DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

February 5 2019

Logistic Regression

- Given $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}$
  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

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

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

Probabilistic Interpretation

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

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

$$Max_\theta\, L(\theta) = P[Y|X;\theta] = \prod_{i=1}^{n} P[y^{(i)}|x^{(i)};\theta]$$

Assumption: training points are independent

Find model parameter θ with Maximum Likelihood

General probabilistic method for classifier training

Gradient Descent for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)} \qquad \right]$$

Outline

- Evaluation of classifiers
  - Metrics
  - ROC curves
- Linear Discriminant Analysis (LDA)
- Lab (logistic regression, LDA, kNN)
- Feature selection
  - Wrapper
  - Filter
  - Embedded methods

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:

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

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

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

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

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

True Positive Rate

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

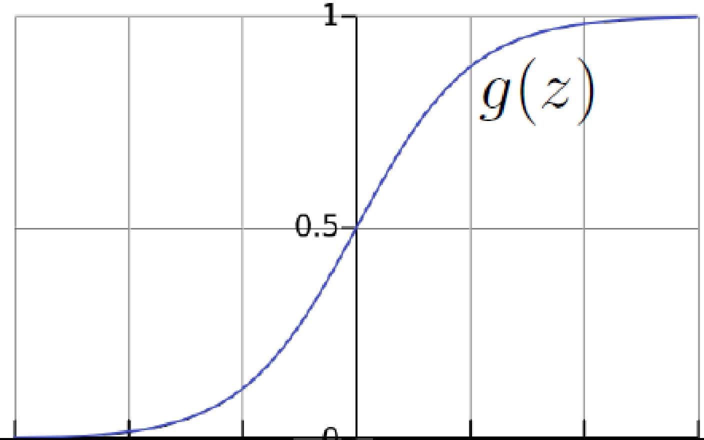
- Clearly A is useless, since it always predicts 1
- B is slightly better than C
  - less probability mass wasted on the off-diagonals
- But, here are the performance metrics:

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

Logistic Regression

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

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



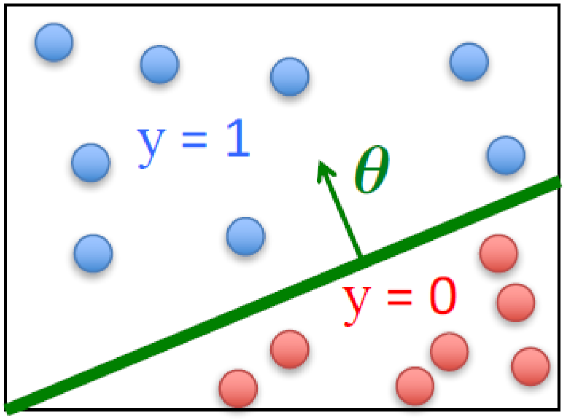
$\boldsymbol{\theta}^{\intercal}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^{\intercal}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

Probabilistic model $h_{\theta(x)} = \mathrm{P}[y = 1|x;\theta]$

– Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$

– Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Classifiers can be tuned

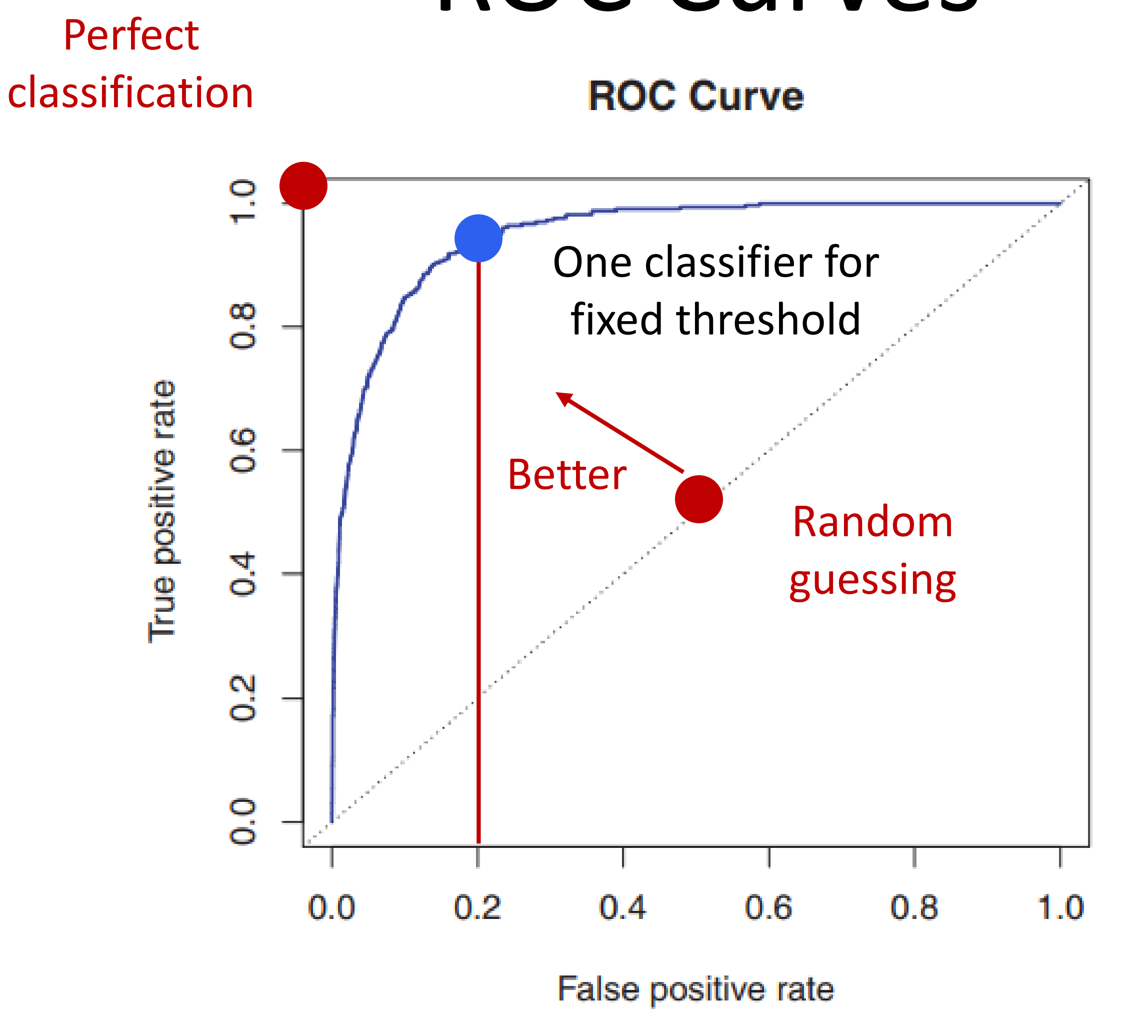
- Logistic regression sets by default the threshold at 0.5 for classifying positive and negative instances
- Some applications have strict constraints on false positives (or other metrics)
  - Example: very low false positives in security (spam)
- Solution: choose different threshold

