DS 4400

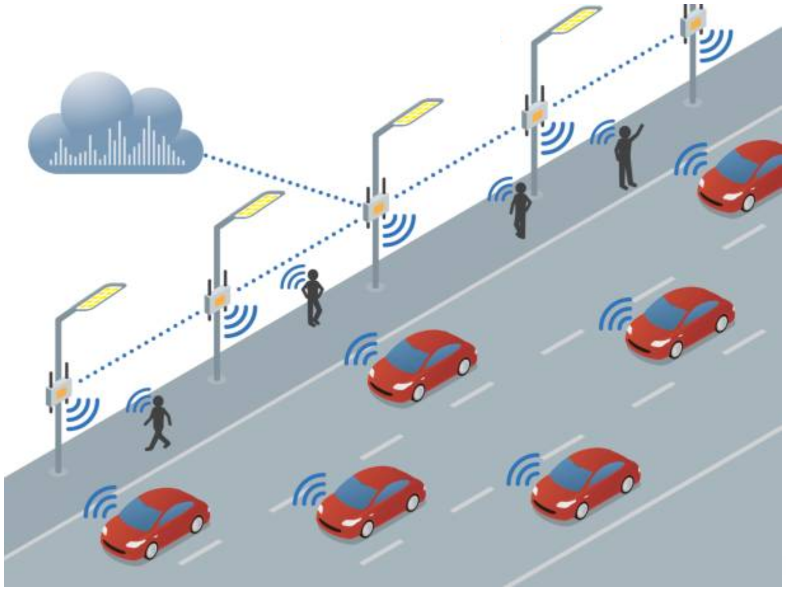
Machine Learning and Data Mining I

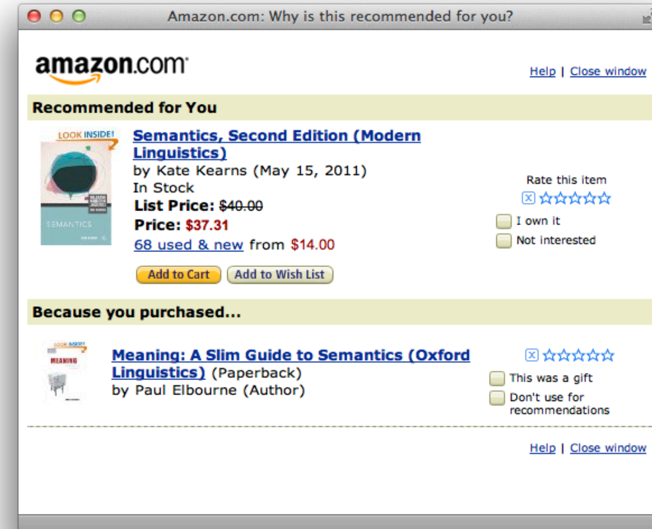
Alina Oprea
Associate Professor, CCIS
Northeastern University

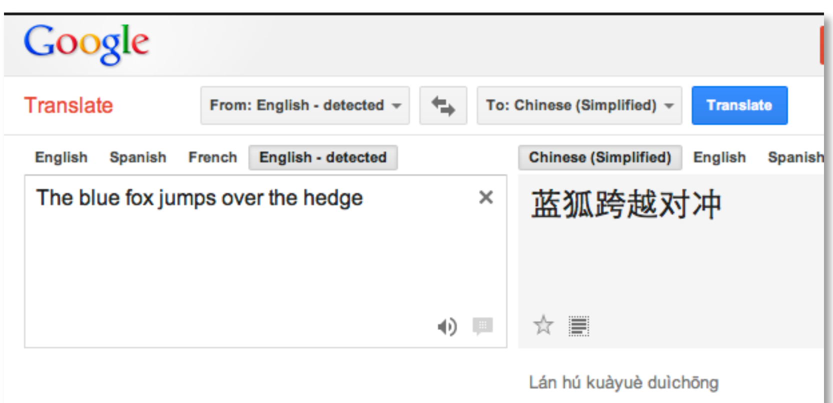
October 23 2019

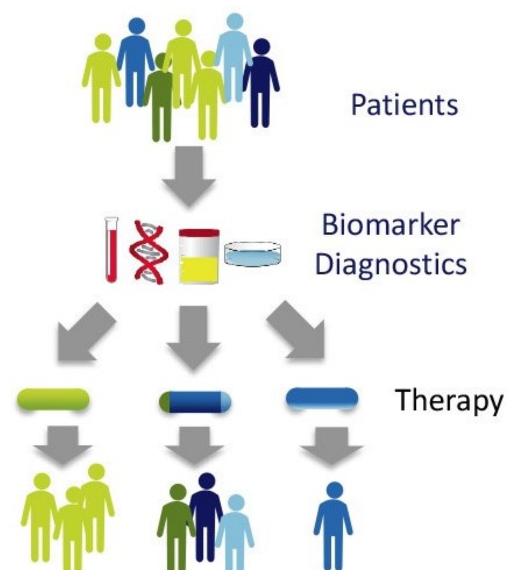
Midterm Review

Machine learning is everywhere









What we covered so far

Linear classification
- Perceptron
- Logistic regression
- LDA

Non-linear classification
- kNN
- Decision trees
- Naïve Bayes

- Metrics
- Cross-validation
- Regularization
- Feature selection
- Gradient Descent
- Maximum Likelihood Estimation (MLE)

