# DS 4400

## Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 25 2018

# Review

- SVMs find optimal linear separator
- The kernel trick makes SVMs learn non-linear decision surfaces

- Strength of SVMs:
  - Good theoretical and empirical performance
  - Supports many types of kernels

- Disadvantages of SVMs:
  - "Slow" to train/predict for huge data sets (but relatively fast!)
  - Need to choose the kernel (and tune its parameters)

# Outline

- Naïve Bayes classifier
  - Density Estimation
- Application
  - Document classification
- Review of supervised learning methods
- Metrics for evaluating classifiers

# Prior and Joint Probabilities

- **Prior probability**: degree of belief without any other evidence
- **Joint probability**: matrix of combined probabilities of a set of variables

Russell & Norvig's Alarm Domain: (boolean RVs)

- A world has a specific instantiation of variables:

  $(\text{alarm} \wedge \text{burglary} \wedge \neg\text{earthquake})$

- The joint probability is given by:

$\text{P}(\text{Alarm}, \text{Burglary})$ =

| | alarm | ¬alarm |
|---|---|---|
| burglary | 0.09 | 0.01 |
| ¬burglary | 0.1 | 0.8 |

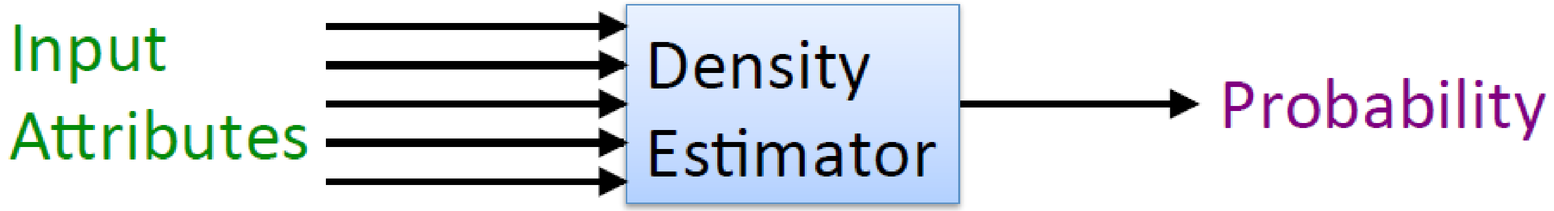
Prior probability of burglary:

$\text{P}(\text{Burglary}) = 0.1$

by marginalization over $\text{Alarm}$

# Density Estimation

- Our joint distribution learner is an example of something called **Density Estimation**
- A Density Estimator learns a mapping from a set of attributes to a probability



# Example – Learning Joint Probability Distribution

This Joint PD was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

| gender | hours_worked | wealth | |
|---|---|---|---|
| Female | v0:40.5- | poor | 0.253122 |
| | | rich | 0.0245895 |
| | v1:40.5+ | poor | 0.0421768 |
| | | rich | 0.0116293 |
| Male | v0:40.5- | poor | 0.331313 |
| | | rich | 0.0971295 |
| | v1:40.5+ | poor | 0.134106 |
| | | rich | 0.105933 |

# Pros and Cons of Density Estimators

- Pros
  - Density Estimators can learn distribution of training data
  - Can compute probability for a record
  - Can do inference (predict likelihood of record)
- Cons
  - Can overfit to the training data and not generalize to test data
  - Curse of dimensionality

Naïve Bayes classifier fixes these cons!

# Bayes' Rule

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

- Exactly the process we just used
- The most important formula in probabilistic machine learning

(Super Easy) Derivation:

$$\boxed{P(A \wedge B)} = P(A \mid B) \times P(B)$$
$$\boxed{P(B \wedge A)} = P(B \mid A) \times P(A)$$

these are the same

Just set equal...

$$P(A \mid B) \times P(B) = P(B \mid A) \times P(A)$$

and solve...

**Bayes, Thomas (1763)** An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, **53:370-418**

# Bayes' Rule

- Allows us to reason from **evidence** to **hypotheses**
- Another way of thinking about Bayes' rule:

$$P(\text{hypothesis} \mid \text{evidence}) = \frac{P(\text{evidence} \mid \text{hypothesis}) \times P(\text{hypothesis})}{P(\text{evidence})}$$

In the flu example:

$\mathrm{P}(\text{headache}) = 1/10$ $\mathrm{P}(\text{flu}) = 1/40$

$\mathrm{P}(\text{headache} \mid \text{flu}) = 1/2$

Given evidence of headache, what is $\mathrm{P}(\text{flu} \mid \text{headache})$ ?

Solve via Bayes rule!

# LDA

- Classify to one of k classes
- Logistic regression computes directly
  - $\mathrm{P}[Y = 1|X = x]$
  - Assume sigmoid function

Discriminative model

- 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 $f_k(x) = \mathrm{P}[X = x|Y = k]$

Generative model

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

# Classification Setting

- Recall Baye's Rule:

$$P(\text{hypothesis} \mid \text{evidence}) = \frac{P(\text{evidence} \mid \text{hypothesis}) \times P(\text{hypothesis})}{P(\text{evidence})}$$

- Equivalently, we can write:

$$\mathrm{P}[Y = k | X = x] = \frac{\mathrm{P}[Y = k]\,\mathrm{P}[X = x | Y = k]}{\mathrm{P}[X = x]}$$

where $X$ is a random variable representing the evidence and $Y$ is a random variable for the label

- 

# Naïve Bayes Classifier

**Idea:** Use the training data to estimate

$P(X \mid Y)$ and $P(Y)$ .