Probabilistic model $h_{\theta(x)} = \mathrm{P}[y = 1|x; \theta]$

- Predict y = 1 if $h_\theta(x) \geq$ T
- Predict y = 0 if $h_\theta(x) <$ T

Higher T, lower FP
Lower T, lower FN

ROC Curves

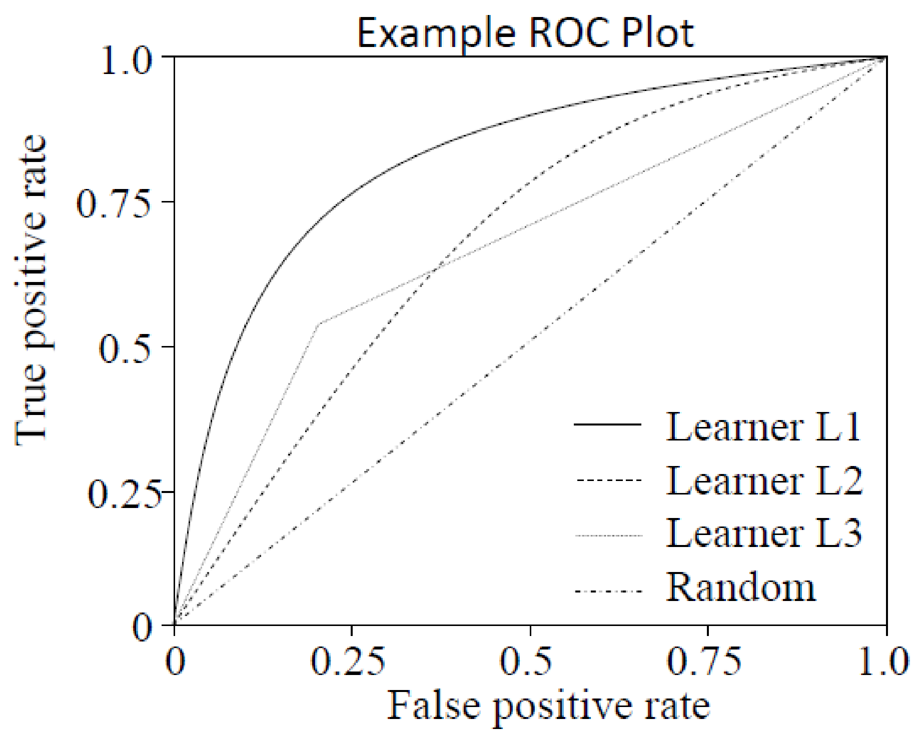


- Receiver Operating Characteristic (ROC)
- Determine operating point (e.g., by fixing false positive rate)

