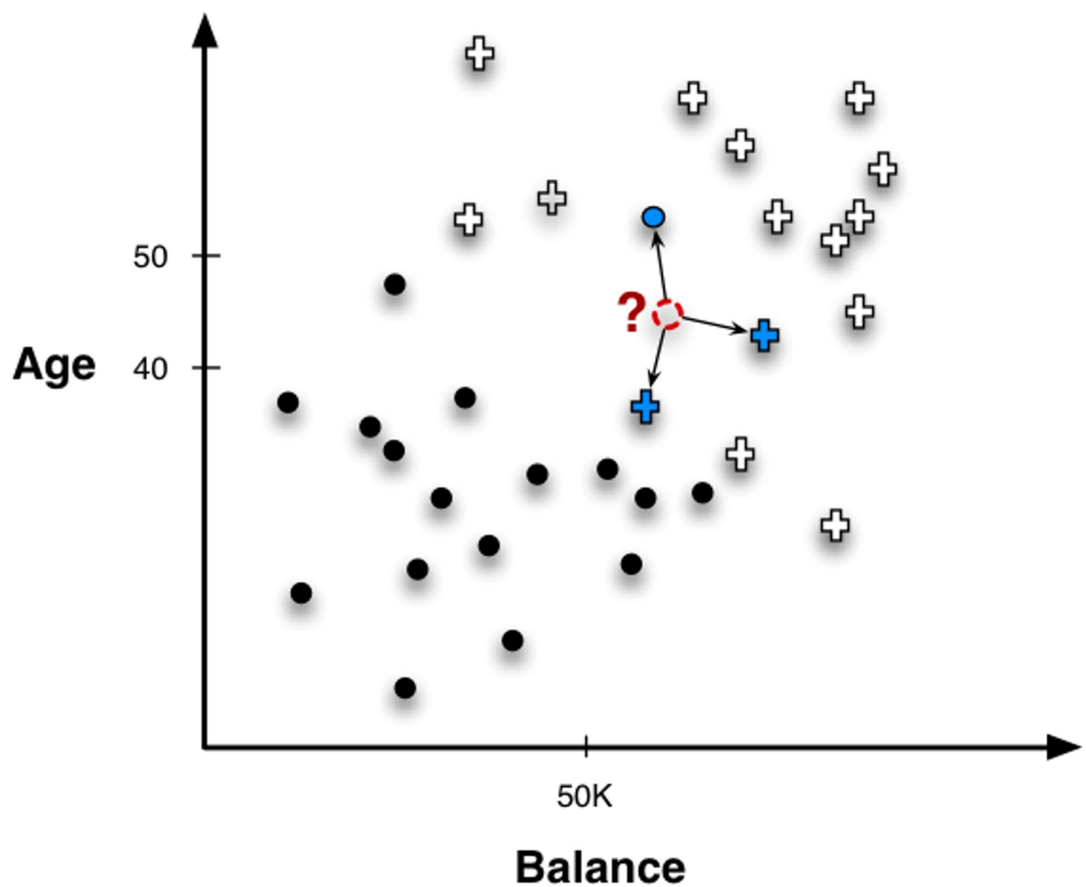
Week 3 Lecture

MGTF 495

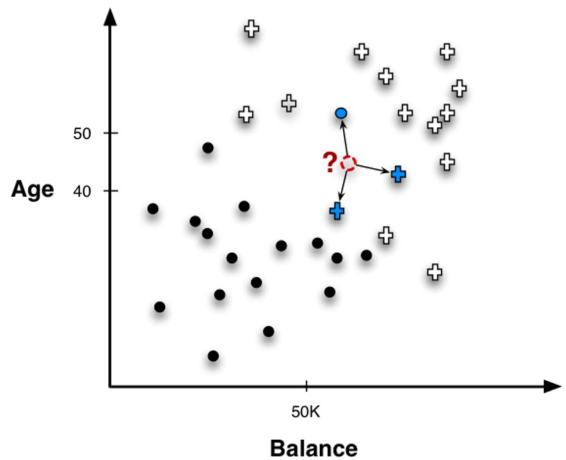
Class Outline

- Nearest Neighbor Classification
- Algorithmic Analysis
- Ensemble Methods
- Evaluation Metrics

Nearest neighbors for predictive modeling



Nearest neighbor in classification



Majority vote

How many neighbors?

Also consider the distance to vary the influence of the neighbors

Customer	Age	Income (1000s)	Cards	Response (target)	Distance from David
David	*37*	*50*	*2*	*?*	
John	35	35	3	Yes	$\sqrt{(35-37)^2+(35-50)^2+(3-2)^2} = 15.16$
Rachael	22	50	2	No	$\sqrt{(22-37)^2+(50-50)^2+(2-2)^2} = 15$
Ruth	63	200	1	No	$\sqrt{(63-37)^2+(200-50)^2+(1-2)^2} = 152.23$
Jefferson	59	170	1	No	$\sqrt{(59-37)^2+(170-50)^2+(1-2)^2} = 122$
Norah	25	40	4	Yes	$\sqrt{(25-37)^2+(40-50)^2+(4-2)^2} = 15.74$

Name	Distance	Similarity Weight	Contribution	Class
Rachael	15.0	0.004444	0.344	No
John	15.2	0.004348	0.336	Yes
Norah	15.7	0.004032	0.312	Yes
Jefferson	122.0	0.000067	0.005	No
Ruth	152.2	0.000043	0.003	No

Nearest neighbor classification

Given a labeled training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$

Example: the MNIST dataset of handwritten digits.

To classify a new instance x :

Find its nearest neighbor amongst the $x^{(i)}$, Return $y^{(i)}$

The data space

We need to choose a distance function.

Each image is 28 × 28 grayscale.
One option: Treat images as 784-dimensional vectors, and use Euclidean (l_2) distance:

$$\|x - x'\| = \sqrt{\sum_{i=1}^{784} (x_i - x'_i)^2}.$$

Summary:

- Data space $X = R^{784}$ with l_2 distance
- Label space $Y = \{0, 1, . . . , 9\}$

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points?

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier?

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Ideas for improvement: (1) *k*-NN (2) better distance function.

K -nearest neighbor classification

Classify a point using the labels of its *k* nearest neighbors among the training points.

K -nearest neighbor classification

Classify a point using the labels of its *k* nearest neighbors among the training points.

MNIST:

k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

K -nearest neighbor classification

Classify a point using the labels of its k nearest neighbors among the training points.

MNIST:

k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

How to choose k in general?

K -nearest neighbor classification

Classify a point using the labels of its *k* nearest neighbors among the training points.