Linear Regression

Linear algebra

Probability and statistics

Terminology

- Hypothesis space $H = \{f: X \rightarrow Y\}$
- Training data $\mathrm{D} = (x_{\mathrm{i}}, y_{\mathrm{i}}) \in X \times Y$
- Features: $x_i \in X$
- Labels / response variables $y_{\mathrm{i}} \in Y$
  - Classification: discrete $y_{\mathrm{i}} \in \{0,1\}$
  - Regression: $y_{\mathrm{i}} \in \mathrm{R}$
- Loss function: $L(f, D)$
  - Measures how well f fits training data
- Training algorithm: Find hypothesis $\hat{f}: X \rightarrow Y$
  - $\hat{f} = \operatorname*{argmin}_{f \in H} L(f, D)$

Supervised Learning: Classification

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled | Normalization | Feature Selection | Classification

$x_i, y_i \in \{0,1\}$ | $f(x)$

Testing

New data → Learning model → Predictions $y' = f(x') \in \{0,1\}$

Unlabeled | $f(x)$

x'

Positive
Negative

Classification

Supervised Learning: Regression

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled

$x_i, y_i \in R$

Normalization

Feature Selection

Regression

$f(x)$

Testing

New data → Learning model → Predictions

$y' = f(x') \in R$

Unlabeled

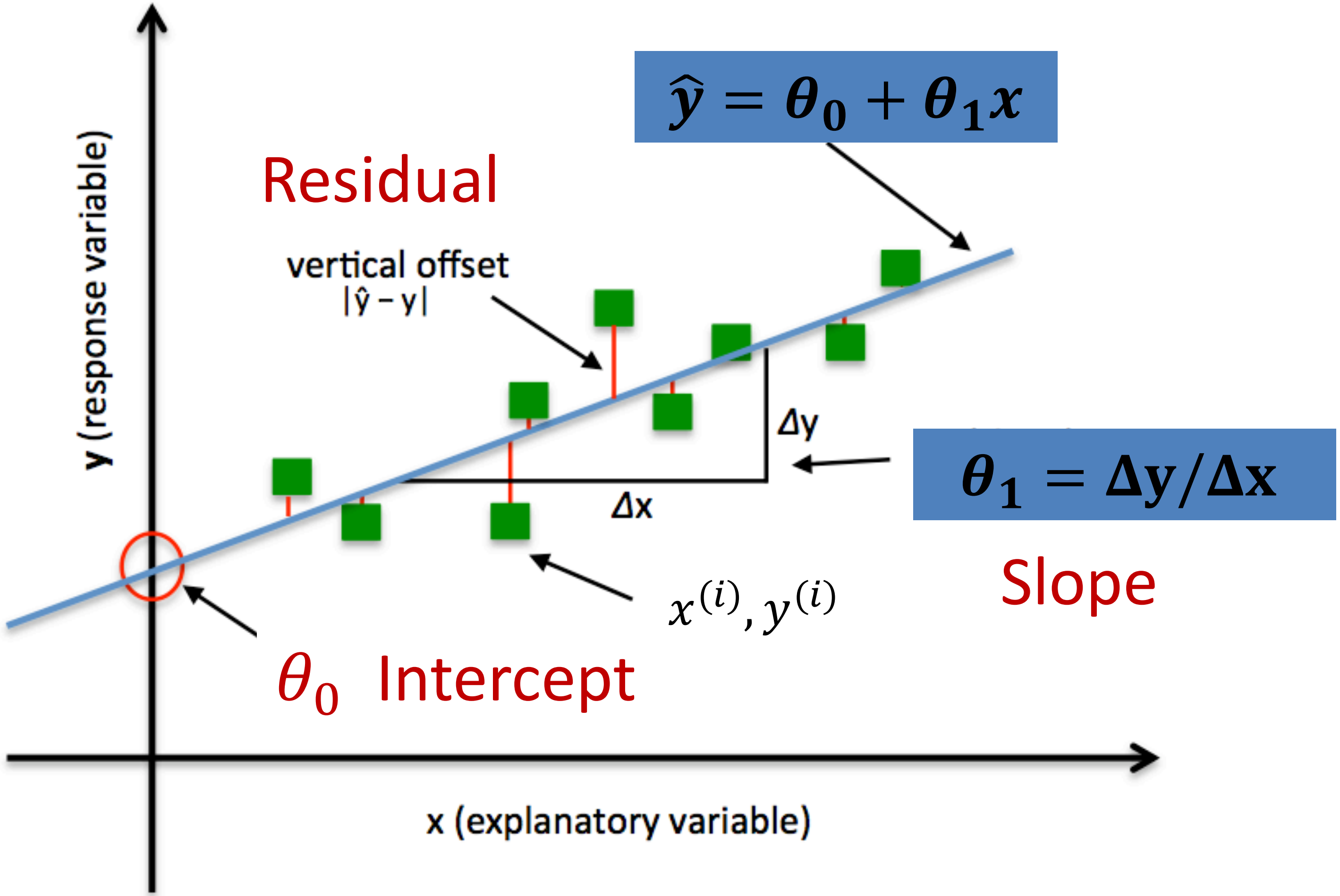
x'