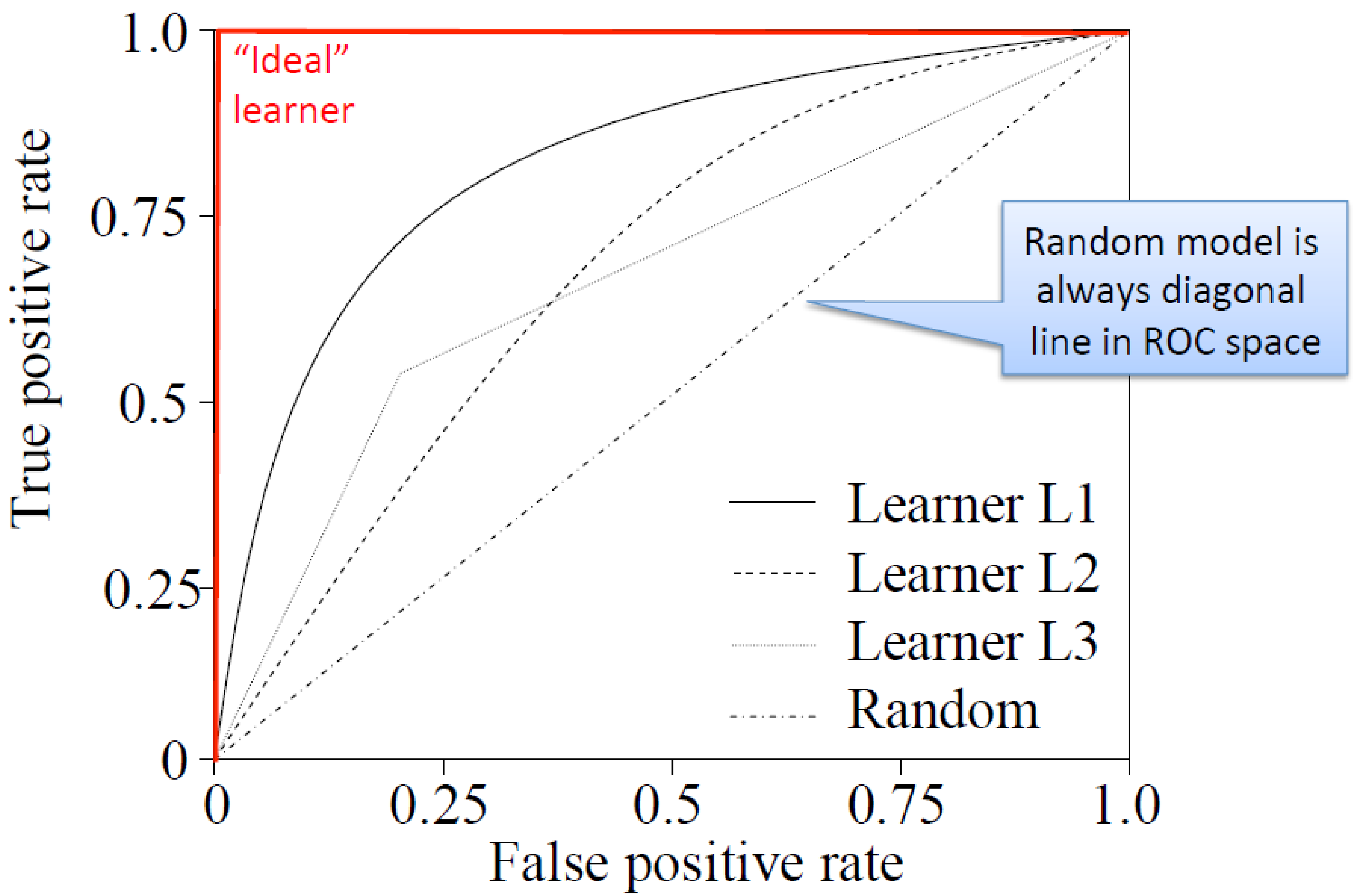
Performance Depends on Threshold

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

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

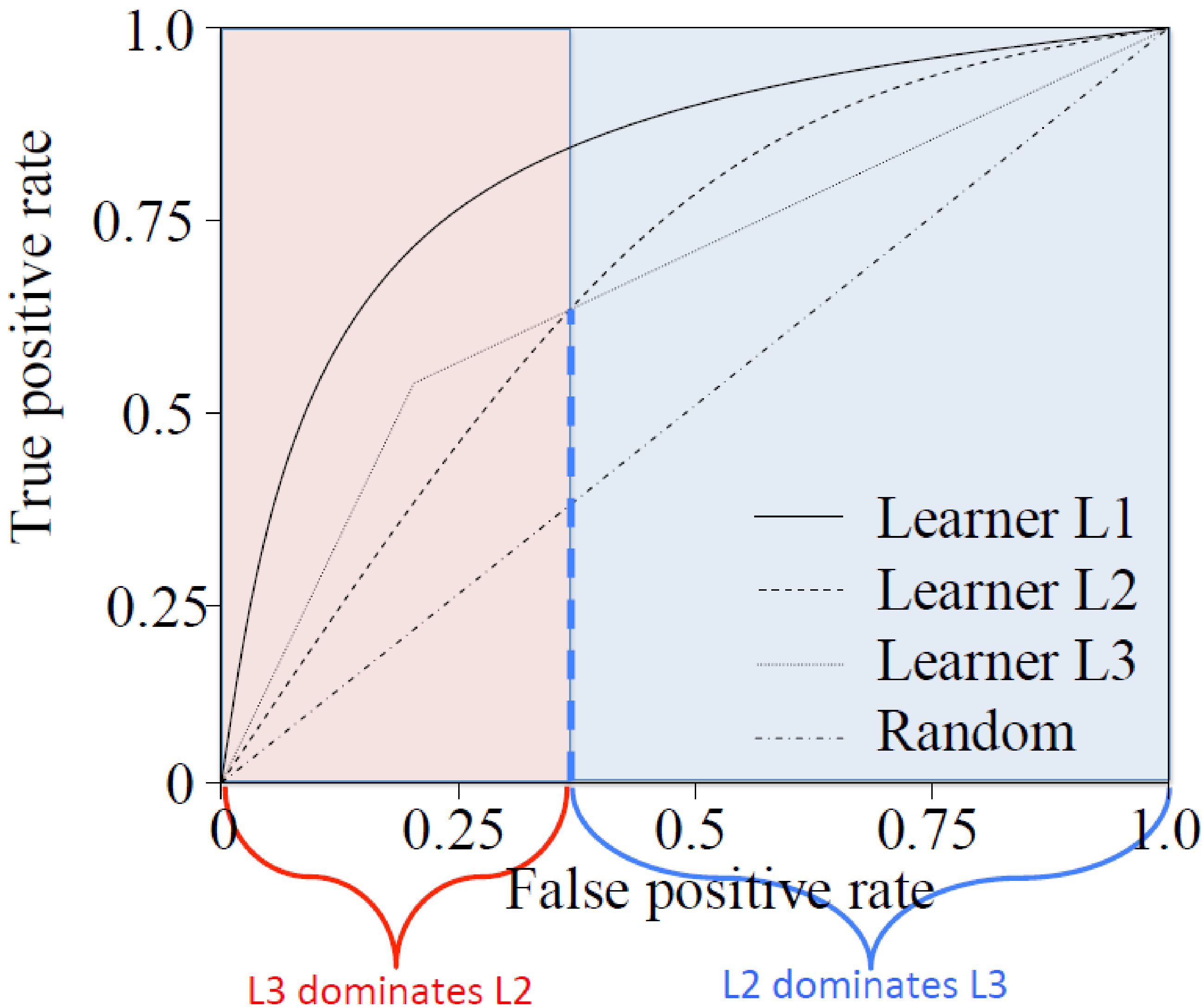


ROC Curve

