

2. Let $X = \{(x, y) \mid x, y \in \mathfrak{R}\}$. Let $H =$ set of all linear decision surfaces in the plane (i.e., lines that separate examples into positive and negative classifications).

What is $VC(H)$?

- Show that there is a set of two points that can be shattered by constructing all dichotomies and showing that there is a $h \in H$ that represents each one.
- Is there a set of three points that can be shattered?
Yes, if not co-linear.
- Is there a set of four points that can be shattered?
Why or why not?
- More generally: the VC dimension of linear decision surfaces in an r -dimensional space (i.e., the VC dimension of a perceptron with r inputs) is $r + 1$.

Finding VC dimension

General approach:

- To show that $VC(H) \geq m$, show that there exists a subset of m instances that is shattered by H .
- Then, to show that $VC(H) = m$, show that no subset of $m+1$ instances is shattered by H .

Alternative measures of classifier performance

In what cases can “accuracy on test set” be a misleading statistic?

Precision / Recall

- Confusion matrix for a classifier:

	Classified Positive	Classified Negative
Positive Examples	True positives (TP)	False negatives (FN)
Negative Examples	False positives (FP)	True negatives (TN)

Some performance measures

- **Accuracy:** proportion of classifications, over all the N examples, that were correct:

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

- **Recall (or true positive rate, or “detection rate”):** Proportion of positive examples that were classified correctly:

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

- **Precision :** Proportion of correct positive classifications over all positive classifications:

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

Example

<u>Test data</u>	<u>Correct Classification</u>	<u>Model's Classification</u>
X_1	T	T
X_2	T	F
X_3	F	T
X_4	F	F
X_5	F	T
X_6	F	F
X_7	F	F
X_8	F	T

Accuracy =

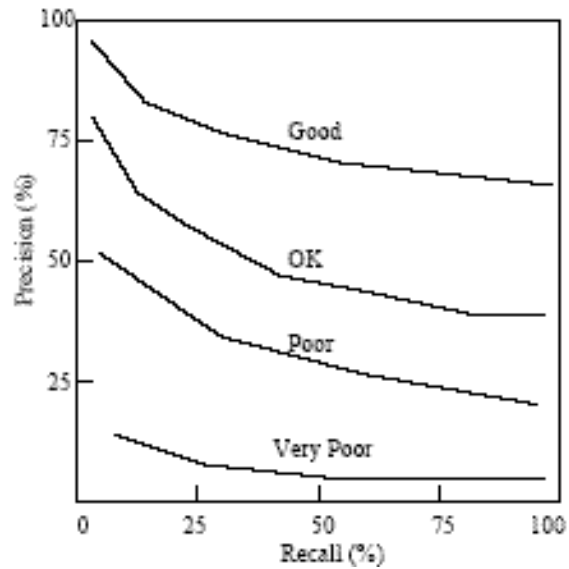
Recall =

Precision =

Interpretation of precision and recall

- Precision and recall are often plotted against one another, especially in “detection” applications (such as spam detection), when positive examples are sparse in the observed data.
- **Recall:** How often did the system correctly identify positive examples when it encountered them?
- **Precision:** How often did the system get positive classifications correct?
- How do these two measures trade off against one another?

Precision / Recall Curves



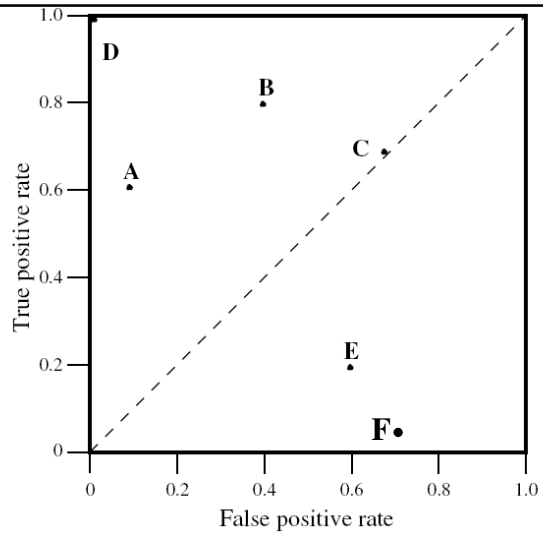
Receiver Operating Characteristic (ROC) Curves

- Alternative to precision/recall graphs
- Shows tradeoff between true positive rate and false positive rate.