MNIST:

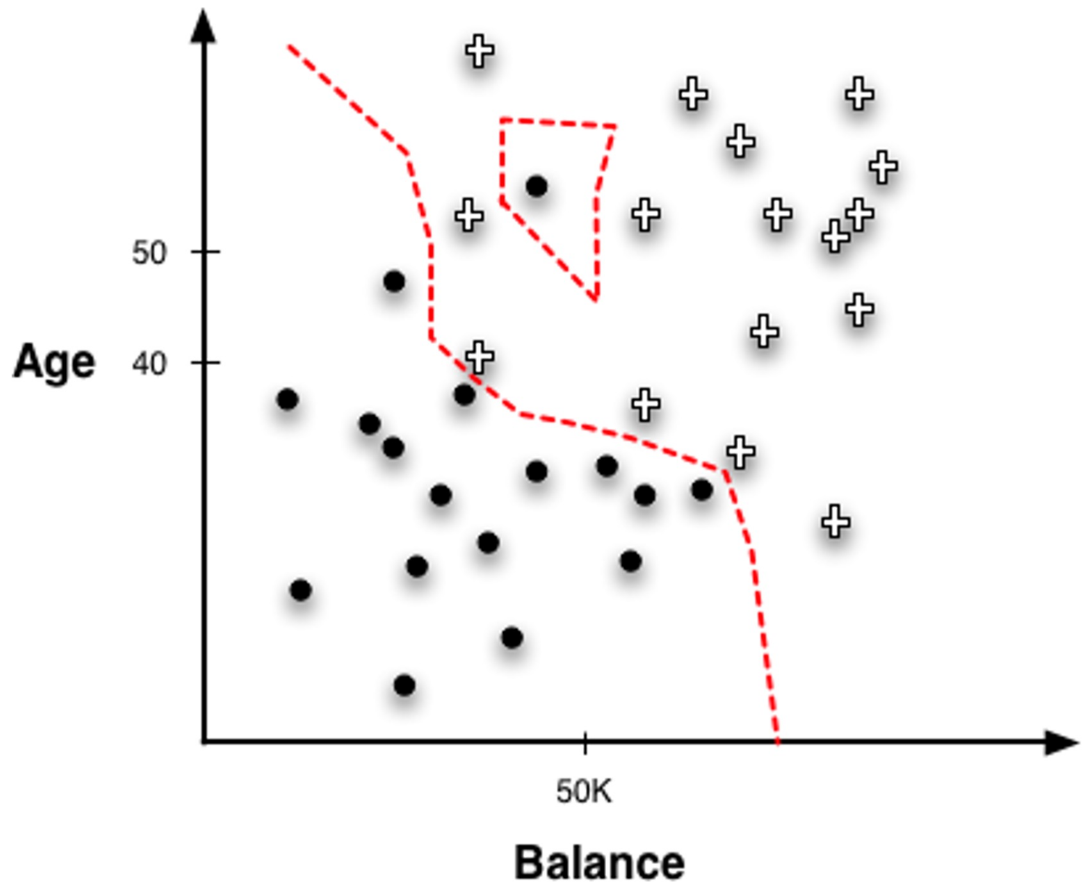
k	1	3	5	7	9	11
Test error (%)	3.09	2.94	3.13	3.10	3.43	3.34

How to choose *k* in general?

- Let $S \in Z^n$ be the training set, where $Z = X \times Y$ is the space of *labeled* points.
- Select a Hold-out set **V** – also called the Validation set
- Train the k-NN on (S – V) for some value of *k*
- Check the performance of this classifier on **V**
- Repeat for a different value of *k*
- Pick the best *k* (one with minimum test error)

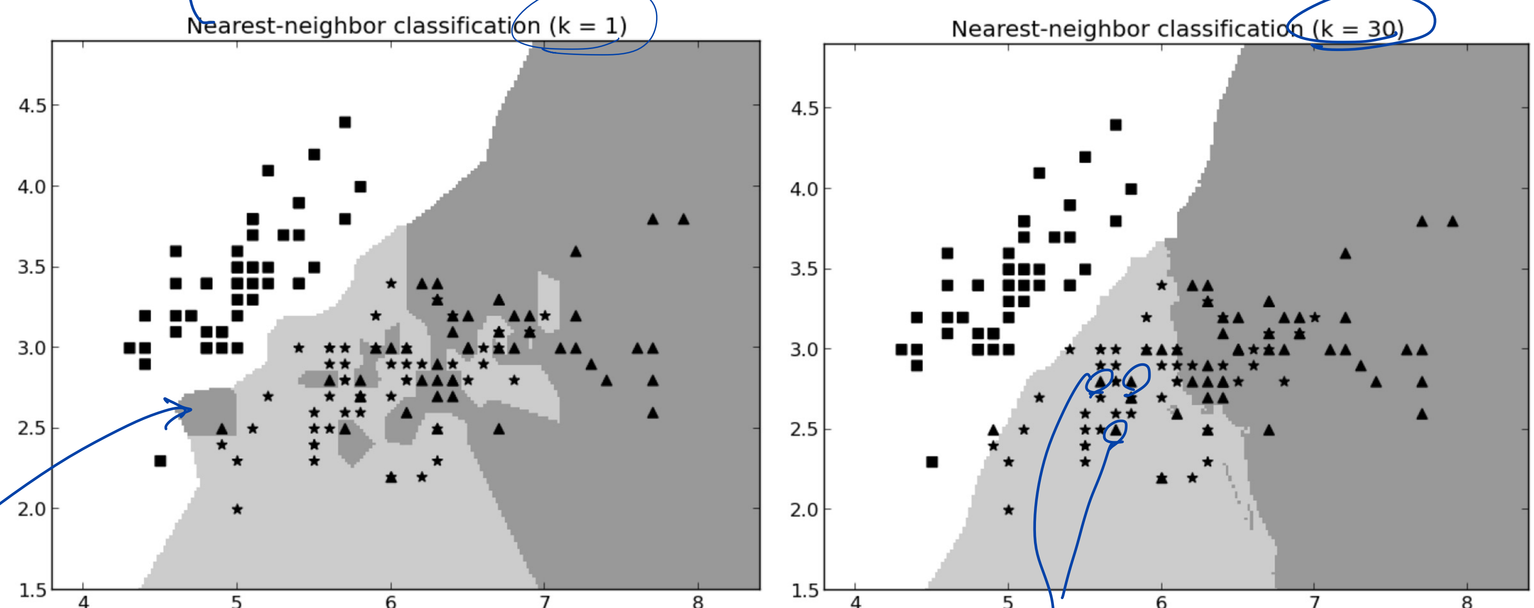
The above procedure can also be performed using Leave-one-out cross validation in which we choose one training point at a time as the validation set

Geometric Interpretation, Over-fitting, and Complexity



Nearest neighbor in classification

如果用留一法，可能会选K=1的



overlap

也有不准确的，算outlier。Boundary可行性高

Nearest neighbor classifiers follow very specific boundaries

1-NN strongly tends to overfit (k is a complexity parameter!)

Use cross-validation or nested holdout testing

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

It makes sense to choose distance measures that are invariant under:

- Small translations and rotations. E.g. *tangent distance.*
- A broader family of natural deformations. E.g. *shape context*

Better distance functions

Let *x* be an image. Consider an image *x'* that is just like *x* , but is either:

- shifted one pixel to the right, or
- Rotated slightly.

Then $\|x - x'\|$ could easily be quite large.