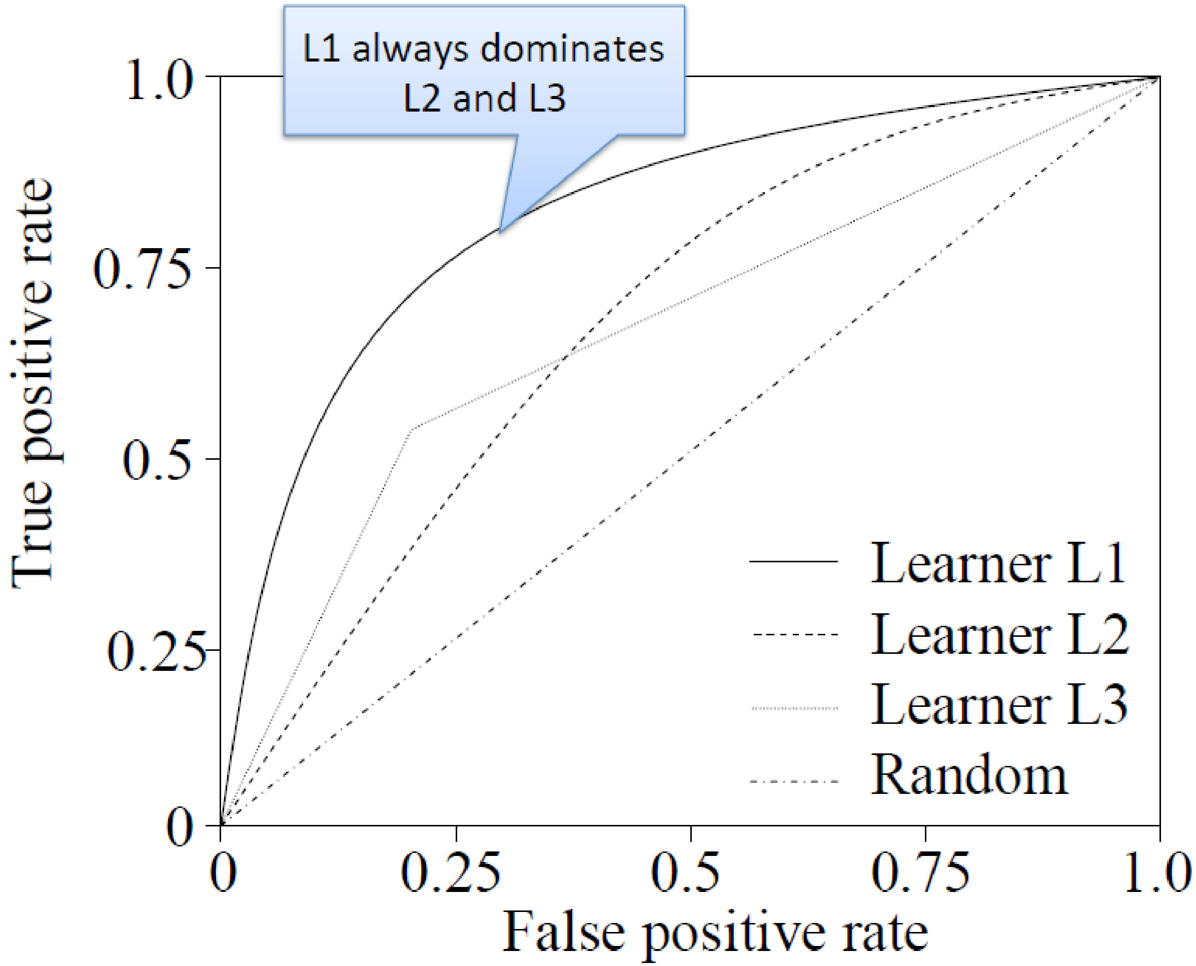


“Ideal”
learner
Random model is always diagonal line in ROC space
1.0
0.75
0.5
0.25
0
True positive rate
0
0.25
0.5
0.75
1.0
False positive rate
Learner L1
Learner L2
Learner L3
Random