$f(x)$

Response variable

Regression

Linear Regression



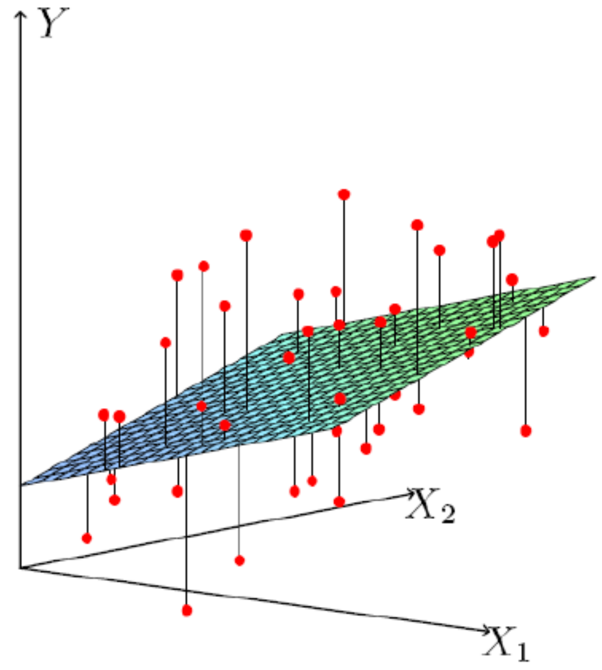
$$h_\theta(x) = \theta_0 + \theta_1 x$$

MSE$= \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2$

Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{N}\sum (\theta^T x_i - y_i)^2$ Loss / cost

$$\theta = (X^\top X)^{-1} X^\top y$$



Maximum Likelihood Estimation (MLE)

Given training data $X = \{x_1, \dots, x_N\}$ with labels $\mathrm{Y} = \{y_1, \dots, y_N\}$

What is the likelihood of training data for parameter θ?

Define likelihood function

$$Max_\theta\, L(\theta) = P[Y|X;\theta] = f(y_1, \dots, y_N | x_1, \dots, x_N; \theta)$$

Assumption: training points are independent!

$$L(\theta) = \prod_{i=1}^{N} P[y_i | x_i; \theta]$$

MLE for Linear Regression

$$L(\theta) = \prod_{i=1}^{N} P[y_i|x_i;\theta] = \prod_{i=1}^{N} f(y_i|x_i;\theta,\sigma)$$

$$\log L(\theta) = -\mathrm{c}\sum_{i=1}^{N}[y_i - (\theta_0 + \theta_1 x_i)]^2$$

Max likelihood θ is the same as Min MSE θ!

The MSE metric has statistical motivation

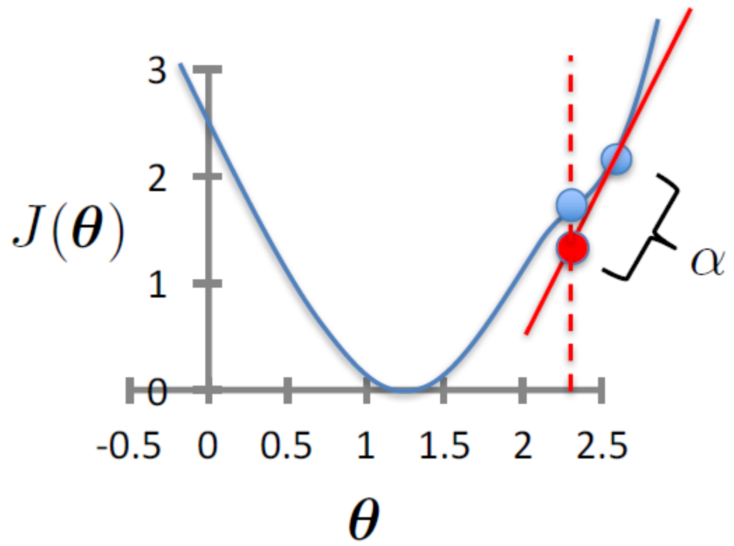
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05

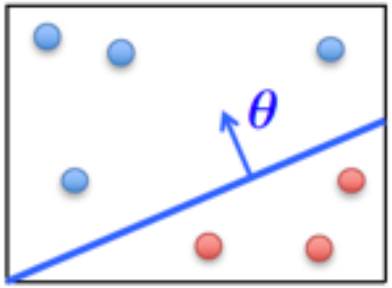


Gradient = slope of line tangent
to curve at the same point

Linear Classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$



$h_\theta(x) = f(\theta^T x)$ linear function

- If $\theta^T x > 0$ classify 1
- If $\theta^T x < 0$ classify 0

All the points x on the hyperplane satisfy: $\theta^T x = 0$

The Perceptron

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\boxed{\theta_j \leftarrow \theta_j - \frac{1}{2}\underbrace{(h_\theta(x_i) - y_i)}_{\text{either 2 or -2}}x_{ij}}$$

