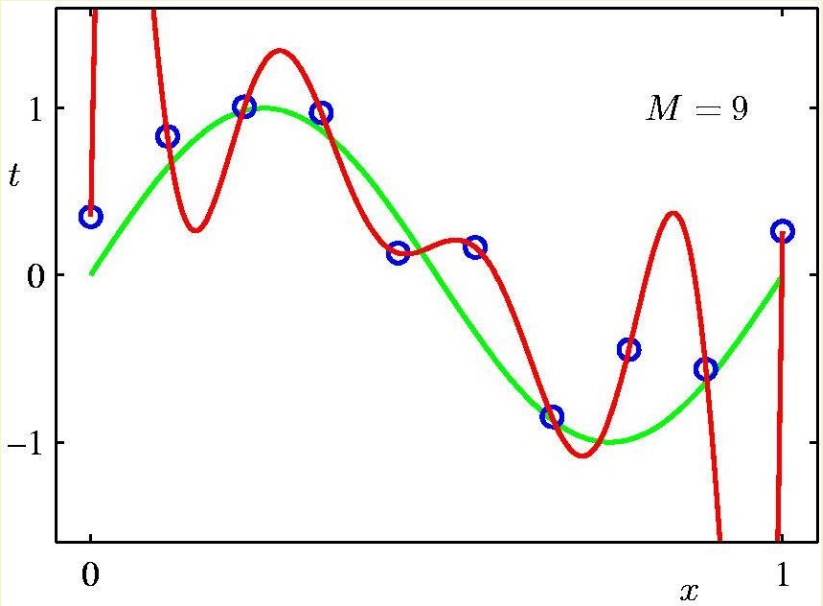
Slide courtesy of William Cohen

3rd Order Polynomial



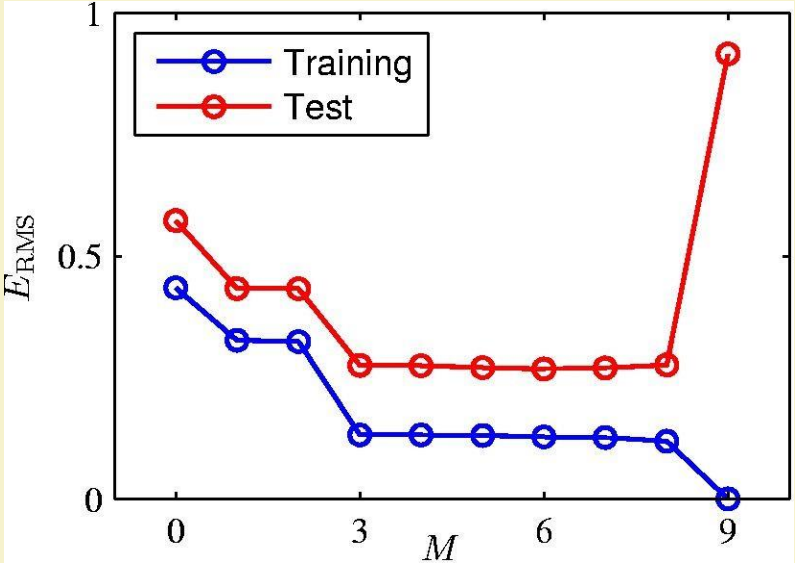
Slide courtesy of William Cohen

9th Order Polynomial



Slide courtesy of William Cohen

Over-fitting



Root-Mean-Square (RMS) Error

Slide courtesy of William Cohen

Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Regularization

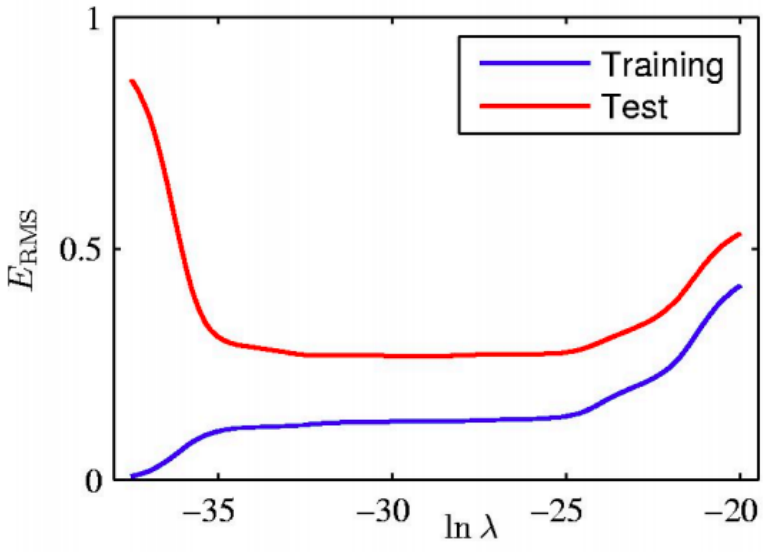