ROC Curve

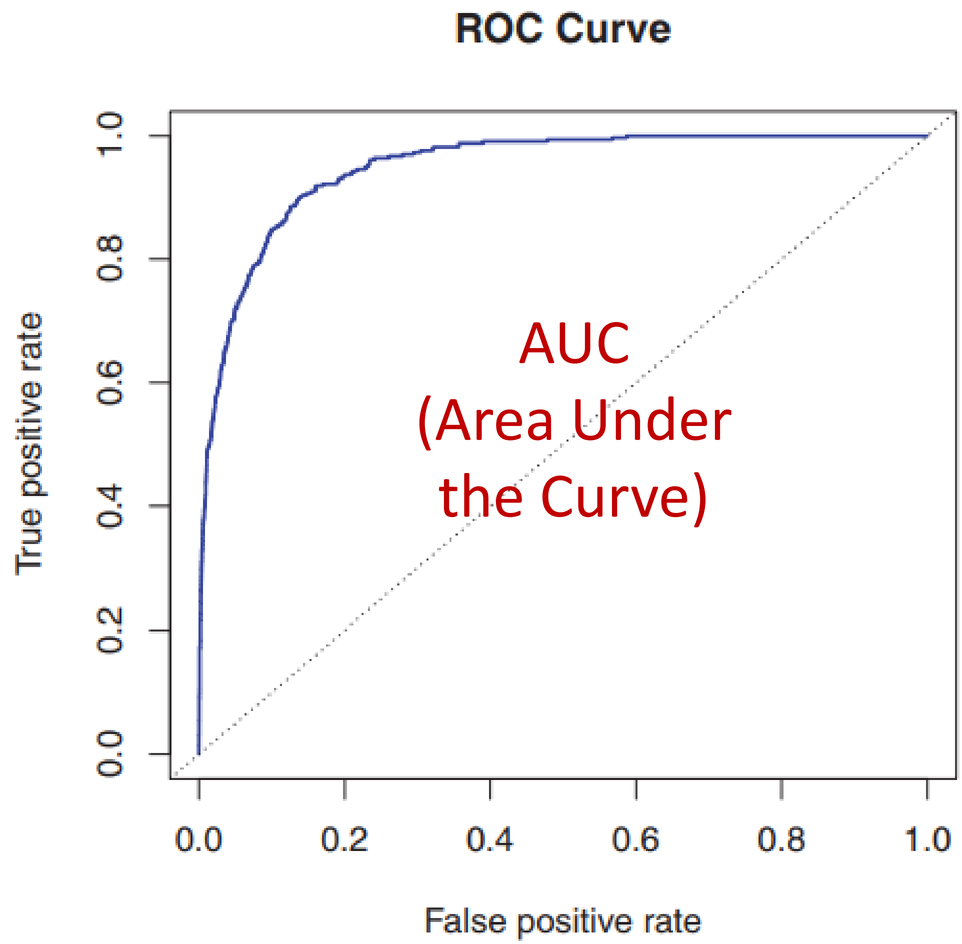

1.0
0.75
0.5
0.25
0
True positive rate
0
0.25
0.5
0.75
1.0
False positive rate
Learner L1
Learner L2
Learner L3
Random
L3 dominates L2
L2 dominates L3

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

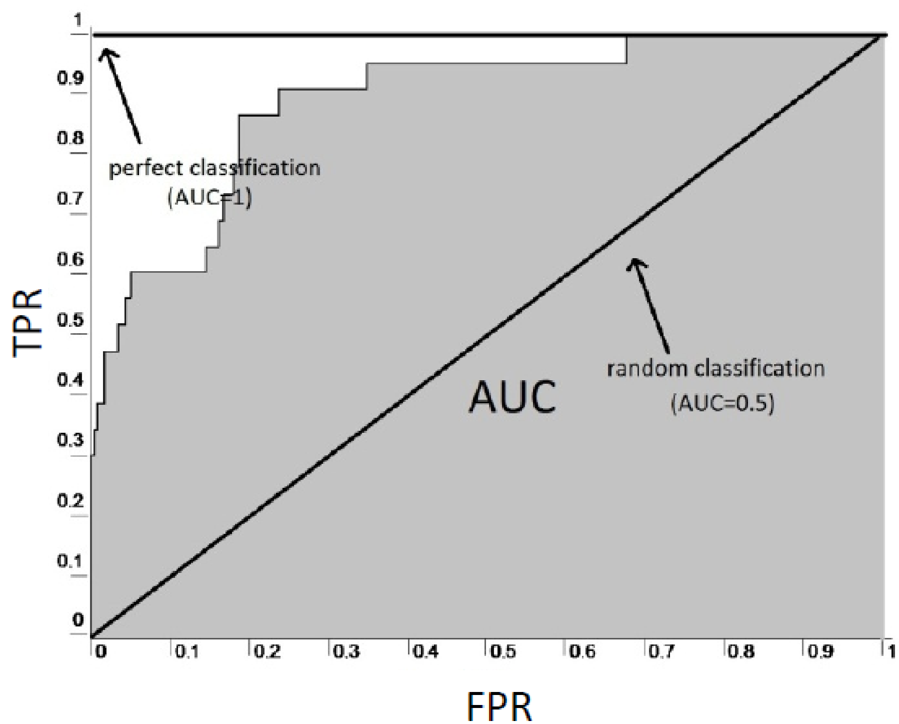
ROC Curves

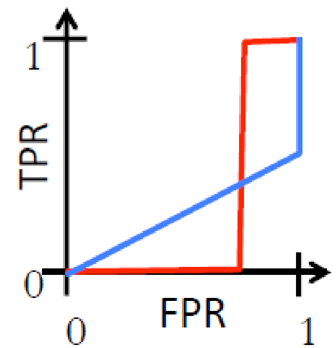


- Another useful metric: Area Under the Curve (AUC)
- The closest to 1, the better!

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

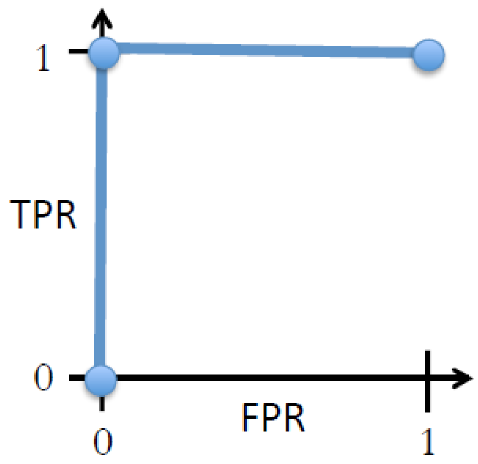




Same AUC, very different performance

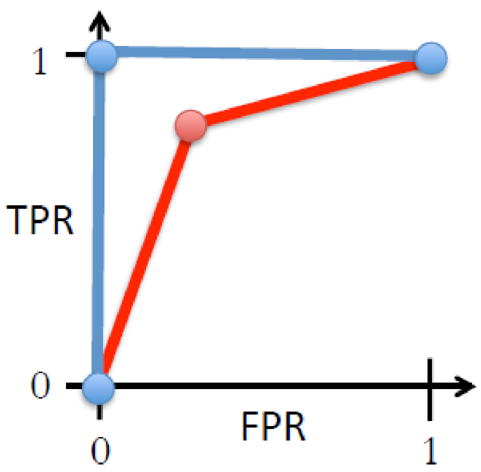
ROC Example

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
			$TPR = 5/5 = 1$ $FPR = 4/4 = 1$	$TPR = 5/5 = 1$ $FPR = 0/4 = 0$	$TPR = 0/5 = 0$ $FPR = 0/4 = 0$