It makes sense to choose distance measures that are invariant under:

- Small translations and rotations. E.g. *tangent distance.*
- A broader family of natural deformations. E.g. *shape context*

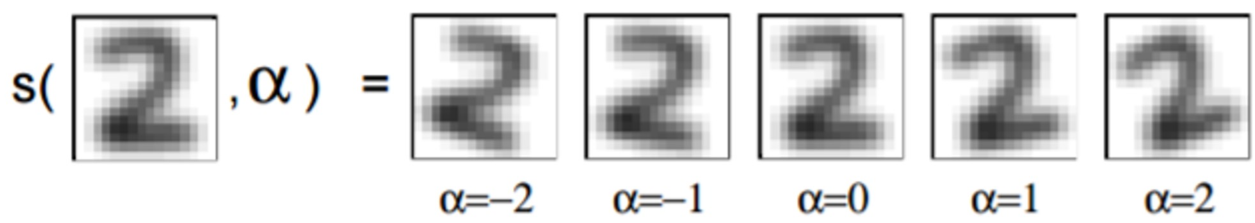
Test error rates:

ℓ_2	tangent distance	shape context
3.09	1.10	0.63

Better distance functions

Tangent Distance

- Is an invariant distance measure
- Especially effective for OCR
- Small transformations of certain image objects does not affect class membership

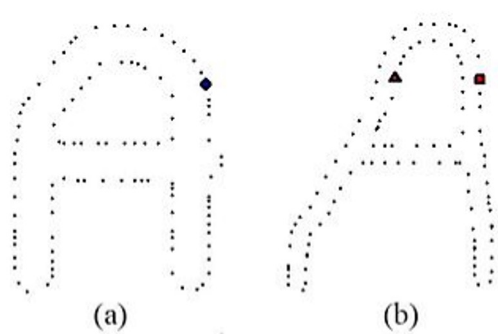


Refer: http://yann.lecun.com/exdb/publis/pdf/simard-00.pdf for more details

Better distance functions

Shape Context

- Is a feature descriptor in object recognition
- A way of describing shapes that allows for measuring shape similarity and the recovering of point correspondences
- pick n points on the contours of a shape



References:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.441.6897&rep=rep1&type=pdf
https://en.wikipedia.org/wiki/Shape_context

Better distance functions

Shape Context

- Is a feature descriptor in object recognition
- A way of describing shapes that allows for measuring shape similarity and the recovering of point correspondences
- pick n points on the contours of a shape

(a) (b)

Are there other families of distance functions that are often useful?

ℓ_p norms

How can we measure the length of a vector in $\mathbb{R}^m$?
Usual choice is the *Euclidean norm*:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}.$$

ℓ_p norms

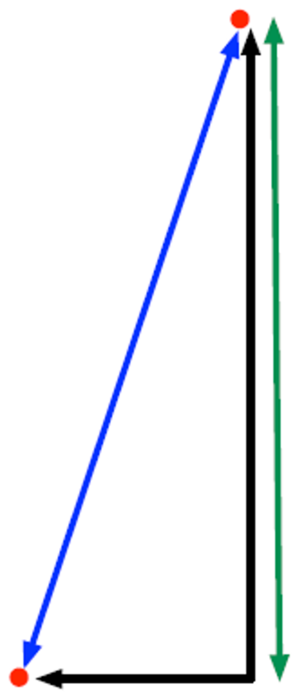
How can we measure the length of a vector in $\mathbb{R}^m$?
Usual choice is the *Euclidean norm*:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}.$$

Generalization: For $p \geq 1$, the ℓ_p norm is

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}$$

- $p = 2$: Euclidean norm
- ℓ_1 norm: $\|x\|_1 = \sum_{i=1}^{m} |x_i|$
- ℓ_∞ norm: $\|x\|_\infty = \max_i |x_i|$



Quick quiz

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector?

Quick quiz

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector?
2. Suppose $\|x\|_1 = 1$.

What is the maximum value of the l_2 norm?

Quick quiz: Answers

Suppose data lie in R^p

1. What is the l_2 norm of the all-ones vector? $\mathbf{1/(p)^{1/2}}$
2. Suppose $\|x\|_1 = 1$.

What is the maximum value of the l_2 norm? **One.**

Metric spaces

A more general notion is a *metric space*.

Metric spaces

A more general notion is a *metric space*.

Let X be the space in which data lie. A distance function $d : X \times X \rightarrow \mathrm{R}$ is a *metric* if it satisfies these properties:

- $d(x, y) \geq 0$ (non-negativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

Metric spaces

A more general notion is a *metric space*.

Let X be the space in which data lie. A distance function $d : X \times X \rightarrow \mathbb{R}$ is a *metric* if it satisfies these properties:

- $d(x, y) \geq 0$ (non-negativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

For instance:

- $\mathcal{X} = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p$
- $\mathcal{X} = \{\text{strings over some alphabet}\}$ and $d =$ edit distance.

Class Outline

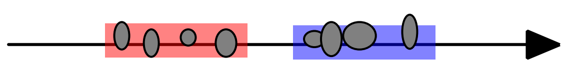
- Nearest Neighbor Classification
- Algorithmic Analysis
- Ensemble Methods
- Evaluation Metrics

Sensitivity to noise

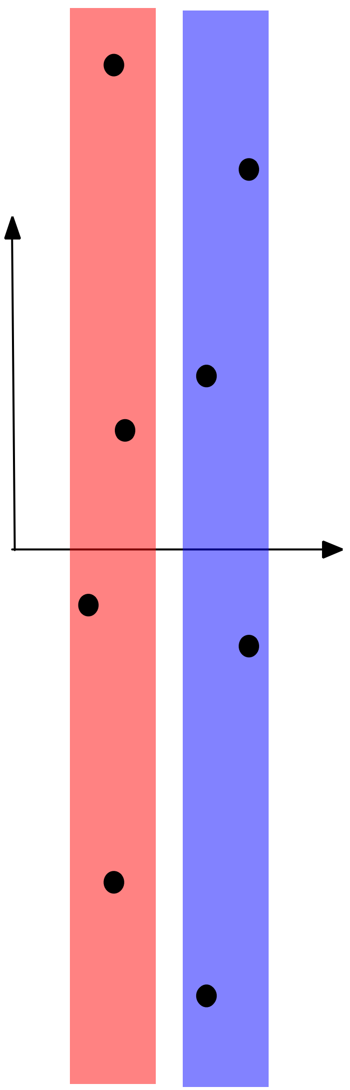
Adding a single sufficiently noisy feature can wreak havoc with nearest neighbor classifiers.

Sensitivity to noise

Adding a single sufficiently noisy feature can wreak havoc with nearest neighbor classifiers.



versus



How to do feature selection/reweighting for nearest neighbor?

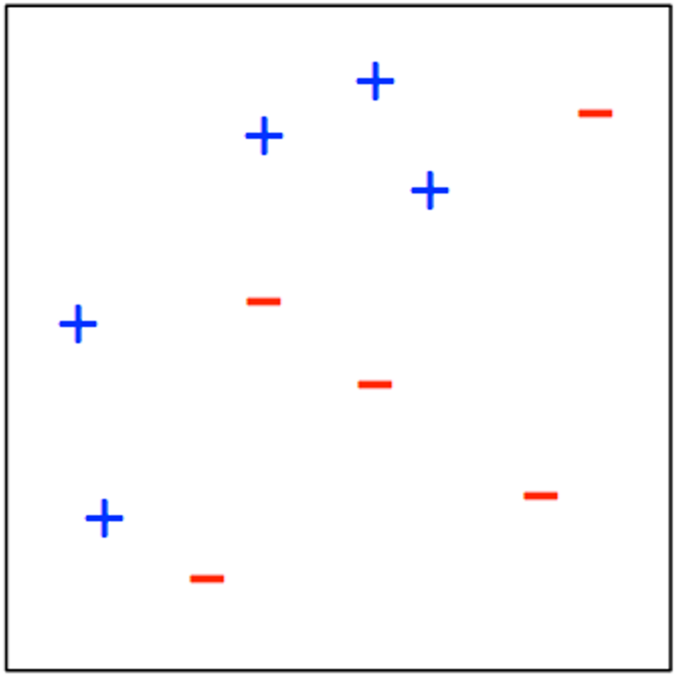
Class Outline

- Nearest Neighbor Classification
- Algorithmic Analysis
- Ensemble Methods
- Evaluation Metrics

Boosting

(From Freund and Schapire's tutorial.)