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

---

$$\mathrm{P}[Y = k|X = x] = \frac{\underbrace{\boxed{\mathrm{P}[Y = k]}}_{\text{Easy to estimate from data}}\ \underbrace{\boxed{\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d|Y = k]}}_{\text{Impractical, but necessary}}}{\underbrace{\boxed{\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}}_{\text{Unnecessary, as it turns out}}}$$

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

# Naïve Bayes Classifier

**Problem:** estimating the joint PD or CPD isn't practical

– Severely overfits, as we saw before

However, if we make the assumption that the attributes are independent given the class label, estimation is easy!

$$P(X_1, X_2, \ldots, X_d \mid Y) = \prod_{j=1}^{d} P(X_j \mid Y)$$

- In other words, we assume all attributes are *conditionally independent* given $Y$
- Often this assumption is violated in practice, but more on that later…

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| Sky | Temp | Humid | Wind | Water | Forecast | Play? |
|---|---|---|---|---|---|---|
| sunny | warm | normal | strong | warm | same | *yes* |
| sunny | warm | high | strong | warm | same | *yes* |
| rainy | cold | high | strong | warm | change | *no* |
| sunny | warm | high | strong | cool | change | *yes* |

$\mathrm{P(play)} = ?$

$\mathrm{P(Sky = sunny \mid play)} = ?$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = ?$

$\mathrm{P(Sky = sunny \mid \neg play)} = ?$

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| **Sky** | **Temp** | **Humid** | **Wind** | **Water** | **Forecast** | **Play?** |
|---|---|---|---|---|---|---|
| sunny | warm | normal | strong | warm | same | *yes* |
| sunny | warm | high | strong | warm | same | *yes* |
| rainy | cold | high | strong | warm | change | *no* |
| sunny | warm | high | strong | cool | change | *yes* |

$\mathrm{P}(\mathrm{play}) =$ 3/4

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \mathrm{play}) = ?$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \mathrm{play}) = ?$

$\ldots$

$\mathrm{P}(\neg\mathrm{play}) =$ 1/4

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \neg\mathrm{play}) = ?$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \neg\mathrm{play}) = ?$

$\ldots$

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| Sky | Temp | Humid | Wind | Water | Forecast | Play? |
|---|---|---|---|---|---|---|
| sunny | | | | | | *yes* |
| sunny | | | | | | *yes* |
| rainy | cold | high | strong | warm | change | *no* |
| sunny | | | | | | *yes* |

$\mathrm{P(play)} = 3/4$

$\mathrm{P(Sky = sunny \mid play)} = 1$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = 1/4$

$\mathrm{P(Sky = sunny \mid \neg play)} = ?$

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| Sky | Temp | Humid | Wind | Water | Forecast | Play? |
|---|---|---|---|---|---|---|
| sunny | warm | normal | strong | warm | same | *yes* |
| sunny | warm | high | strong | warm | same | *yes* |
| rainy | | | | | | *no* |
| sunny | warm | high | strong | cool | change | *yes* |

$\mathrm{P(play)} = 3/4$

$\mathrm{P(Sky = sunny \mid play)} = 1$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = 1/4$

$\mathrm{P(Sky = sunny \mid \neg play)} =$ 0

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| Sky | Temp | Humid | Wind | Water | Forecast | Play? |
|---|---|---|---|---|---|---|
| | | normal | | | | *yes* |
| | | high | | | | *yes* |
| rainy | cold | high | strong | warm | change | *no* |
| | | high | | | | *yes* |

$\mathrm{P}(\mathrm{play}) = 3/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \mathrm{play}) = 1$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \mathrm{play}) =$ 2/3

...

$\mathrm{P}(\neg\mathrm{play}) = 1/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \neg\mathrm{play}) = 0$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \neg\mathrm{play}) = ?$

...

# Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

| Sky | Temp | Humid | Wind | Water | Forecast | Play? |
|---|---|---|---|---|---|---|
| sunny | warm | normal | strong | warm | same | *yes* |
| sunny | warm | high | strong | warm | same | *yes* |
| | | high | | | | *no* |
| sunny | warm | high | strong | cool | change | *yes* |

$\mathrm{P}(\mathrm{play}) = 3/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \mathrm{play}) = 1$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \mathrm{play}) = 2/3$

$\ldots$

$\mathrm{P}(\neg\mathrm{play}) = 1/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \neg\mathrm{play}) = 0$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \neg\mathrm{play}) = 1$

$\ldots$

# Laplace Smoothing

- Notice that some probabilities estimated by counting might be zero
  - Possible overfitting!

- Fix by using Laplace smoothing:
  - Adds 1 to each count

$$P(X_j = v \mid Y = k\ ) = \frac{c_v + 1}{\sum_{v' \in \text{values}(X_j)} c_{v'} + |\text{values}(X_j)|}$$

where

- $c_v$ is the count of training instances with a value of $v$ for attribute $j$ and class label $k$
- $|\text{values}(X_j)|$ is the number of values $X_j$ can take on

# Using the Naïve Bayes Classifier

- Now, we have

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

This is constant for a given instance, and so irrelevant to our prediction

# Naïve Bayes Classifier

- For each class label $k$
  1. Estimate prior $P[Y = k]$ from the data
  2. For each value $v$ of attribute $X_j$
     - Estimate $\mathrm{P}[X_j = v | Y = k]$

- Classify a new point via:

$$h(\mathbf{x}) = \arg\max_{y_k} \log P(Y = k) + \sum_{j=1}^{d} \log P(X_j = x_j \mid Y = k)$$

# Computing Probabilities

- NB classifier gives predictions, not probabilities, because we ignore $\mathrm{P}(\mathrm{X})$ (the denominator in Bayes rule)
- Can produce probabilities by:
  - For each possible class label $y_k$, compute

$$\underbrace{\tilde{P}(Y = k \mid X = \mathbf{x})}_{\substack{\text{This is the numerator of Bayes rule, and is} \\ \text{therefore off the true probability by a factor} \\ \text{of } \alpha \text{ that makes probabilities sum to 1}}} = P(Y = k) \prod_{j=1}^{d} P(X_j = x_j \mid Y = k)$$