Penalize large coefficient values

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \frac{1}{2}\sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Slide courtesy of William Cohen

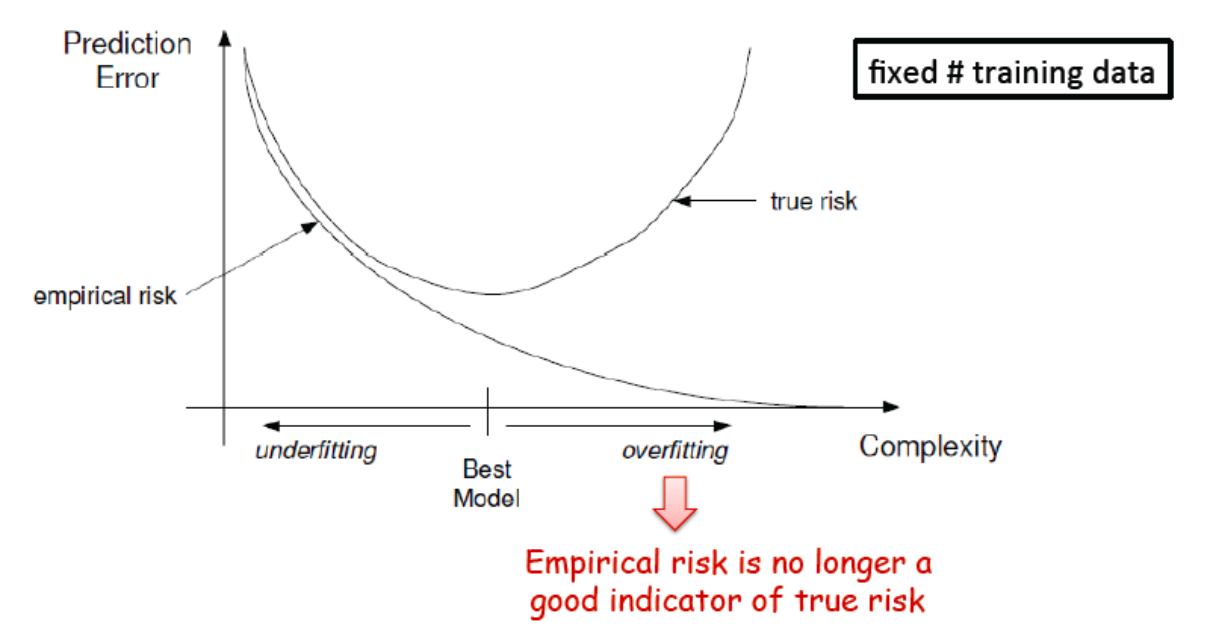
Regularization

9th Order Polynomial



Effect of Model Complexity

- If we allow very complicated predictors, we could overfit the training data.