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

The Perceptron

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\boxed{\theta_j \leftarrow \theta_j - \frac{1}{2}\underbrace{(h_\theta(x_i) - y_i)}_{\text{either 2 or -2}}x_{ij}}$$

- Re-write as $$\boxed{\theta_j \leftarrow \theta_j + y_i x_{ij}}$$ (only upon misclassification)

Perceptron Rule: If x_i is misclassified, do

$$\theta \leftarrow \theta + y_i\, x_i$$

Online Perceptron

Let $\theta \leftarrow [0,0,\ldots,0]$
Repeat:
  Receive training example (x_i, y_i)
  If $y_i \theta^T x_i \leq 0$ // prediction is incorrect
    $\theta \leftarrow \theta + y_i\, x_i$

Online learning – the learning mode where the model update is performed each time a single observation is received

Batch learning – the learning mode where the model update is performed after observing the entire training set

Batch Perceptron

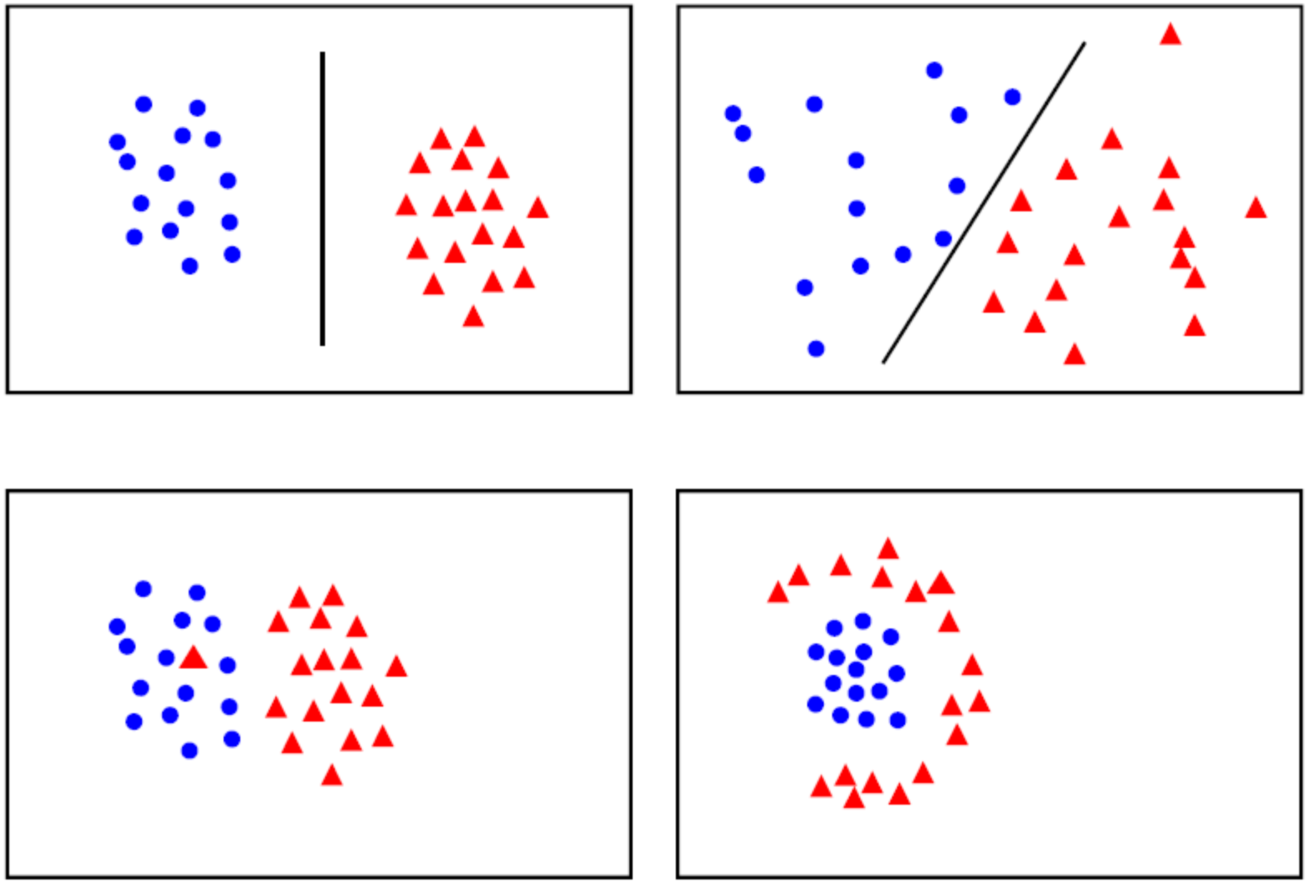
Given training data $\{(x_i, y_i)\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
  Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
  for $i = 1 \ldots n$, do
    if $y_i \theta^T x_i \leq 0$ // prediction for i^{th} instance is incorrect
      $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y_i x_i$
  $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$ // compute average update
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$