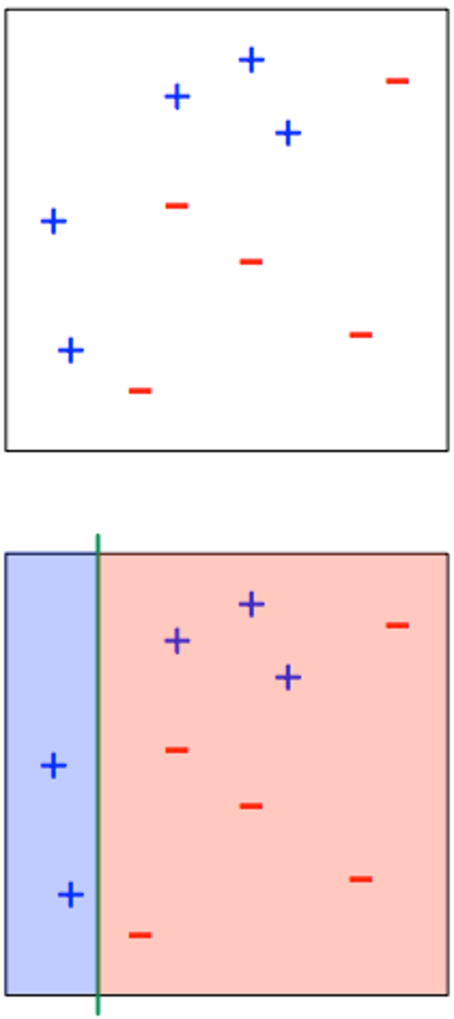
Training set:



Weak classifiers: single-coordinate thresholds, popularly known as "decision stumps" (in this case, horizontal and vertical lines)