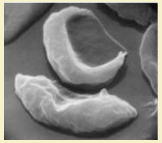
Discrete and Continuous Labels

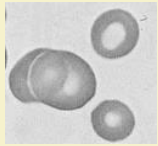
Classification

X = Document

Sports
Science
News

Y = Topic





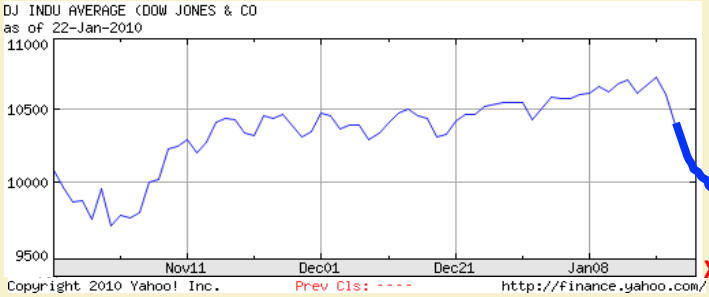
X = Cell Image

Anemic cell
Healthy cell

Y = Diagnosis

Regression

Stock Market Prediction



Y = ?

X = Feb01

An example application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

From Linear to Logistic Regression

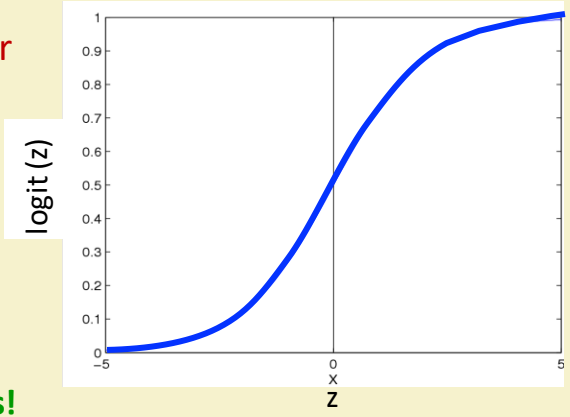
Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to a linear function of the data

Logistic function (or Sigmoid):

$$\frac{1}{1+exp(-z)}$$



Features can be discrete or continuous!

Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

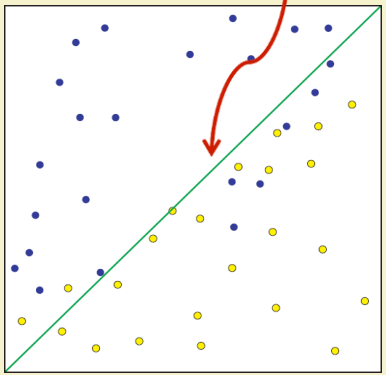
Decision boundary:

$$P(Y=0|X) \underset{1}{\overset{0}{\gtrless}} P(Y=1|X)$$

$$w_0 + \sum_i w_i X_i \underset{1}{\overset{0}{\gtrless}} 0$$

(Linear Decision Boundary)

$$w_0 + \sum_i w_i X_i = 0$$



Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow P(Y=0|X) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1+\exp(w_0 + \sum_i w_i X_i)}$$

$$\Rightarrow \frac{P(Y=0|X)}{P(Y=1|X)} = \exp(w_0 + \sum_i w_i X_i) \underset{1}{\overset{0}{\gtrless}} 1$$

$$\Rightarrow \boxed{w_0 + \sum_i w_i X_i \underset{1}{\overset{0}{\gtrless}} 0}$$

Other classifiers

- Naïve Bayes
- Support vector Machines
- Neural Networks.
- K- nearest neighbors.
- Random Forests
- etc.

Unsupervised Learning

The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
  - *Latent Semantic Indexing*, a dimensionality reduction technique used for data visualization or data pre-processing before supervised techniques are applied, and
  - *Clustering*, a broad class of methods for discovering unknown subgroups in data.

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

K-Means

- Assumes datapoints are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\vec{\mu}(c) = \frac{1}{|c|} \sum_{\vec{x} \in c} \vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select K random datapoints $\{s_1, s_2, \ldots s_K\}$ as seeds.