  - α is given by $\alpha = \dfrac{1}{\sum_{k=1}^{\#classes} \tilde{P}(Y = k \mid X = \mathbf{x})}$
  - Class probability is given by

$$P(Y = k \mid X = \mathbf{x}) = \alpha \tilde{P}(Y = k \mid X = \mathbf{x})$$

# Naïve Bayes Summary

**Advantages:**

- Fast to train (single scan through data)
- Fast to classify
- Not sensitive to irrelevant features
- Handles real and discrete data
- Handles streaming data well

**Disadvantages:**

- Assumes independence of features

# Document Classification

**PROBLEM SETTING**

**Given:**

- Representation of a document
- Set of classes 1,…,$K$

(AI) (Programming) (HCI)

***Classes:***

| ML | Planning | Semantics | Garb.Coll. | Multimedia | GUI |
|---|---|---|---|---|---|
| learning | planning | programming | garbage | … | … |
| intelligence | temporal | semantics | collection | | |
| algorithm | reasoning | language | memory | | |
| reinforcement | plan | proof… | optimization | | |
| network… | language… | | region… | | |

***Training Data:*** (rows above)

# Document Classification

***Test Data:***

"planning
language
proof
intelligence"

**PROBLEM SETTING**

**Given:**
- Representation of a document
- Set of classes 1,...,$K$

**Determine:**
- Class to which document $d$ belongs

(AI) (Programming) (HCI)

***Classes:***

ML | Planning | Semantics | Garb.Coll. | Multimedia | GUI

***Training Data:***

| ML | Planning | Semantics | Garb.Coll. | Multimedia | GUI |
|---|---|---|---|---|---|
| learning | <u>planning</u> | programming | garbage | ... | ... |
| <u>intelligence</u> | temporal | semantics | collection | | |
| algorithm | reasoning | <u>language</u> | memory | | |
| reinforcement | plan | <u>proof</u>... | optimization | | |
| network... | language... | | region... | | |

# Text Classification: Examples

- Classify news stories as *World, US, Business, SciTech, Sports, etc.*
- Add terms to Medline abstracts (e.g. "Conscious Sedation" [E03.250])
- Classify business names by industry
- Classify student essays as *A/B/C/D/F*
- Classify email as *Spam/Other*
- Classify email to tech staff as *Mac/Windows/ ...*
- Classify pdf files as *ResearchPaper/Other*
- Determine authorship of documents
- Classify movie reviews as *Favorable/Unfavorable/Neutral*
- Classify technical papers as *Interesting/Uninteresting*
- Classify jokes as *Funny/NotFunny*
- Classify websites of companies by Standard Industrial Classification (SIC) code

# Bag of Words Representation

What is the ~~best~~ representation for documents?

simplest, yet useful

**Idea:** Treat each document as a sequence of words

- Assume that word positions are generated *independently*

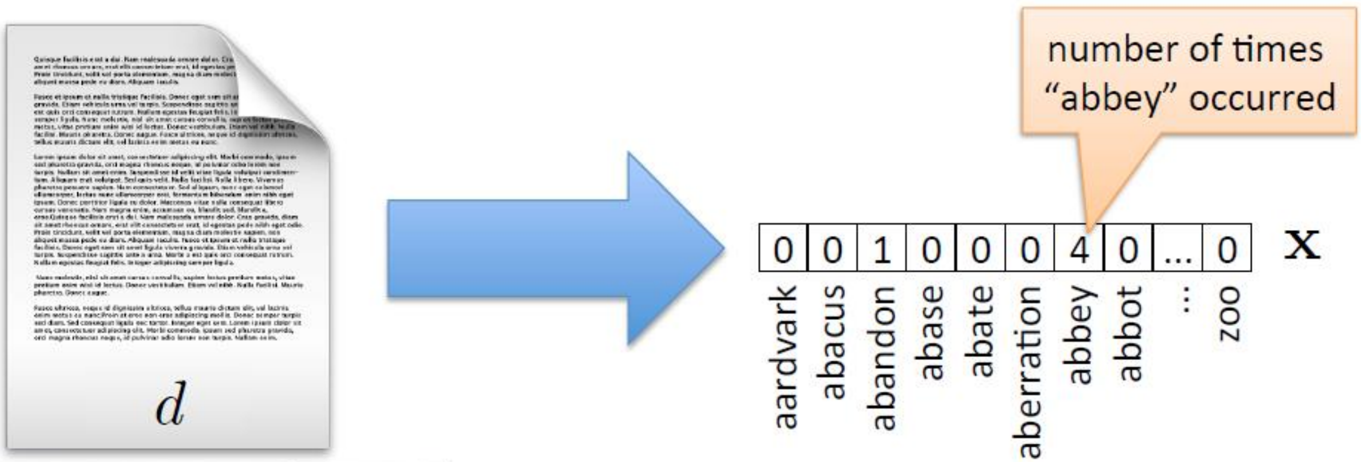
<u>Dictionary</u>: set of all possible words

- Compute over set of documents
- Use Webster's dictionary, etc.

# Bag of Words Representation

Represent document $d$ as a vector of word counts $\mathbf{x}$

- $x_j$ represents the count of word $j$ in the document
  - $\mathbf{x}$ is sparse (few non-zero entries)




# Another View of Naïve Bayes

- Let the model parameters for class $c$ be given by:

$$\boldsymbol{\theta}_c = \{\theta_{c1}, \theta_{c2}, \ldots, \theta_{c\underbrace{|D|}_{\text{size of dictionary } D}}\}$$

  - $\theta_{cj} = \mathrm{P}(\text{word } j \text{ occurs in a document from } c)$
  - Also have that $\sum_j \theta_{cj} = 1$

- The likelihood of a document $d$ characterized by $\mathbf{x}$ is

$$P(d \mid \boldsymbol{\theta}_c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

  - This is just the multinomial distribution, a generalization of the binomial distribution $\binom{n}{k} p^k (1-p)^{n-k}$