Boosting

D_1

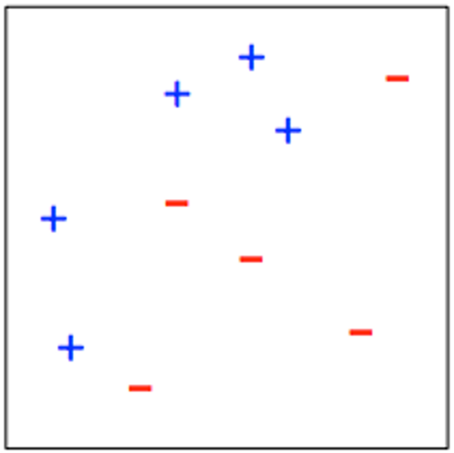


h_1

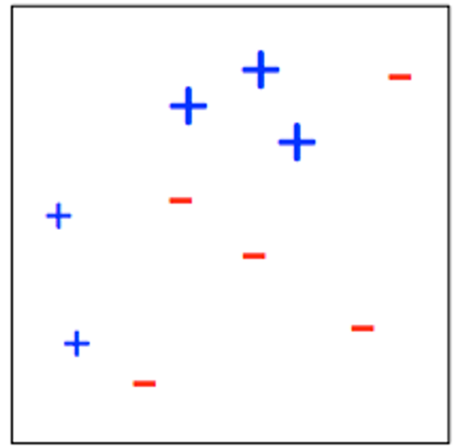
$r_1 = 0.40, \alpha_1 = 0.42$

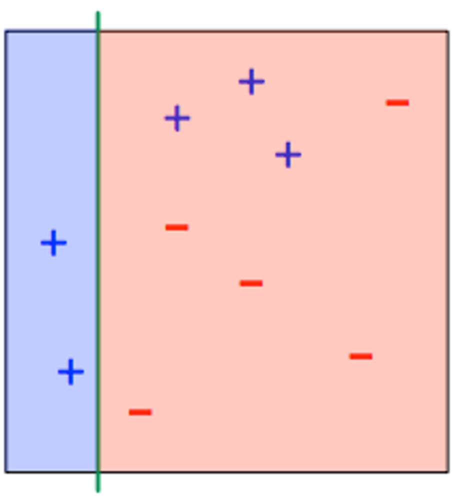
Boosting

D_1



D_2

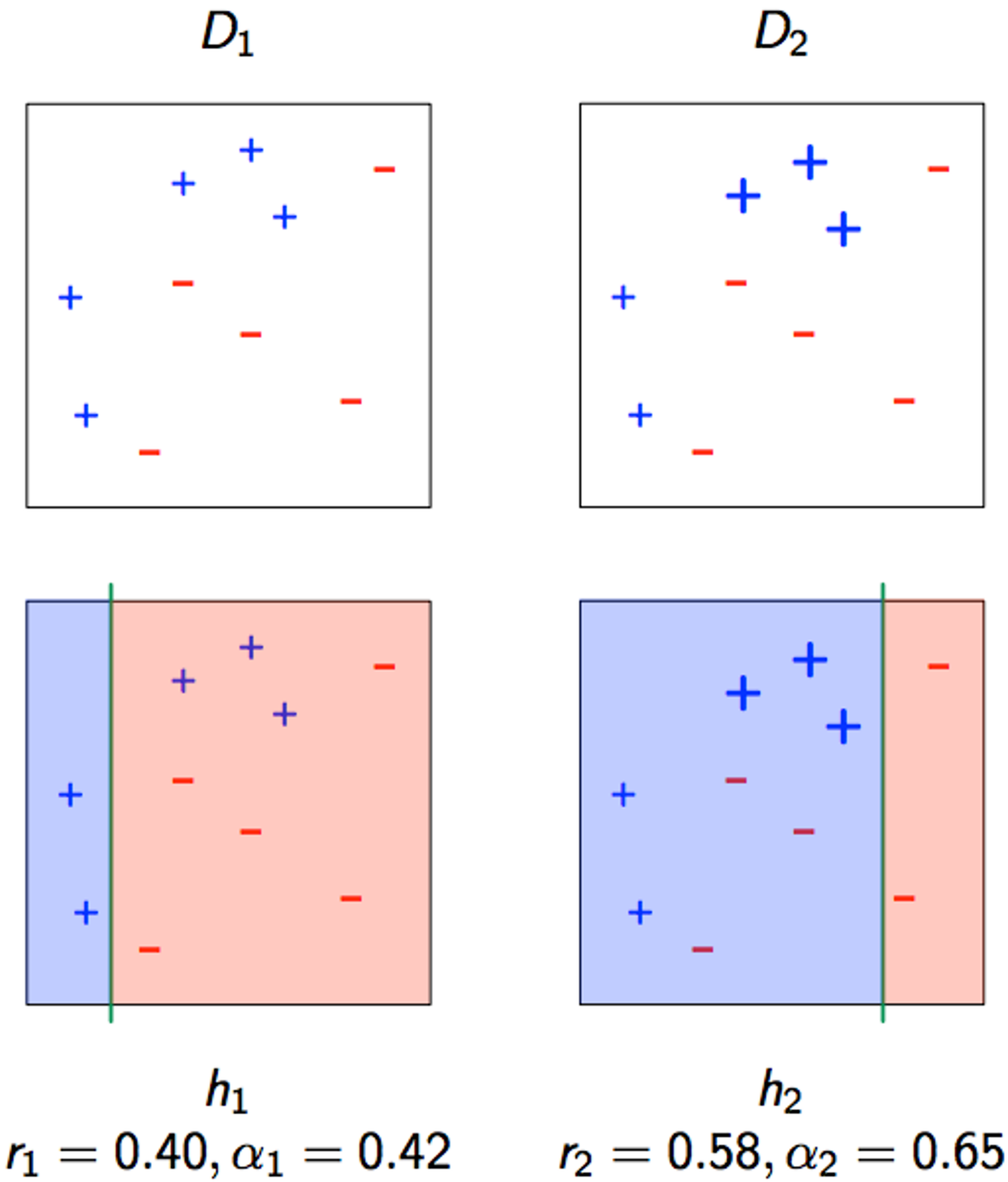




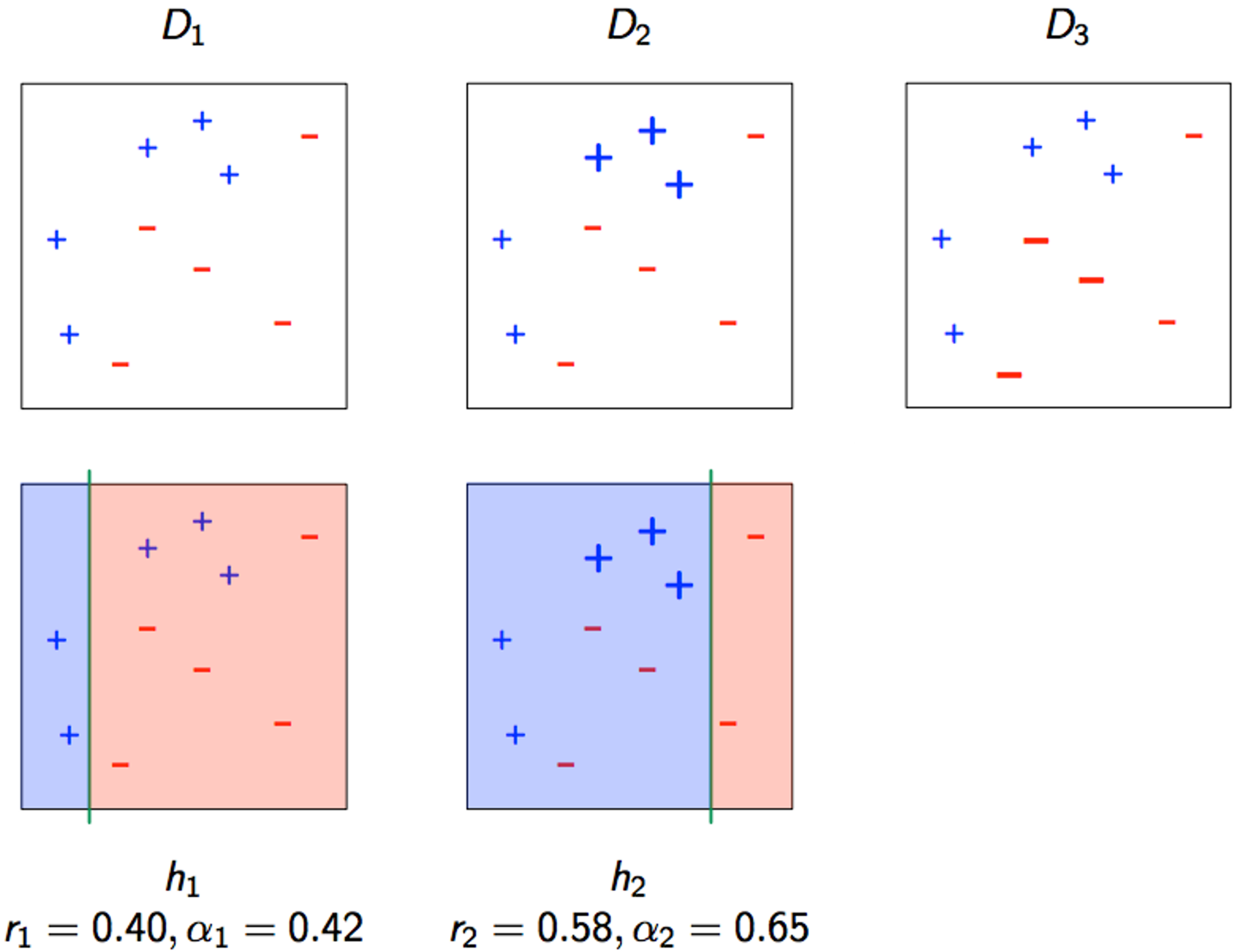
h_1

$r_1 = 0.40, \alpha_1 = 0.42$

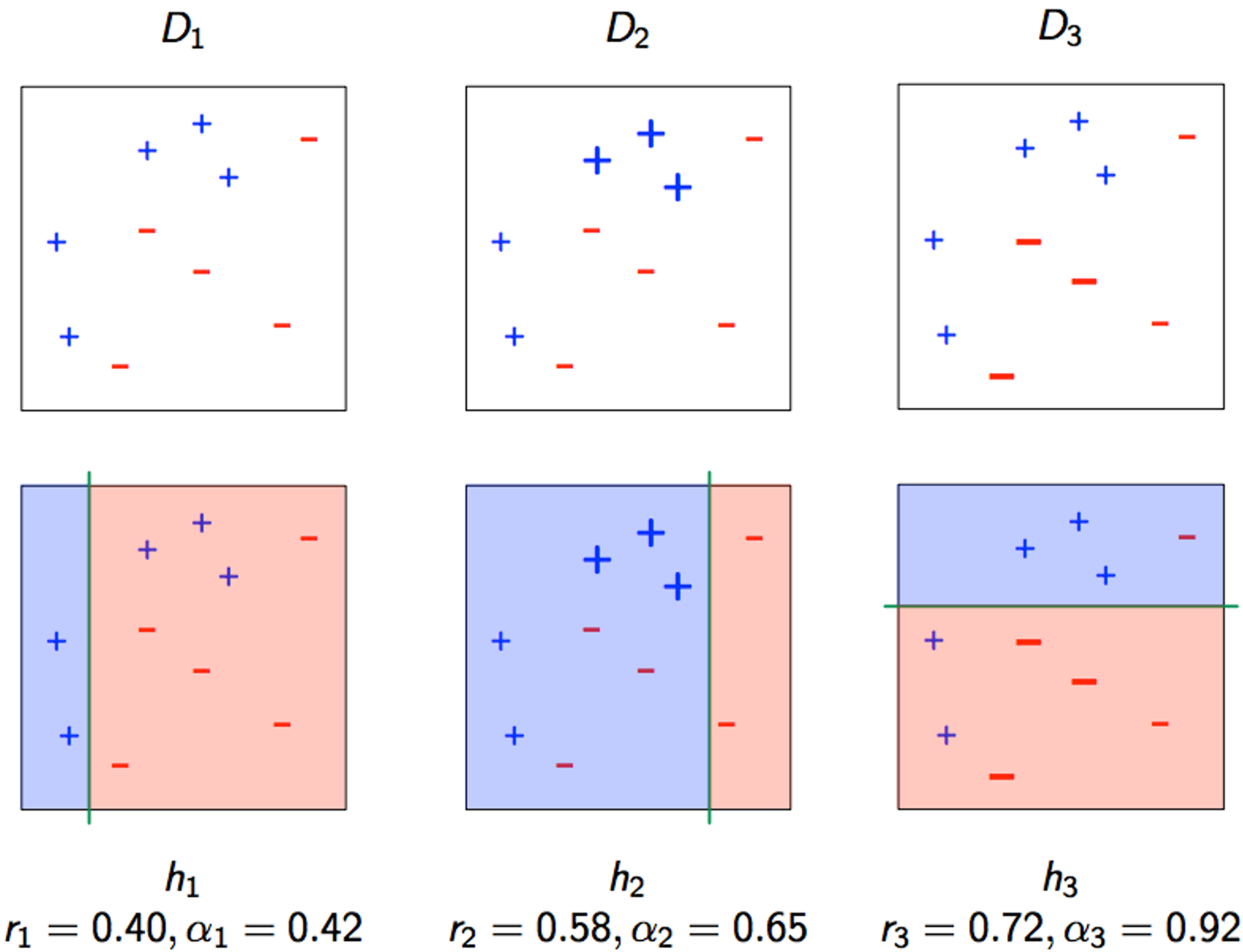
Boosting



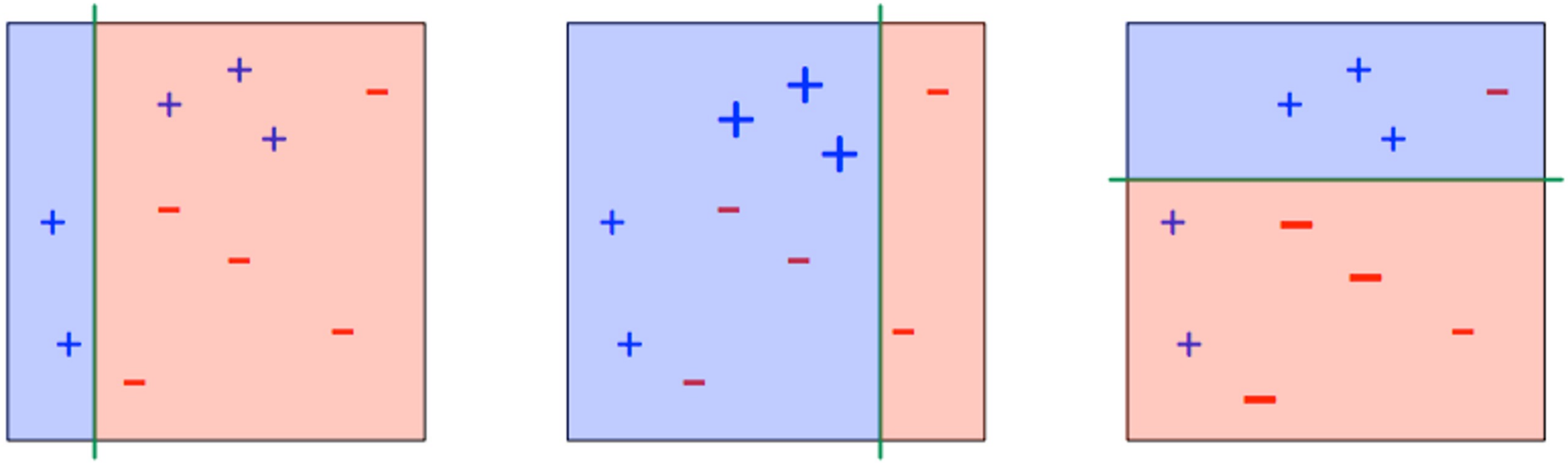
Boosting



Boosting


D_1
D_2
D_3
h_1
$r_1 = 0.40, \alpha_1 = 0.42$
h_2
$r_2 = 0.58, \alpha_2 = 0.65$
h_3
$r_3 = 0.72, \alpha_3 = 0.92$

Boosting



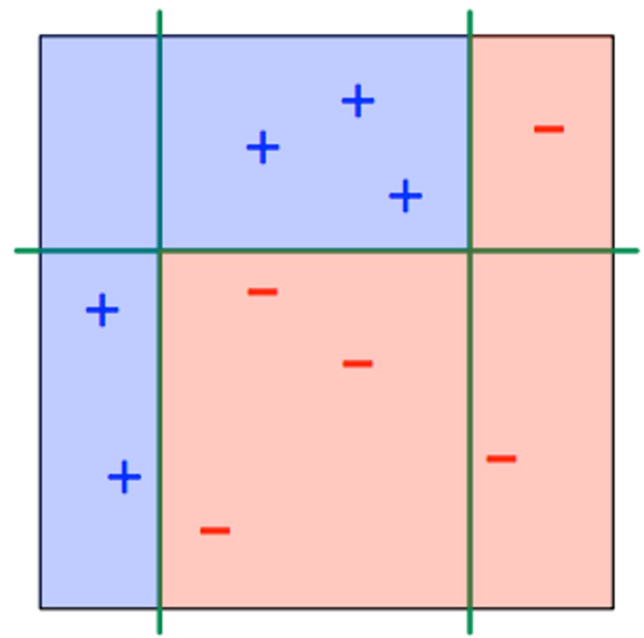
h_1 $\alpha_1 = 0.42$

h_2 $\alpha_2 = 0.65$

h_3 $\alpha_3 = 0.92$

Final classifier:

$\text{sign}(0.42 h_1(x) + 0.65 h_2(x) + 0.92 h_3(x))$



Bagging

Given a data set S of n labeled points:

For $t = 1$ to T :

Choose n' points randomly, with replacement, from S.

Fit a classifier h_t to these points.

(For instance: $T = 100$ or 1000, $n' = n$.)

Final predictor: majority vote of $h_1, \ldots, h_T$

The resampling and averaging reduces overfitting.

Random forests

Rather like bagging decision trees, but with an additional source of randomization.

Given a data set S of n labeled points:

- For t = 1 to T :
  - Choose n' points randomly, with replacement, from S
  - Fit a decision tree h_t to these points. When building the tree, at each node restrict the split direction to be one of k features at random. chosen at random.

(For instance:

Final predictor: majority vote of $h_1, \ldots, h_T$.