Until clustering *converges* (or other stopping criterion):

For each datapoint d_i:

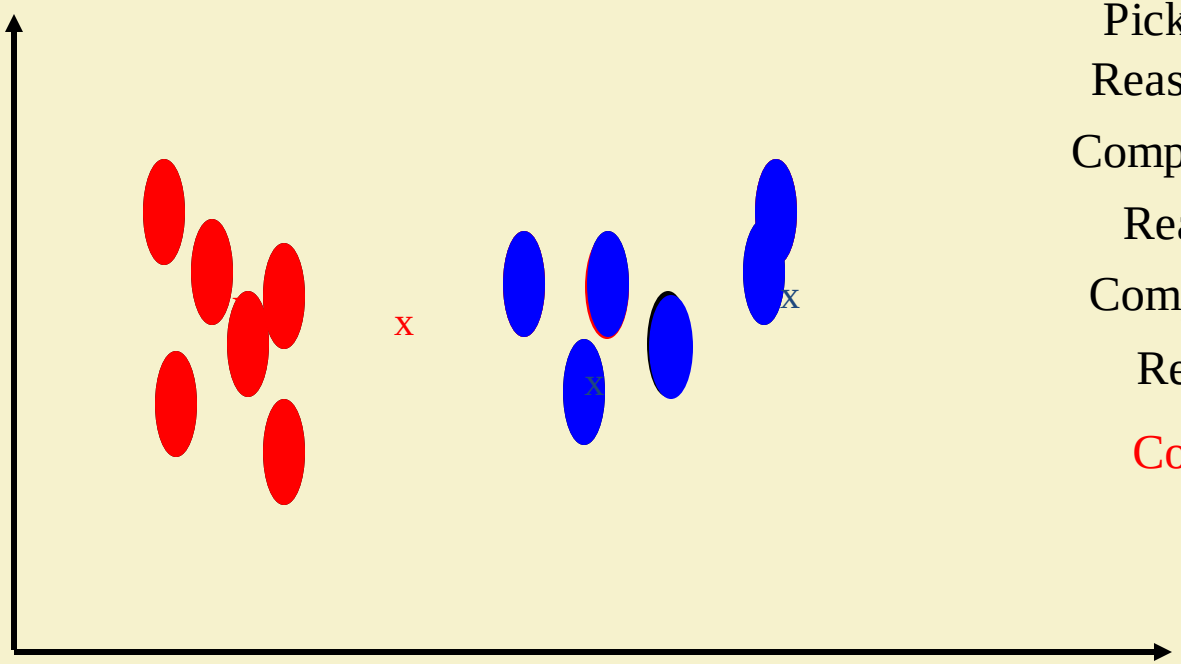
Assign d_i to the cluster c_j such that $dist(x_i, s_j)$ is minimal.

(*Next, update the seeds to the centroid of each cluster*)

For each cluster c_j

$s_j = \mu(c_j)$

K Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations.
  - Partition unchanged.
  - Centroid positions don’t change.

Does this mean that the datapoints in a cluster are unchanged?

Convergence of K-Means

- Define goodness measure of cluster *k* as sum of squared distances from cluster centroid:
  - $G_k = \Sigma_i (d_i - c_k)^2$ (sum over all d_i in cluster *k*)
- $G = \Sigma_k G_k$
- Reassignment monotonically decreases G since each vector is assigned to the closest centroid.

Convergence of K-Means

- Recomputation monotonically decreases each G_k since (m_k is number of members in cluster k):
  - $\Sigma\ (d_i - a)^2$ reaches minimum for:
  - $\Sigma\ -2(d_i - a) = 0$
  - $\Sigma\ d_i = \Sigma\ a$
  - $m_K\ a = \Sigma\ d_i$
  - $a = (1/\ m_k)\ \Sigma\ d_i = c_k$
- *K*-means typically converges quickly

Time Complexity

- Computing distance between two datapoints is O*(M)* where *M* is the dimensionality of the vectors.
- Reassigning clusters: O*(KN)* distance computations, or O*(KNM).*
- Computing centroids: Each datapoint gets added once to some centroid: O*(NM).*
- Assume these two steps are each done once for *I* iterations: O*(IKNM).*

References:

- Christopher M. Bishop. **Pattern Recognition and Machine Learning.** *Springer-Verlag New York Inc.; 1st ed. 2006.*

- Many other books.

- Wikipedia.

Thank You!!