# Another View of Naïve Bayes

- The likelihood of a document $d$ characterized by $\mathbf{x}$ is

$$P(d \mid \boldsymbol{\theta}_c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

- Use Bayes rule:

introduce class priors

$$\log P(\boldsymbol{\theta}_c \mid d) \propto \log \left( P(\boldsymbol{\theta}_c) \prod_{j=1}^{|D|} (\theta_{cj})^{x_j} \right) = \log P(\boldsymbol{\theta}_c) + \sum_{j=1}^{|D|} x_j \log \theta_{cj}$$

# Document Classification with Naïve Bayes

1. Compute dictionary $D$ over training set (if not given)
2. Represent training documents as bags of words over $D$
3. Estimate class priors via counting
4. Estimate conditional probabilities as $\hat{\theta}_{cj} = \dfrac{N_{cj} + 1}{N_c + |D|}$
   - $N_{cj}$ is number of times word $j$ occurs in documents from class $c$
   - $N_c$ is total number of words in all documents from class $c$

- Naïve Bayes model for new documents (represented in $D$) is:

$$h(d) = \arg\max_c \left( \log P(c) + \sum_j x_j \hat{w}_{cj} \right)$$

$$\text{where } \hat{w}_{cj} = \log \hat{\theta}_{cj}$$

# Review Naïve Bayes

- Density Estimators can estimate joint probability distribution from data
- Risk of overfitting and curse of dimensionality
- Naïve Bayes assumes that features are independent given labels
  - Reduces the complexity of density estimation
  - Even though the assumption is not always true, Naïve Bayes works well in practice
- Applications: text classification with bag-of-words representation
  - Naïve Bayes becomes a linear classifier

Generative model

# Traditional learning

- Linear classifiers
  - Logistic regression, LDA, perceptrons
- Decision trees
- Ensembles
  - Random Forests
  - AdaBoost
- SVM
  - Linear SVM
  - Kernels
- Naïve Bayes

# 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 |

# Accuracy and Error

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

| Actual Class | | Predicted Class: Yes | Predicted Class: No |
|---|---|---|---|
| | Yes | TP | FN |
| | No | FP | TN |

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

| Actual Class | | Predicted Class: Yes | Predicted Class: No |
|---|---|---|---|
| | Yes | TP | FN |
| | No | FP | TN |

$$\text{error} = 1 - \frac{TP + TN}{P + N} = \frac{FP + FN}{P + N}$$

# Precision & Recall

**Precision**

- the fraction of positive predictions that are correct
- P(is pos | predicted pos)

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

**Recall**

- fraction of positive instances that are identified
- P(predicted pos | is pos)

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

---

- You can get high recall (but low precision) by only predicting positive
- Recall is a non-decreasing function of the # positive predictions
- Typically, precision decreases as either the number of positive predictions or recall increases
- Precision & recall are widely used in information retrieval

# F-Score

- Combined measure of precision/recall tradeoff

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - This is the harmonic mean of precision and recall
  - In the $F_1$ measure, precision and recall are weighted evenly
  - Can also have biased weightings that emphasize either precision or recall more ($F_2$ = 2 × recall; $F_{0.5}$ = 2 × precision)

- Limitations:
  - F-measure can exaggerate performance if balance between precision and recall is incorrect for application
    - Don't typically know balance ahead of time

# A Word of Caution

- Consider binary classifiers A, B, C:

| | | A | . | B | . | C | . |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | 1 | 0 | 1 | 0 |
| Predictions | 1 | 0.9 | 0.1 | 0.8 | 0 | 0.78 | 0 |
| | 0 | 0 | 0 | 0.1 | 0.1 | 0.12 | 0.1 |

- Clearly A is useless, since it always predicts 1
- B is slightly better than C
  - less probability mass wasted on the off-diagonals
- But, here are the performance metrics:

| Metric | A | B | C |
|---|---|---|---|
| Accuracy | 0.9 | 0.9 | 0.88 |
| Precision | 0.9 | 1.0 | 1.0 |
| Recall | 1.0 | 0.888 | 0.8667 |
| F-score | 0.947 | 0.941 | 0.9286 |

# Receiver Operating Characteristic (ROC)

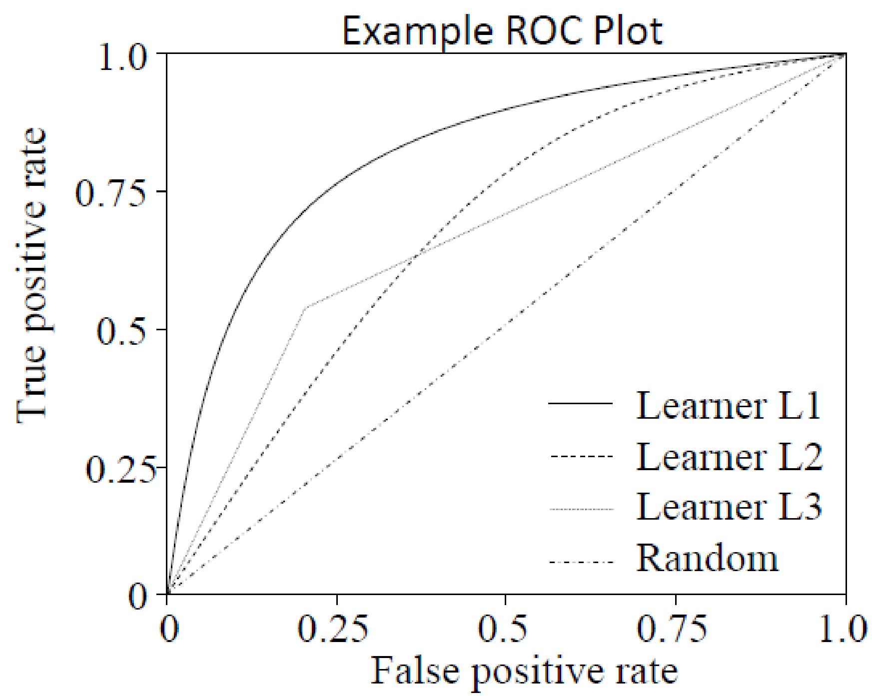
ROC curves assess predictive behavior independent of error costs or class distributions