Guaranteed to find separating hyperplane if data is linearly separable

Linear separability

linearly separable

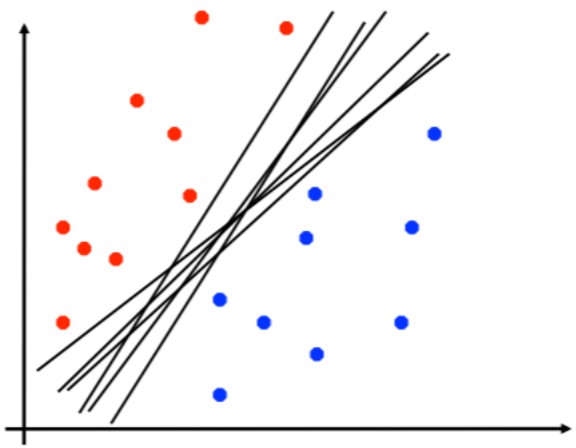
not linearly separable



- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  - Move the decision boundary for every example

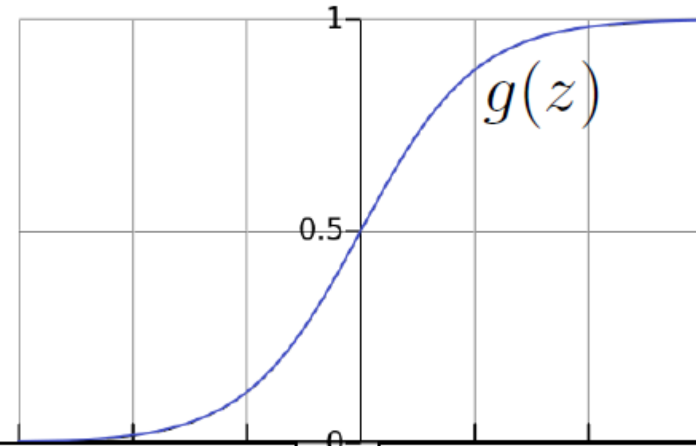


Which of this is optimal?

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

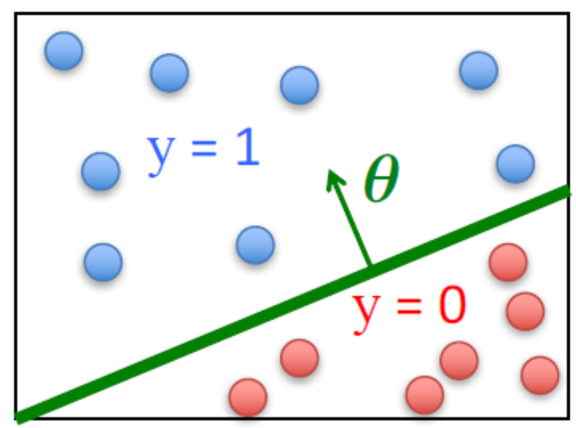
$$g(z) = \frac{1}{1+e^{-z}}$$



$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large positive values for positive instances

- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Logistic Regression is a linear classifier!

LDA

- Classify to one of k classes
- Logistic regression computes directly
  - $\mathrm{P}[Y = 1|X = x]$
  - Assume sigmoid function
- LDA uses Bayes Theorem to estimate it
  - $\mathrm{P}[Y = k|X = x] = \frac{\mathrm{P}[X = x|Y = k]\mathrm{P}[Y=k]}{\mathrm{P}[X=x]}$
  - Let $\pi_k = \mathrm{P}[Y = k]$ be the prior probability of class k and $f_k(x) = \mathrm{P}[X = x|Y = k]$

LDA

$$\Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2 \right)$$

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Assumption: $\sigma_1 = \ \ldots \sigma_k = \ \sigma$