Class Outline

- Nearest Neighbor Classification
- Algorithmic Analysis
- Ensemble Methods
- Evaluation Metrics

Evaluation Metrics

- Up to now: measure a model‘s performance by some simple metric
  - classifier error rate, accuracy, …
- Simple example: accuracy

$$accuracy = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

- Classification accuracy is popular, but usually **too simplistic** for applications of data mining to real business problems
- **Decompose** and count the different types of correct and incorrect decisions made by a classifier

Unequal costs and benefits

- How much do we care about the different **errors** and correct decisions?
  - Classification accuracy makes no distinction between **false positive** and **false negative** errors
  - In real-world applications, different kinds of errors lead to different consequences!
- Examples for medical diagnosis:
  - a patient has cancer (although he does not)
    → **false positive error**, expensive, but not life threatening
  - a patient has cancer, but she is told that she has not
    → **false negative error**, more serious
- Errors should be counted separately
  - Estimate cost or benefit of each decision

Confusion Matrix

- A confusion matrix for a problem involving n classes
  - is an $n \times n$ matrix with the columns labeled with actual classes and the rows labels with predicted classes

		Predicted Classes	
		p	**n**
True Values	**1**	True Positives (TP)	False Negative (FN)
	0	False Positives (FP)	True Negatives (TN)