- Originated from signal detection theory
- Common in medical diagnosis, now used for ML

**Plots TP rate vs FP Rate**

$$\text{TP rate} = \text{TP}/\text{P}$$
$$\text{FP rate} = \text{FP}/\text{N}$$

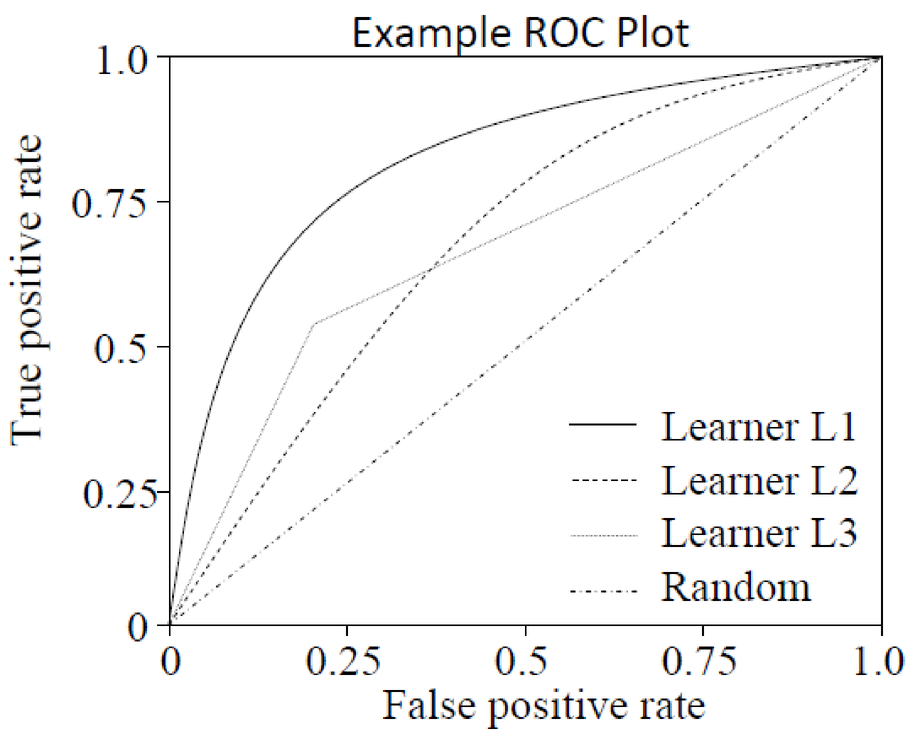
| | | Predicted Class | |
|---|---|---|---|
| | | Yes | No |
| Actual Class | Yes | TP | FN |
| | No | FP | TN |



# Performance Depends on Threshold

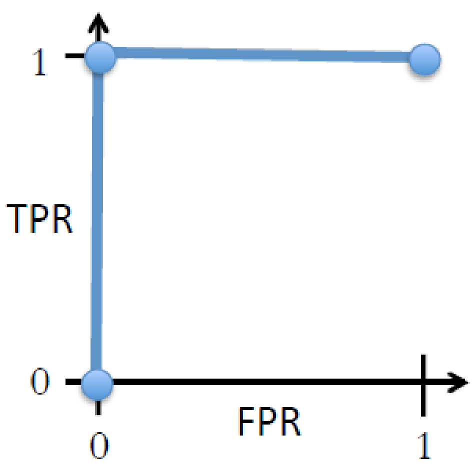
Predict positive if $\mathrm{P}(y = 1 \mid \mathbf{x}) > \theta$, otherwise negative

- Number of TPs and FPs depend on threshold $\theta$
- As we vary $\theta$, we get different (TPR, FPR) points



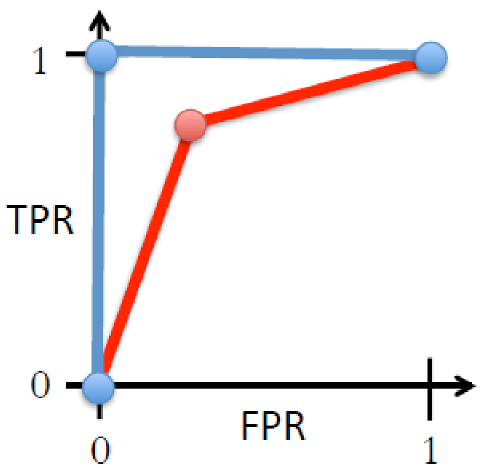
# ROC Example

| $i$ | $y_i$ | $p(y_i = 1 \mid \mathbf{x}_i)$ | $h(\mathbf{x_i} \mid \theta = 0)$ | $h(\mathbf{x_i} \mid \theta = 0.5)$ | $h(\mathbf{x_i} \mid \theta = 1)$ |
|---|---|---|---|---|---|
| 1 | 1 | 0.9 | 1 | 1 | 0 |
| 2 | 1 | 0.8 | 1 | 1 | 0 |
| 3 | 1 | 0.7 | 1 | 1 | 0 |
| 4 | 1 | 0.6 | 1 | 1 | 0 |
| 5 | 1 | 0.5 | 1 | 1 | 0 |
| 6 | 0 | 0.4 | 1 | 0 | 0 |
| 7 | 0 | 0.3 | 1 | 0 | 0 |
| 8 | 0 | 0.2 | 1 | 0 | 0 |
| 9 | 0 | 0.1 | 1 | 0 | 0 |
| | | | $TPR = 5/5 = 1$ <br> $FPR = 4/4 = 1$ | $TPR = 5/5 = 1$ <br> $FPR = 0/4 = 0$ | $TPR = 0/5 = 0$ <br> $FPR = 0/4 = 0$ |