ROC Example

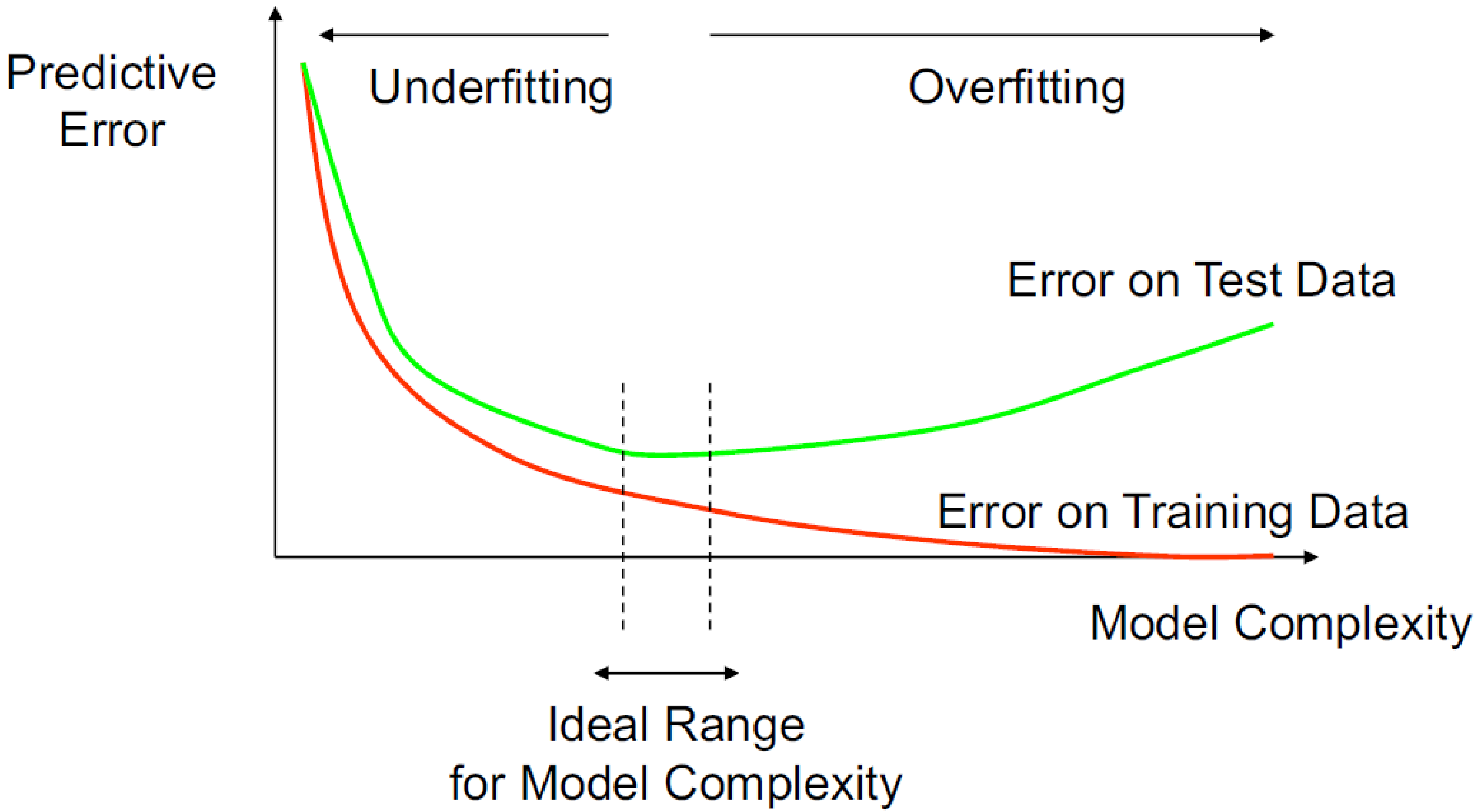
i	y_i	$p(y_i = 1 \mid \mathbf{x}_i)$	$h(\mathbf{x_i} \mid \theta = 0)$	$h(\mathbf{x_i} \mid \theta = 0.5)$	$h(\mathbf{x_i} \mid \theta = 1)$
1	1	0.9	1	1	0
2	1	0.8	1	1	0
3	1	0.7	1	1	0
4	1	0.6	1	1	0
5	1	**0.2**	1	**0**	0
6	0	**0.6**	1	**1**	0
7	0	0.3	1	0	0
8	0	0.2	1	0	0
9	0	0.1	1	0	0
			$TPR = 5/5 = 1$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



Goals of classification

- Produce models with high accuracy / low error
- Generalize well
  - Avoid overfitting (perform well on training set, but poorly on testing data)
- Find the simplest model that produces reasonable accuracy
  - Occam’s Razor
- Reduce both bias and variance!