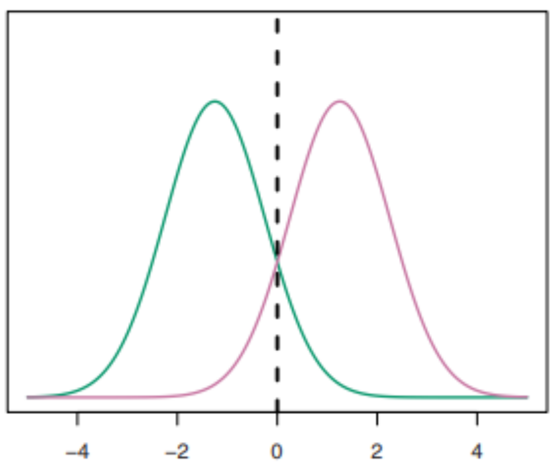
LDA decision boundary

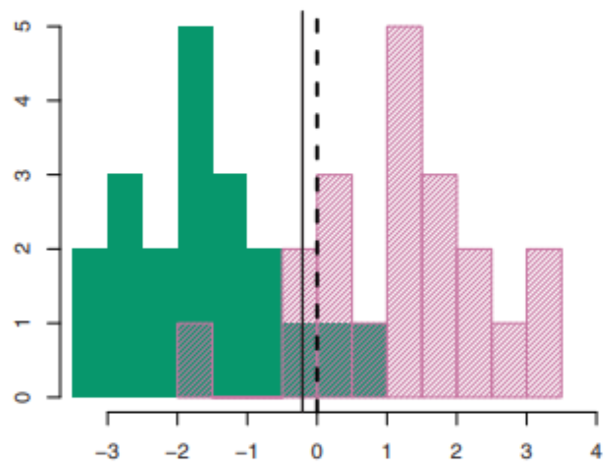
Pick class k to maximize

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2}$





True decision boundary

Estimated decision boundary

LDA

Given training data $(x_i, y_i), i = 1, \ldots, N, y_i \in \{1, \ldots, K\}$

1. Estimate mean and variance

$$\boxed{\begin{aligned} \hat{\mu}_k &= \frac{1}{n_k} \sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \end{aligned}}$$

2. Estimate prior

$$\boxed{\hat{\pi}_k = n_k / n.}$$

Given testing point x, predict k that maximizes:

$$\boxed{\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)}$$

Multi-variate LDA

Given training data $(x_i, y_i), i = 1, \ldots, n, y_i \in \{1, \ldots, K\}$

1. Estimate mean and variance

- $\hat{\pi}_k = N_k/N$, where N_k is the number of class-k observations;
- $\hat{\mu}_k = \sum_{g_i=k} x_i/N_k$;
- $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T/(N - K)$.

2. Estimate prior

Given testing point x, predict k that maximizes:

$$\boxed{\delta_k(x) = x^T \mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T \mathbf{\Sigma}^{-1}\mu_k + \log \pi_k}$$

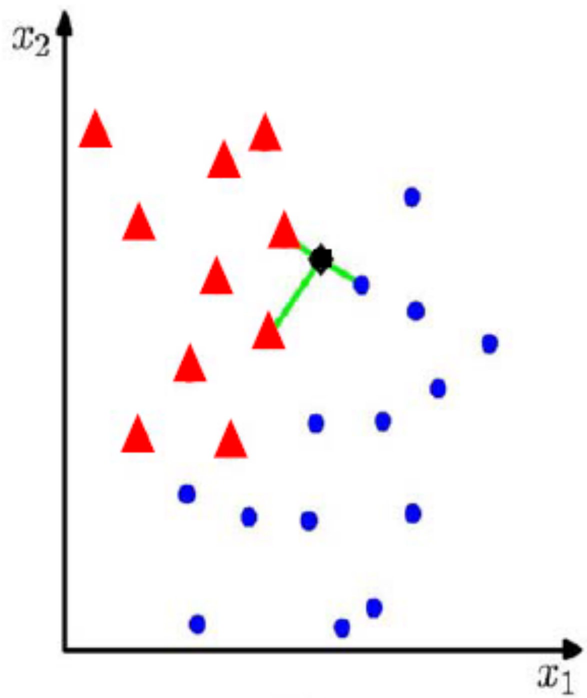
K Nearest Neighbour (K-NN) Classifier

Algorithm

- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

e.g. K = 3

- applicable to multi-class case



Naïve Bayes Classifier

Idea: Use the training data to estimate

$$P(X \mid Y) \text{ and } P(Y) \; .$$

Then, use Bayes rule to infer $P(Y|X_{\text{new}})$ for new data

Easy to estimate from data

Impractical, but necessary

$$\mathrm{P}[Y = k | X = x] = \frac{\boxed{\mathrm{P}[Y = k]}\boxed{\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k]}}{\boxed{\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}}$$

Unnecessary, as it turns out

- Recall that estimating the joint probability distribution $P(X_1, X_2, \ldots, X_d \mid Y)$ is not practical

Confusion Matrix

- Given a dataset of P positive instances and N negative instances:

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

- Imagine using classifier to identify positive cases (i.e., for information retrieval)