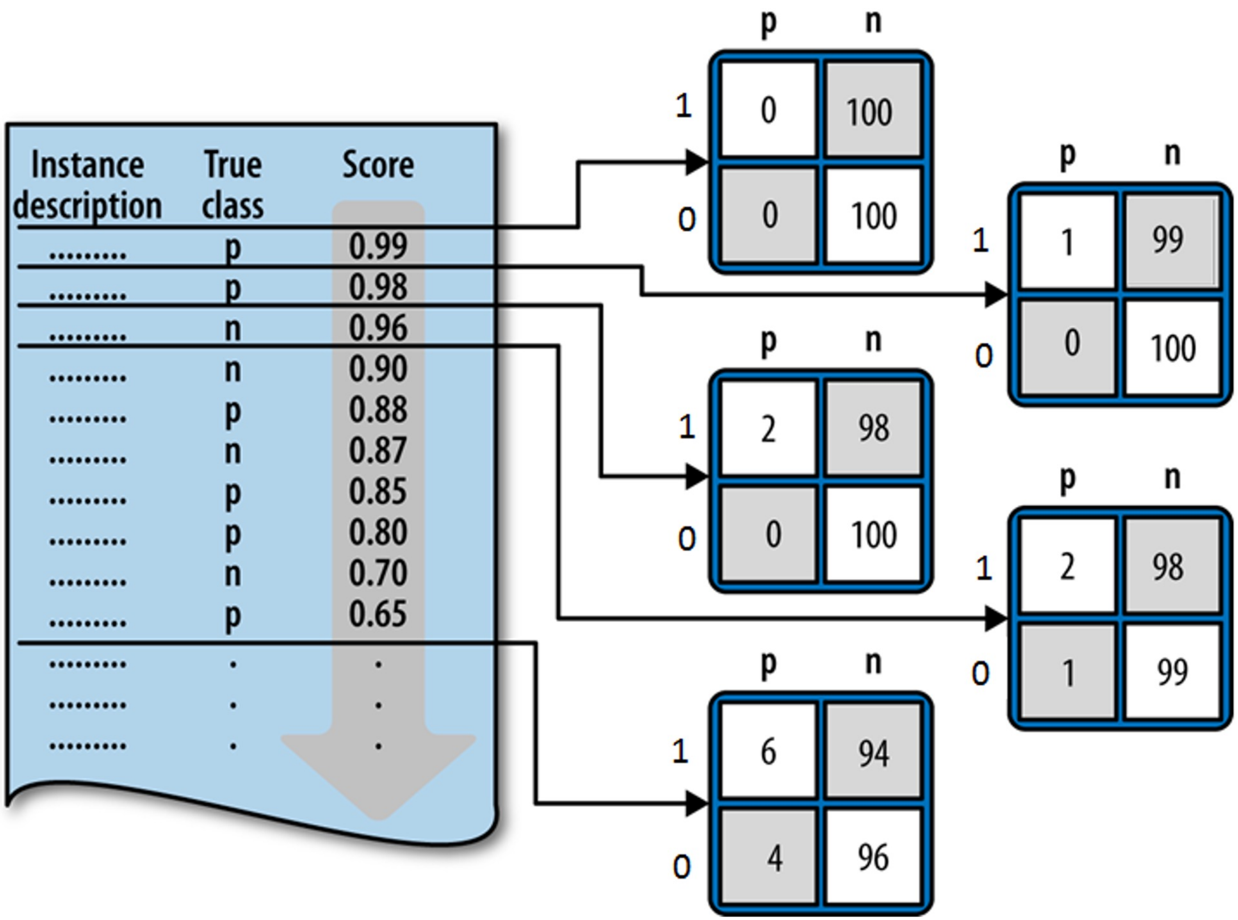
- Each example in a test set has an **actual class label** and the **class predicted** by the classifier
- The confusion matrix separates out the decisions made by the classifier
  - actual/true classes: **1** (Positive Label), **0** (Negative Label)
  - predicted classes: **p**(ositive), **n**(egative)
  - The main diagonal contains the count of correct decisions

Other Evaluation Metrics

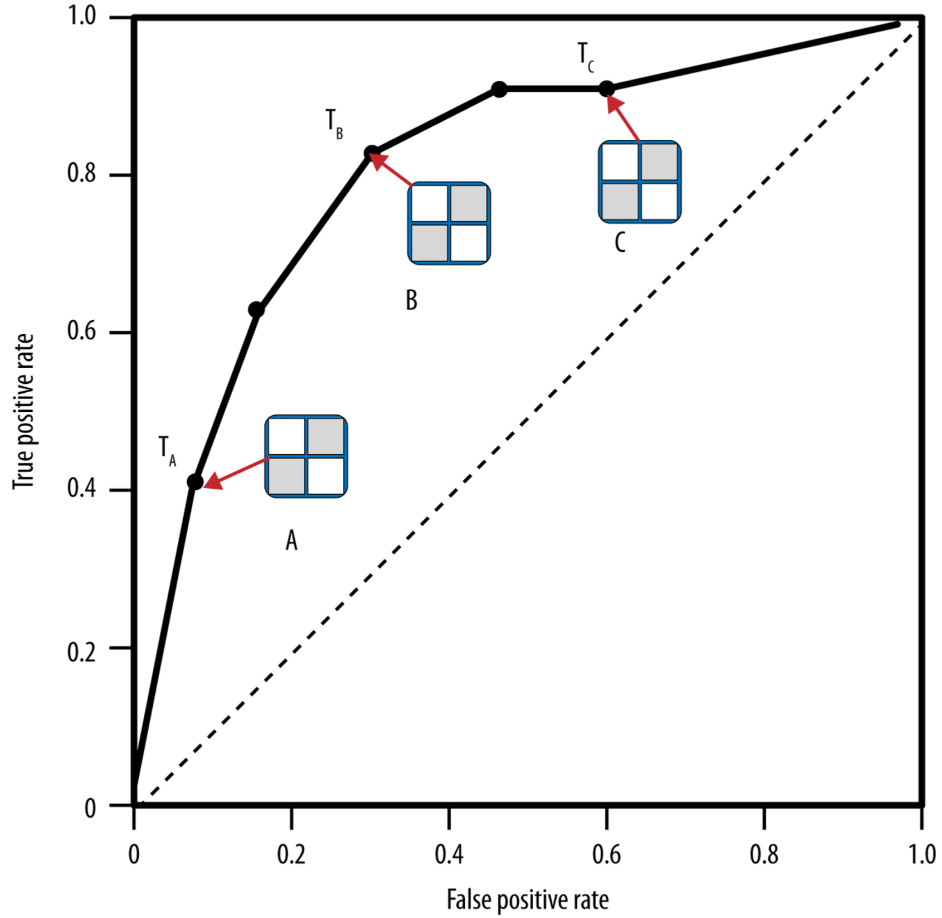
- Based on the entries of the confusion matrix, we can describe various evaluation metrics
  - True positive rate (Recall): $\frac{TP}{TP+FN}$
  - False negative rate: $\frac{FN}{TP+FN}$
  - Precision (accuracy over the cases predicted to be positive): $\frac{TP}{TP+FP}$
  - F-measure (harmonic mean): $2 \cdot \frac{precision \cdot recall}{precision+recall}$
  - Specificity: $\frac{TN}{TN+FP}$
  - Sensitivity: $\frac{TP}{TP+FN}$
  - Accuracy (count of correct decisions): $\frac{TP+TN}{P+N}$
  - False Positive Rate = 1 – Sensitivity = $\frac{FN}{TP+FN}$