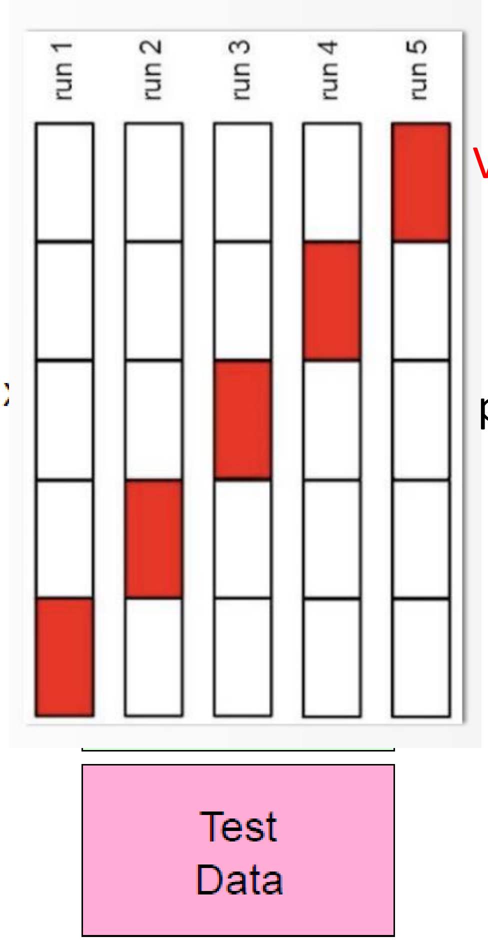
How Overfitting Affects Prediction



How can we avoid over-fitting without having access to testing data?

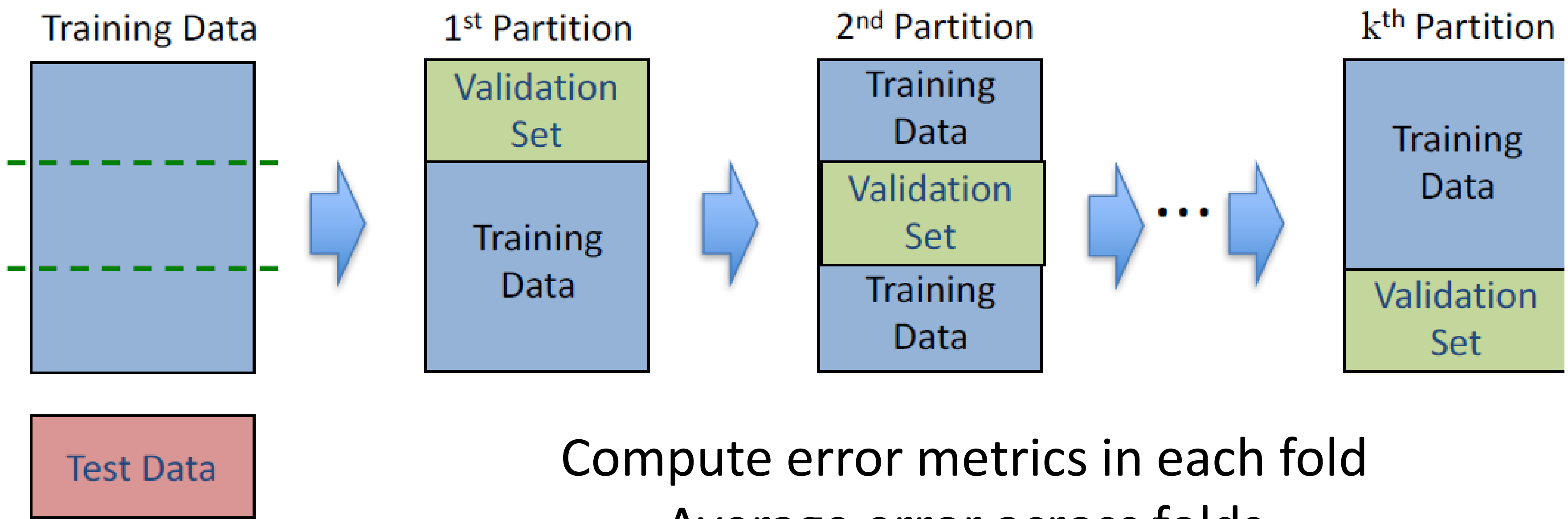
Cross Validation

- Data: labeled instances, e.g. emails marked spam/ham
  – Training set
  - Test set
  - Randomly split training set into training and validation, e.g., 66% - 33%
- Features: attribute-value pairs which characterize each
- Experimentation cycle
  – Select a hypothesis f (Tune hyperparameters on held-out or *validation* set)
  - Estimate and reduce average error during multiple runs by randomly choosing validation set
  - Compute final error on testing set
- Evaluation
  – Accuracy: fraction of instances predicted correctly
  - Use other metrics as appropriate (precision, recall)



- Improves model generalization
- Avoids overfitting

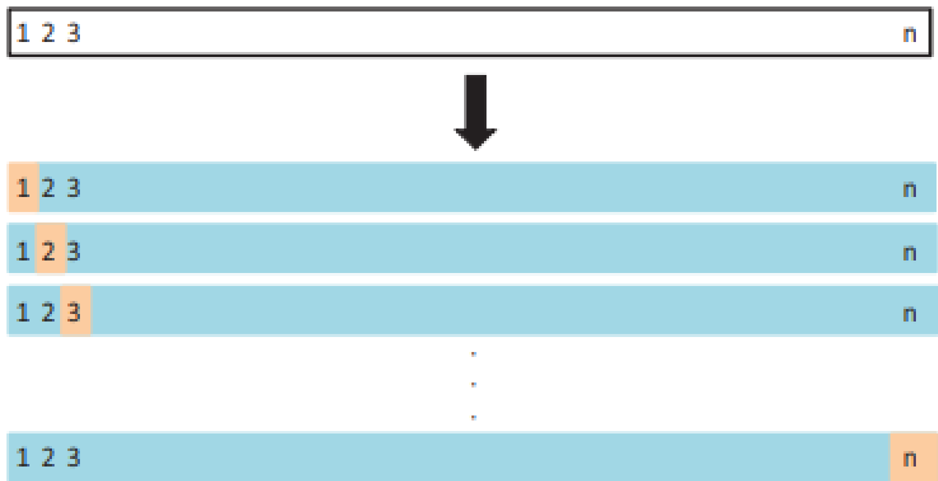
Cross Validation



Compute error metrics in each fold
Average error across folds

- CV can be used for
  - Hyper-parameter selection
  - Comparing different models and features
- 1. k-fold Cross-Validation
  - Split data into k partitions of equal size