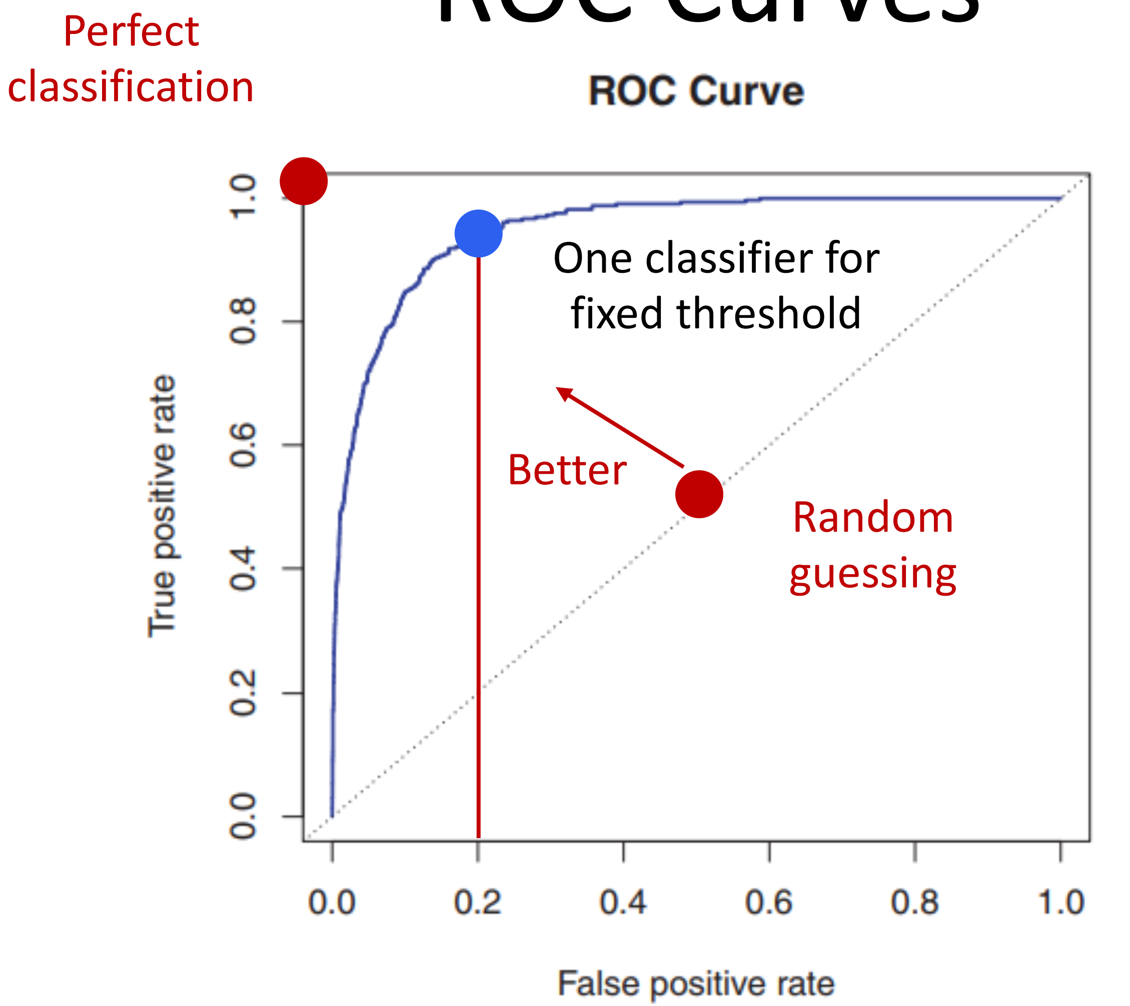
$$\text{precision} = \frac{TP}{TP + FP}$$