$$\text{True positive rate} = \text{TP}/(\text{TP} + \text{FN})$$

$$\text{False positive rate} = \text{FP}/(\text{TN} + \text{FP})$$

Example ROC diagram for discrete-valued classifiers



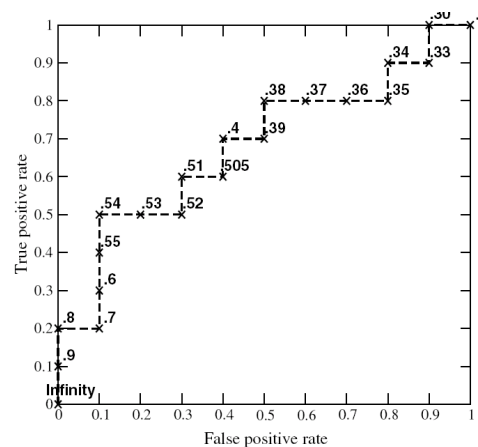
What is the interpretation of TPR = 0, FNR = 0?

How about TPR = 1, FNR = 1?

What is the ideal model? What is random guessing “true” with probability p ? Is F a good classifier?

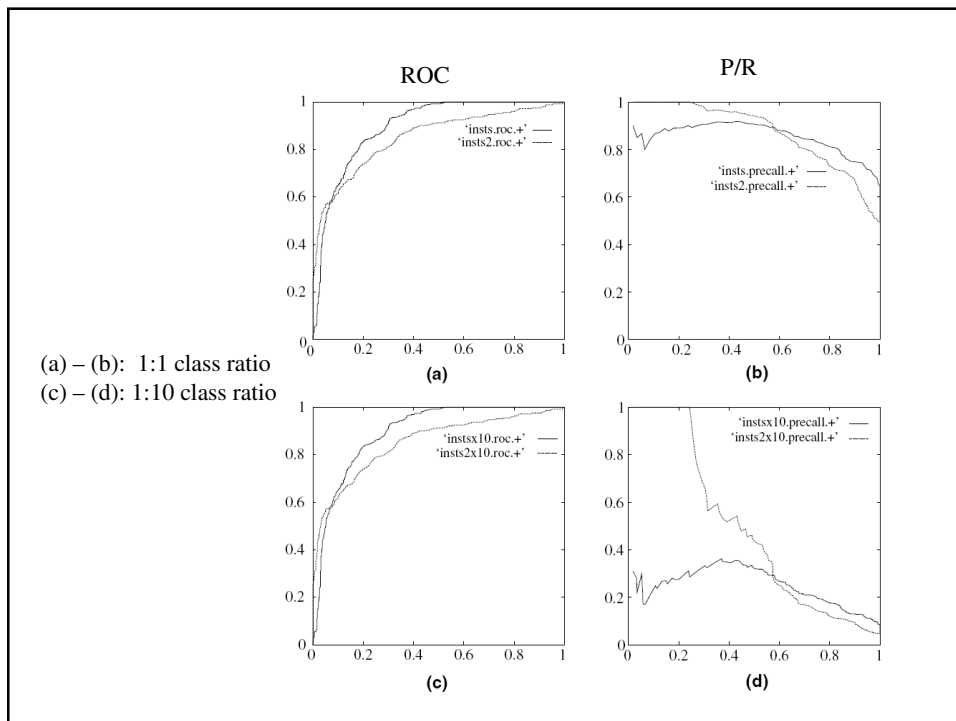
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Example ROC diagram for single continuous-valued classifier with varying threshold between ∞ and 0



What is accuracy of classifier at various points?

ROC curve shape is insensitive “operating conditions”,
i.e., to balance between positive and negative instances or
cost of misclassification.



How to generate a ROC curve from a test set?