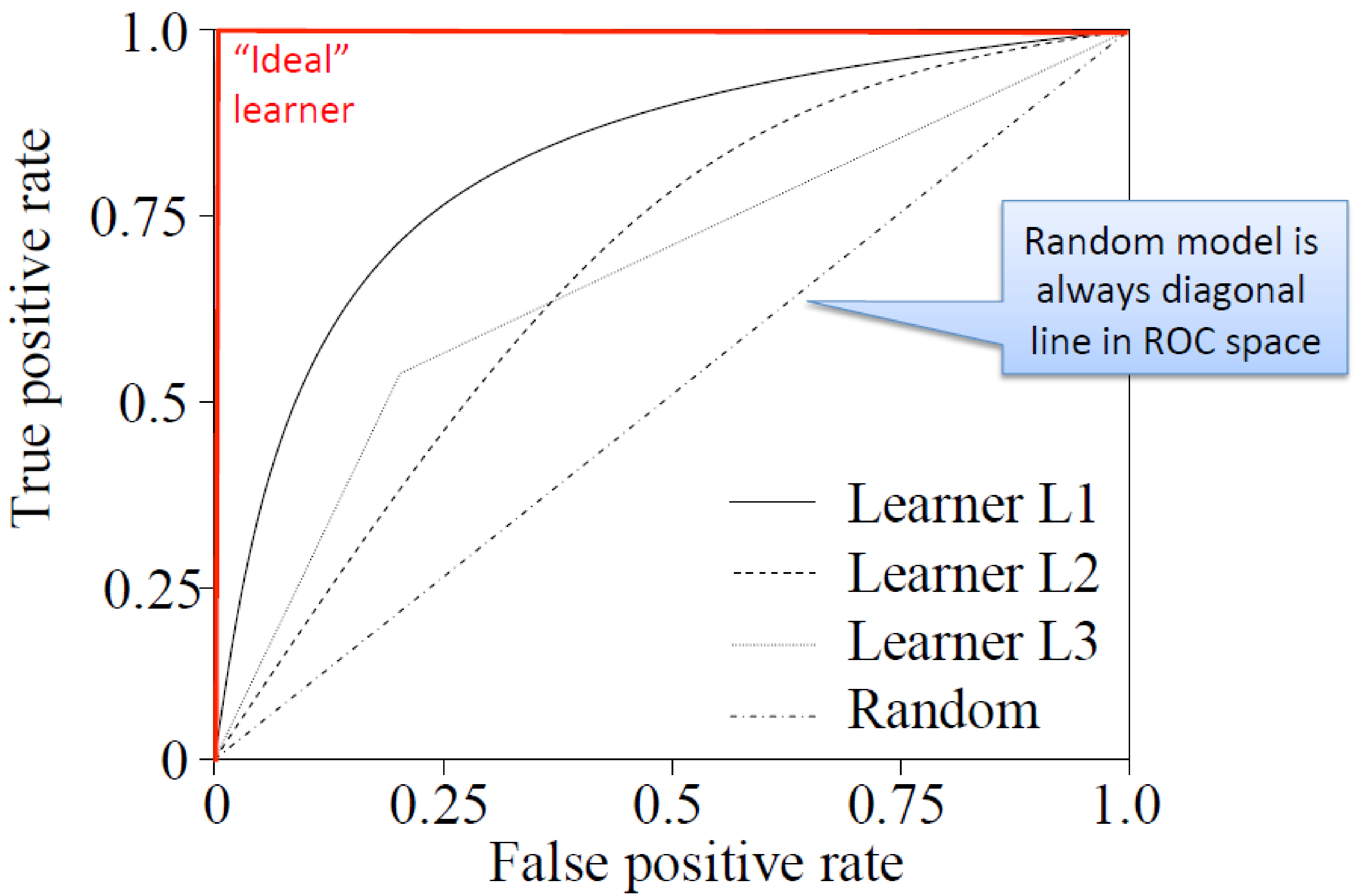


# ROC Example

| $i$ | $y_i$ | $p(y_i = 1 \mid \mathbf{x}_i)$ | $h(\mathbf{x_i} \mid \theta = 0)$ | $h(\mathbf{x_i} \mid \theta = 0.5)$ | $h(\mathbf{x_i} \mid \theta = 1)$ |
|---|---|---|---|---|---|
| 1 | 1 | 0.9 | 1 | 1 | 0 |
| 2 | 1 | 0.8 | 1 | 1 | 0 |
| 3 | 1 | 0.7 | 1 | 1 | 0 |
| 4 | 1 | 0.6 | 1 | 1 | 0 |
| 5 | 1 | **0.2** | 1 | **0** | 0 |
| 6 | 0 | **0.6** | 1 | **1** | 0 |
| 7 | 0 | 0.3 | 1 | 0 | 0 |
| 8 | 0 | 0.2 | 1 | 0 | 0 |
| 9 | 0 | 0.1 | 1 | 0 | 0 |
| | | | $TPR = 5/5 = 1$ | $TPR = 4/5 = 0.8$ | $TPR = 0/5 = 0$ |
| | | | $FPR = 4/4 = 1$ | $FPR = 1/4 = 0.25$ | $FPR = 0/4 = 0$ |