Cross Validation



- 2. Leave-one-out CV (LOOCV)
  - k=n (validation set only one point)
- Pros: Less bias
- Cons: More expensive to implement, higher variance
- Recommendation: perform k-fold CV with k=5 or k=10

Outline

- Evaluation of classifiers
  - Metrics
  - ROC curves
- Linear Discriminant Analysis (LDA)
- Lab (logistic regression, LDA, kNN)
- Feature selection
  - Wrapper
  - Filter
  - Embedded methods

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)

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

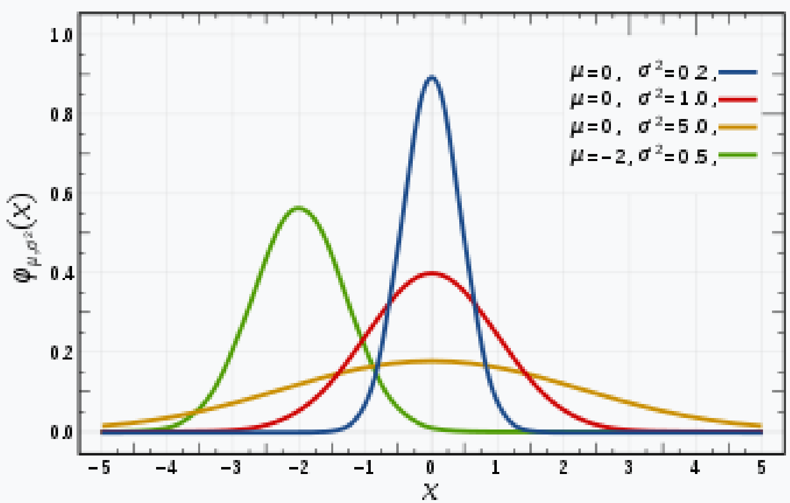
$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

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

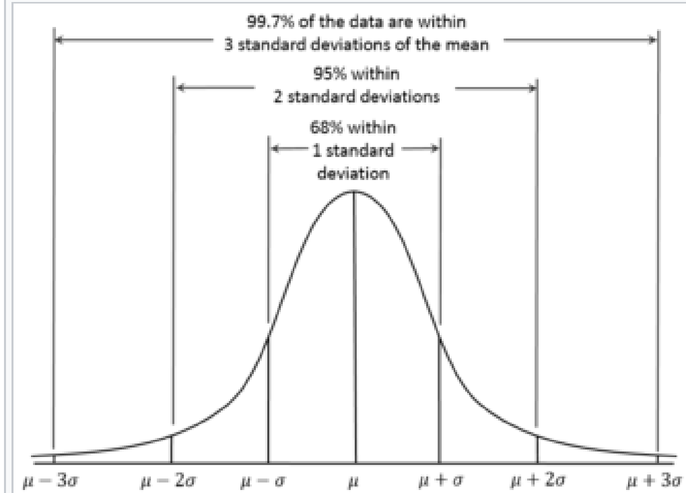
Gaussian Distribution

Normal Distribution

Probability density function



The red curve is the *standard normal distribution*



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Notation	$\mathcal{N}(\mu,\sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 > 0$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

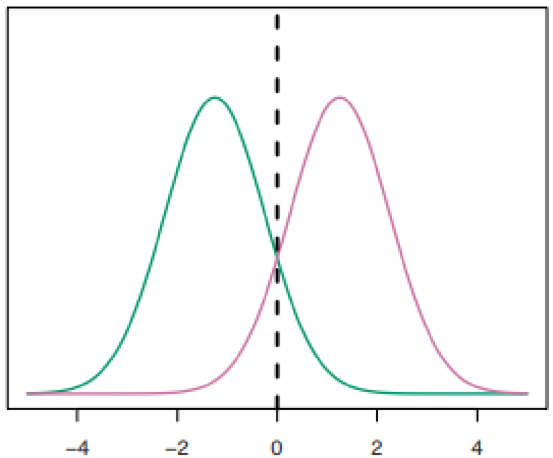
LDA decision boundary

Pick class k to maximize

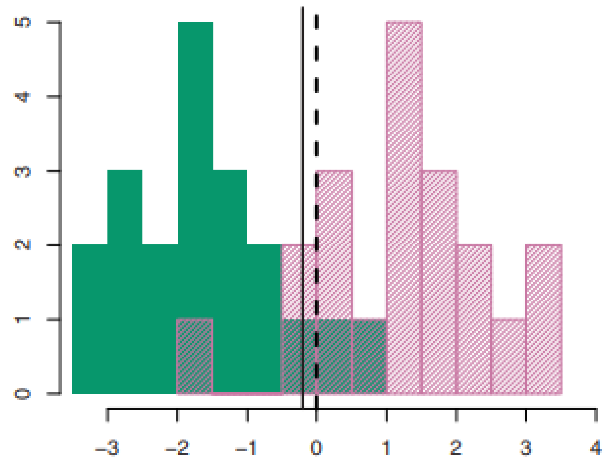
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

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

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



True decision boundary



Estimated decision boundary

LDA in practice

Given training data $(x^{(i)}, y^{(i)}), i = 1, \dots, n, y^{(i)} \in \{1, \dots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x^{(i)}$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x^{(i)} - \hat{\mu}_k)^2$$

2. Estimate prior

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

Given testing point x, predict k that maximizes:

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

Acknowledgements

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