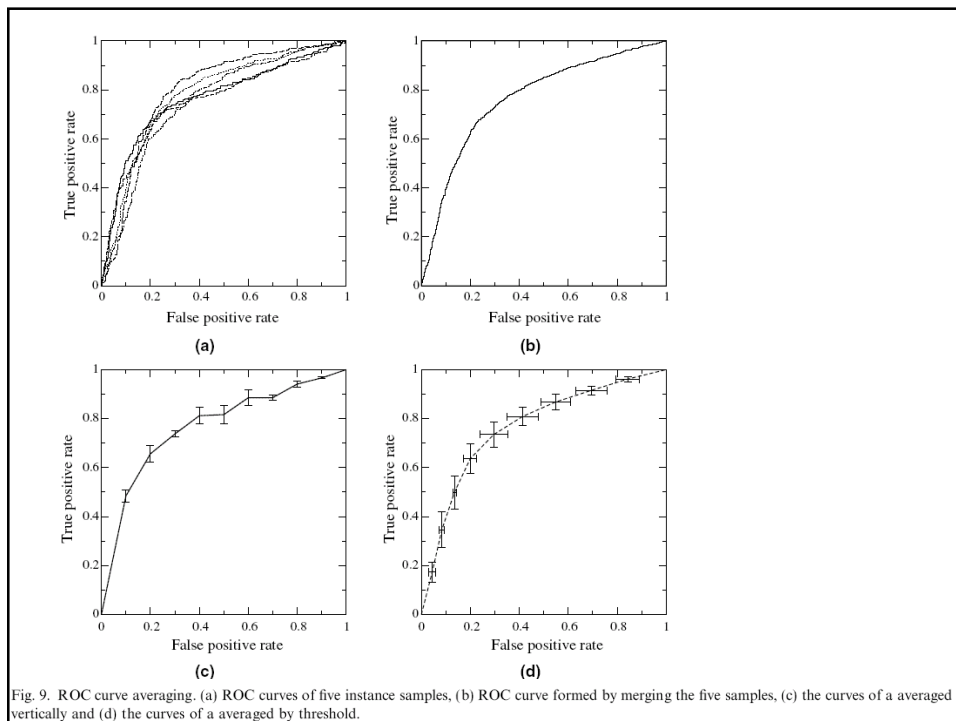
- See algorithm in reading.

Area under ROC curve (AUC)

- Summary statistic: Area under ROC curve (AUC) = probability that classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. (See algorithm in reading for calculating AUC.)
- AUC always between 0 and 1.
- AUC for random guessing?

Averaging ROC curves

- For cross-validation, want to average ROC curves for a classifier using test sets T_0, \dots, T_n
- For cross-validation



- Back to SVMs and kernel functions

Linear, polynomial, and Gaussian kernels

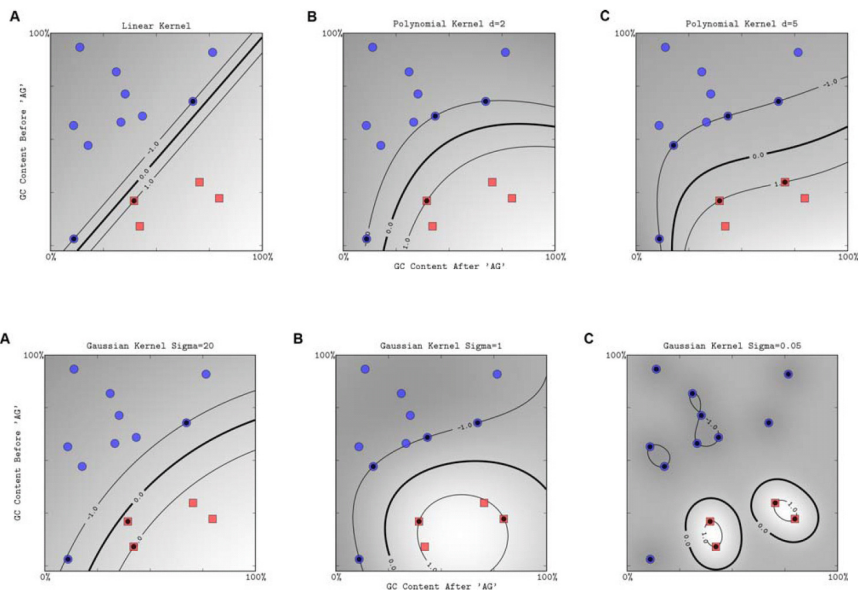


Table 1. SVM accuracy on the task of acceptor site recognition using polynomial and Gaussian kernels with different degrees d and widths σ .