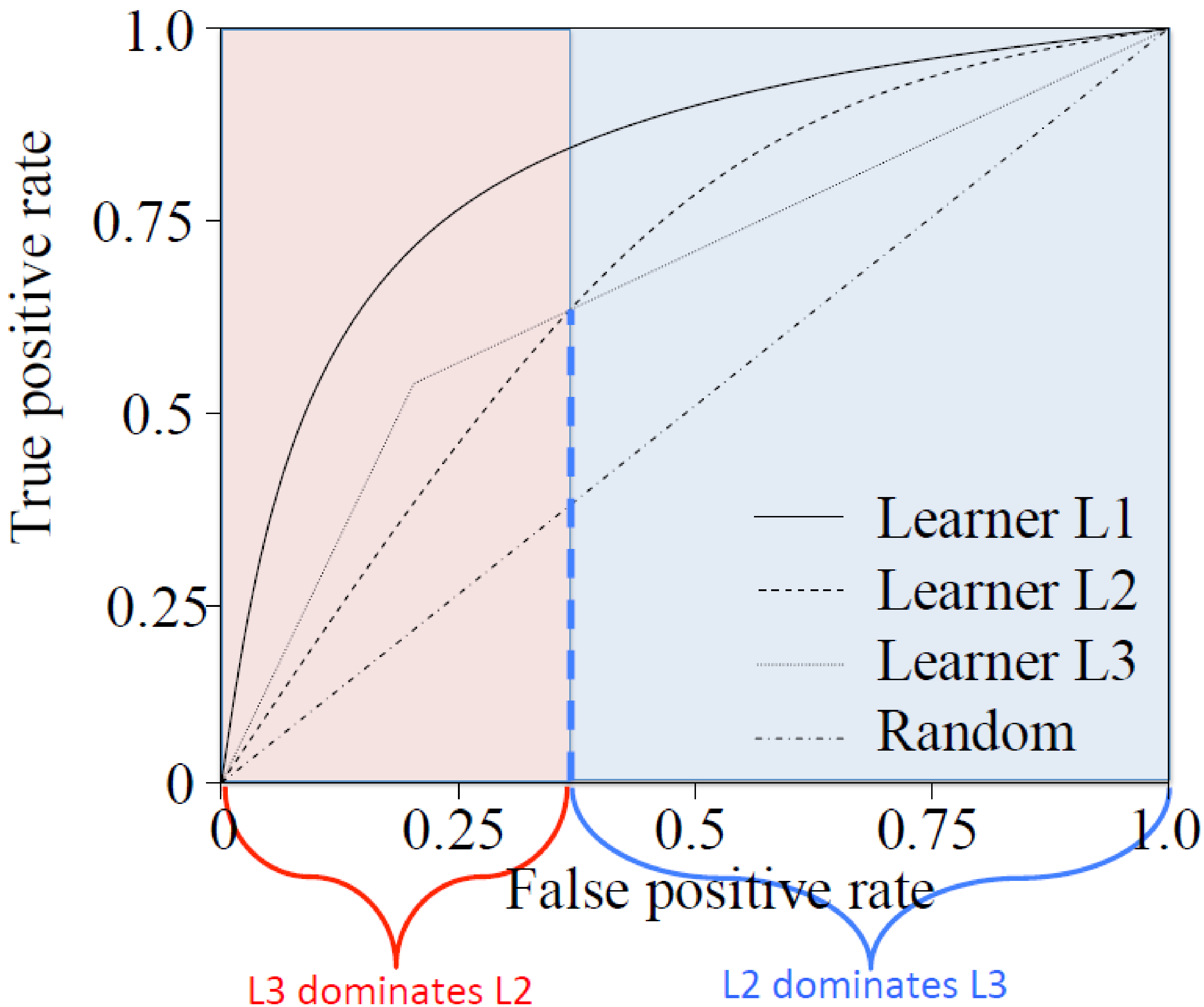


# ROC Curve

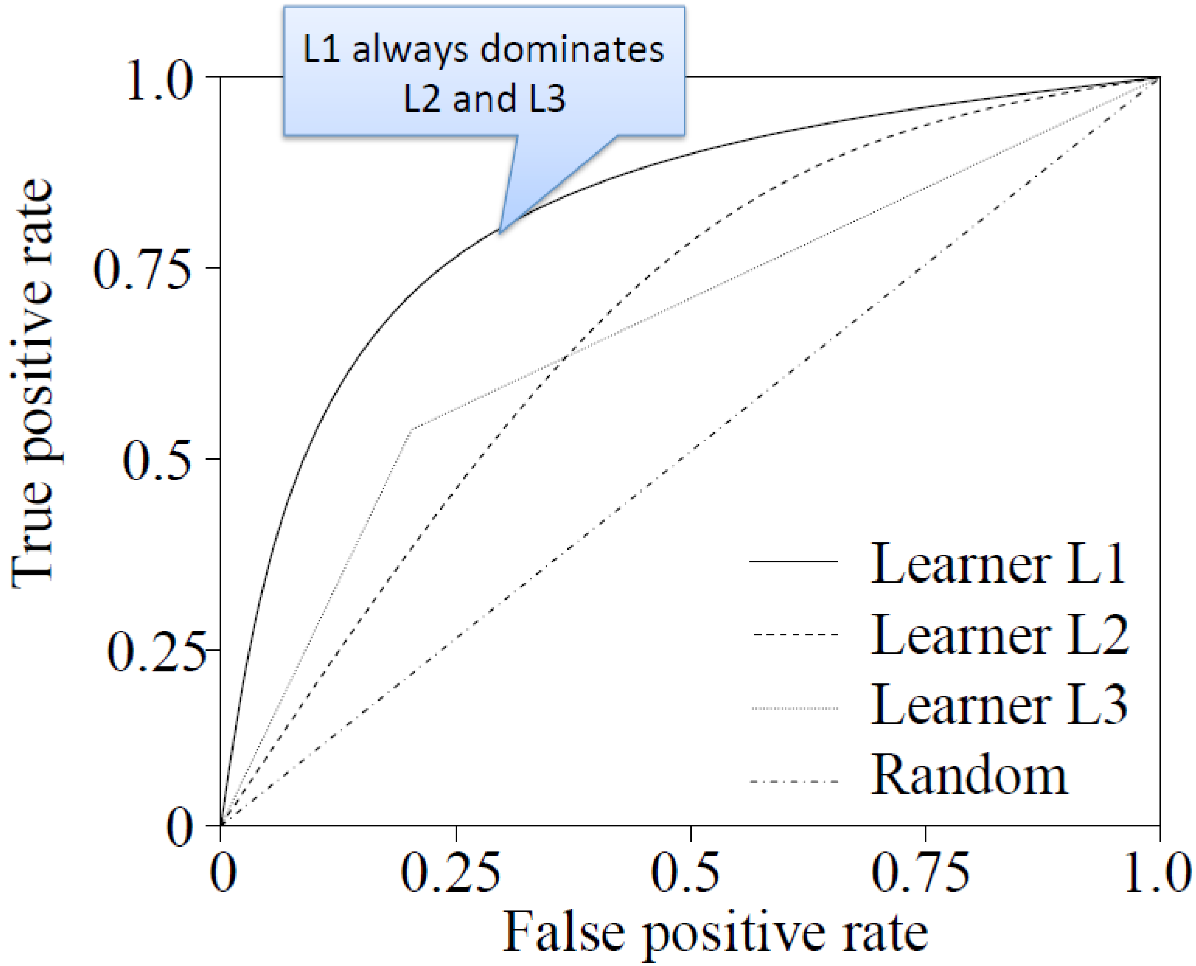


"Ideal"
learner
Random model is always diagonal line in ROC space
True positive rate
False positive rate
1.0
0.75
0.5
0.25
0
0
0.25
0.5
0.75
1.0
Learner L1
Learner L2
Learner L3
Random

# ROC Curve


1.0
0.75
0.5
0.25
0
True positive rate
0
0.25
0.5
0.75
1.0
False positive rate
Learner L1
Learner L2
Learner L3
Random
L3 dominates L2
L2 dominates L3

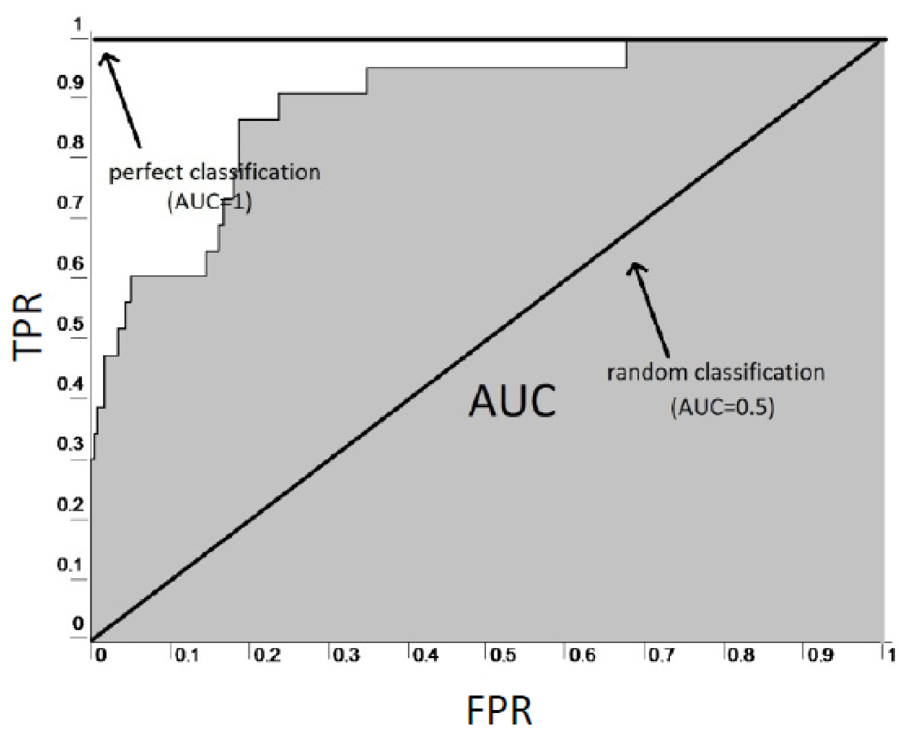
# ROC Curve

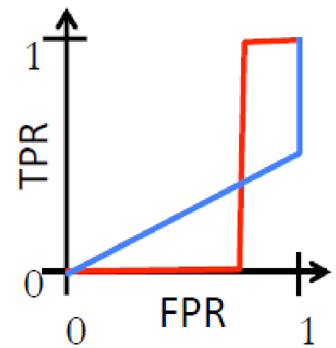


L1 always dominates L2 and L3
1.0
0.75
0.5
0.25
0
True positive rate
0
0.25
0.5
0.75
1.0
False positive rate
Learner L1
Learner L2
Learner L3
Random

# Area Under the ROC Curve

- Can take area under the ROC curve to summarize performance as a single number
  - Be cautious when you see only AUC reported without a ROC curve; AUC can hide performance issues




Same AUC, very different performance

# Comparing Supervised Learning

Comparing Supervised Learning Algorithms : Table

| Algorithm | Problem Type | Results interpretable by you? | Easy to explain algorithm to others? | Average predictive accuracy | Training speed | Prediction speed |
|---|---|---|---|---|---|---|
| KNN | Either | Yes | Yes | Lower | Fast | Depends on n |
| Linear regression | Regression | Yes | Yes | Lower | Fast | Fast |
| Logistic regression | Classification | Somewhat | Somewhat | Lower | Fast | Fast |
| Naive Bayes | Classification | Somewhat | Somewhat | Lower | Fast (excluding feature extraction) | Fast |
| Decision trees | Either | Somewhat | Somewhat | Lower | Fast | Fast |
| Random Forests | Either | A little | No | Higher | Slow | Moderate |
| AdaBoost | Either | A little | No | Higher | Slow | Fast |
| Neural networks | Either | No | No | Higher | Slow | Fast |

# Acknowledgements


- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!