Probability that classifier predicts positive correctly

$$\text{recall} = \frac{TP}{TP + FN}$$

Probability that actual class is predicted correctly

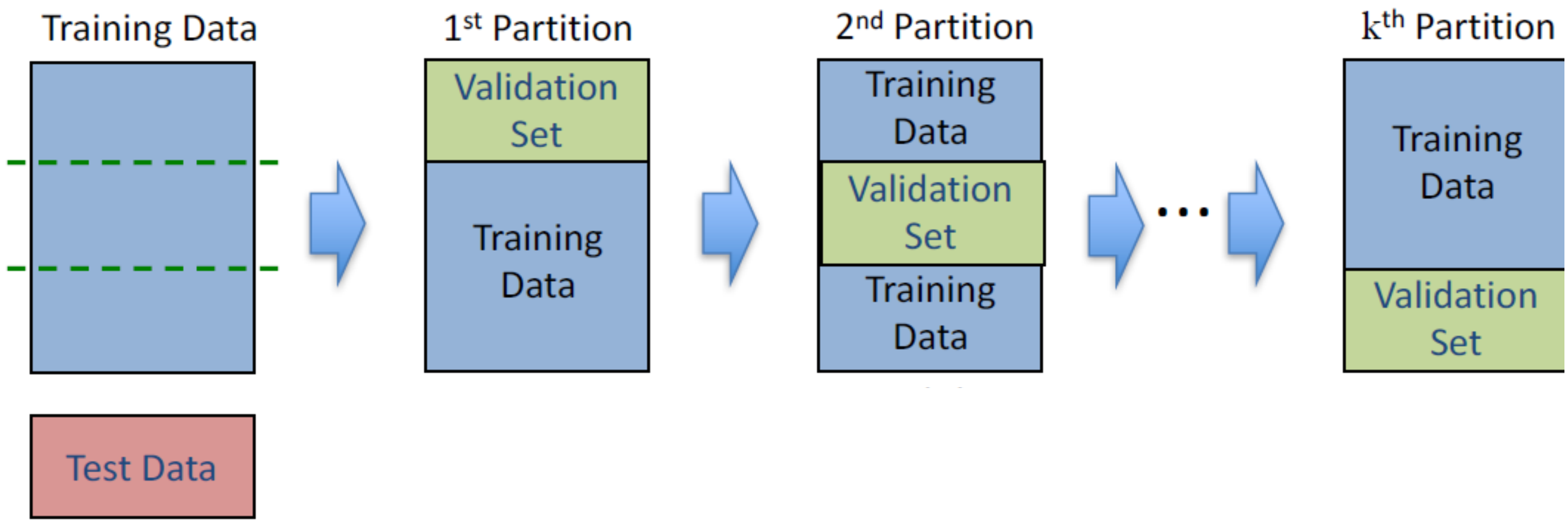
ROC Curves



- Receiver Operating Characteristic (ROC)
- Determine operating point (e.g., by fixing false positive rate)

29
29

Cross Validation

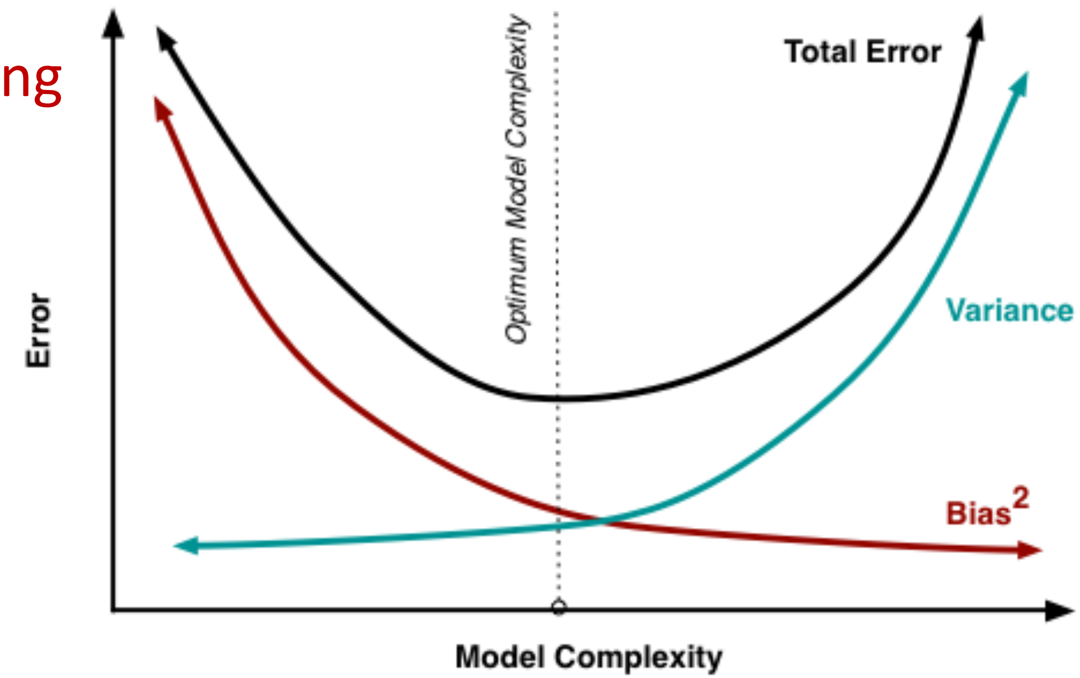


- k-fold CV
  – Split training data into k partitions (folds) of equal size
  – Pick the optimal value of hyper-parameter according to error metric averaged over all folds

Bias-Variance Tradeoff

Under-fitting

Over-fitting



- Bias = Difference between estimated and true models
- Variance = Model difference on different training sets

Regularization

- A method for controlling the complexity of learned hypothesis

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

Ridge

$$J(\theta) = \underbrace{\sum_{i=1}^{N}(h_\theta(x_i) - y_i)^2}_{\text{Squared Residuals}} + \lambda \underbrace{\sum_{j=1}^{d}|\theta_j|}_{\text{Regularization}}$$

LASSO

Type I: Conceptual

- Example 1: Describe difference between classification and regression
- Example 2: List one technique that can be used to improve model generality
- Example 3: Why do we need multiple metrics to evaluate classifiers
- Example 4: Provide advantages and disadvantages of:
  - Linear classifiers compared to more complex ones

More Examples

(a) **[3 points]**

Alice trains a classifier using a training dataset D and reports training error of 0.00001%. However, when Alice applies her classifier to testing data T, the error is 10.5%. What is the likely cause of Alice's problem?

Answer:

(b) **[3 points]** After taking DS 5220 describe some advice you would give Alice to solve her problem.

Answer:

Type II: Pseudocode

- Example 1: Write pseudocode for kNN
- Example 2: Write pseudocode for perceptron
- Example 3: Write pseudocode for …

Type III: Computational

- Example 1: Given a dataset, train a particular ML model
  - E.g., kNN, Naïve Bayes etc.
  - Evaluate model on some simple training and testing data
- Example 2: Given a dataset, compute some metrics / loss function
- Example 3: How many parameters does a model need to store?