ROC Curve

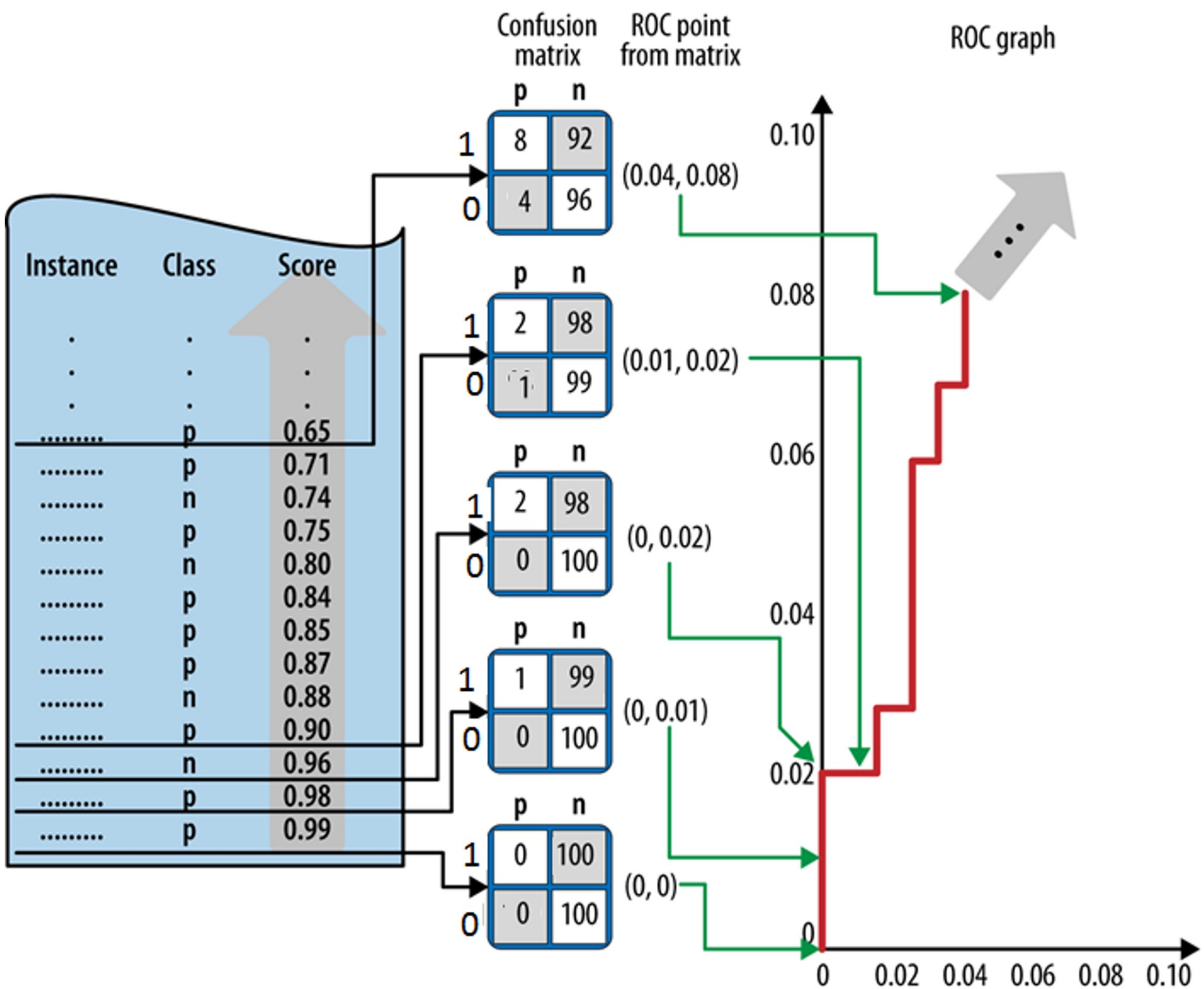
Ranking instead of classifying



ROC graphs and curves



ROC graphs and curves



Getting ROC curve: Algorithm

- Sort the test set by the model predictions
- Start with cutoff = max (prediction)
- Decrease cutoff, after each step count the number of true positives TP (positives with prediction above the cutoff) and false positives FP (negatives above the cutoff)
- Calculate TP rate (TP/P) and FP (FP/N) rate
- Plot current number of TP/P as a function of current FP/N

Area Under the ROC Curve (AUC)

- The area under a classifier's curve expressed as a fraction of the unit square
  - Its value ranges from zero to one
- The AUC is useful when a single number is needed to summarize performance, or when nothing is known about the operating conditions
  - A ROC curve provides more information than its area
- Equivalent to the Mann-Whitney-Wilcoxon measure
  - Also equivalent to the Gini Coefficient (with a minor algebraic transformation)
  - Both are equivalent to the probability that a randomly chosen positive instance will be ranked ahead of a randomly chosen negative instance

Performance evaluation

- Training Set:

Model	Accuracy
Classification Tree	95%
Logistic Regression	93%
k-Nearest Neighbors	100%
Naive Bayes	76%

- Test Set:

Model	Accuracy	AUC
Classification Tree	91.8% ± 0.0	0.614 ± 0.014
Logistic Regression	93.0% ± 0.1	0.574 ± 0.023
k-Nearest Neighbors	93.0% ± 0.0	0.537 ± 0.015
Naive Bayes	76.5% ± 0.6	0.632 ± 0.019

Performance evaluation

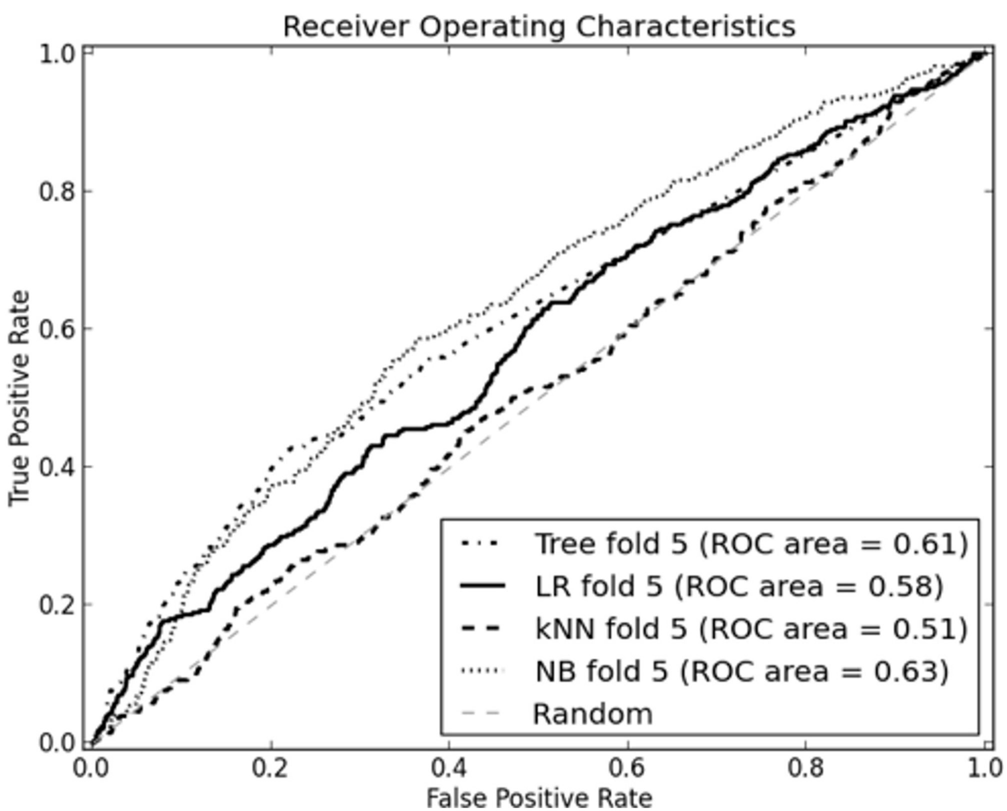
- Naive Bayes confusion matrix:

	p	n
1	127 (3%)	200 (4%)
0	848 (18%)	3518 (75%)

- k-Nearest Neighbors confusion matrix:

	p	n
1	3 (0%)	324 (7%)
0	15 (0%)	4351 (93%)

ROC Curve



11.4 . בהצלחה