Kernel	auROC
Linear	88.2%
Polynomial $d=3$	91.4%
Polynomial $d=7$	90.4%
Gaussian $\sigma=100$	87.9%
Gaussian $\sigma=1$	88.6%
Gaussian $\sigma=0.01$	77.3%

Accuracy is measured using the area under the ROC curve (auROC) and is computed using 5-fold cross-validation (cf. the section Running Example: Splice Site Recognition for details).
doi:10.1371/journal.pcbi.1000173.t001

Ensemble learning

- Example: answer questions (see previous slides)
- Bias/Variance decomposition
- Types of methods:

voting

bagging

boosting

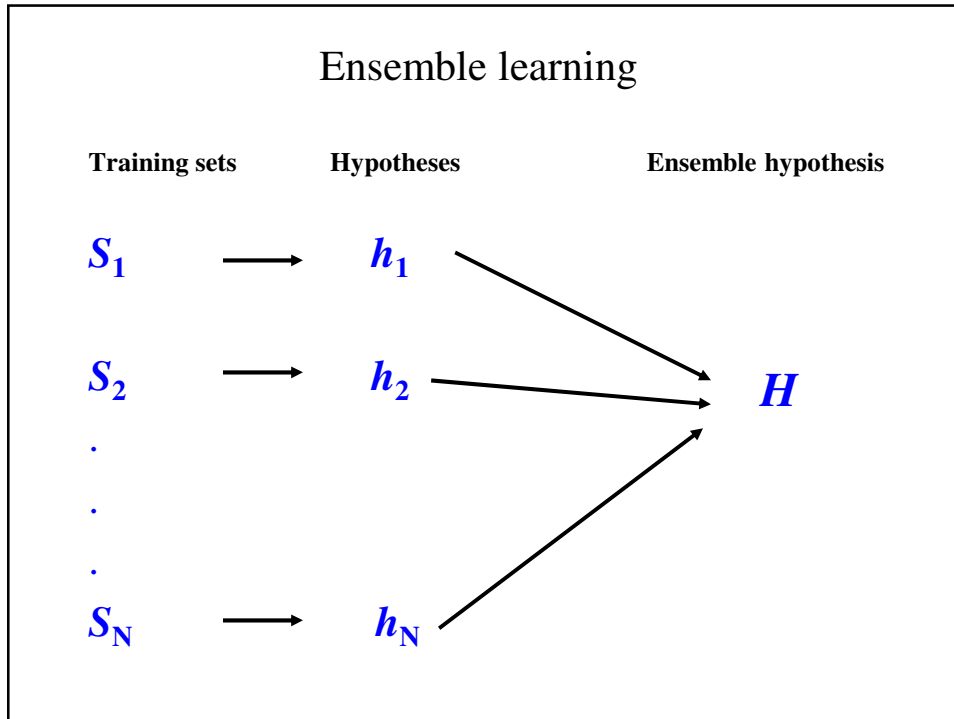
The power of voting

What movie won the Oscar for best picture in 2008?

- (a) Atonement
- (b) Juno
- (c) There Will Be Blood
- (d) No Country for Old Men
- (e) Michael Clayton

What year did Albert Einstein publish his special theory of relativity?

- (a) 1901
- (b) 1905
- (c) 1918
- (d) 1924
- (e) 1931



Advantages of ensemble learning

- Can be very effective at reducing generalization error!
(E.g., by voting.)
- Ideal case: the h_i have independent errors

Example 1: Given three hypotheses, h_1, h_2, h_3 .

Suppose each h_i has 40% generalization error, and assume errors are independent.

Now suppose we classify x as the majority vote of h_1, h_2 , and h_3 . What is probability that the classification is correct?

Example 2: Again, given three hypotheses, h_1, h_2, h_3 .

Suppose each h_i has 60% generalization error, and assume errors are independent.

Now suppose we classify x as the majority vote of h_1, h_2 , and h_3 . What is probability that the classification is correct?

In general, if hypotheses h_1, \dots, h_n all have generalization error $< 1/2$, what is probability that a majority vote will be correct?

Possible problems with ensemble learning

- Errors most likely are not independent
- Training time and classification time are increased by a factor of N .
- Hard to explain how ensemble hypothesis does classification.
- How to get enough data to create N separate data sets, S_1, \dots, S_N ?

Bayes Optimal Classifier

- Bayesian learning: “What is the most probable hypothesis given the training data?” (h_{MAP})
- But what we really want to know is, “what is the most probable classification of a new instance x , given the training data?”
- Could just apply h_{MAP} to x .
- But can do better via “ensemble” method (using all $h \in H$)

Example:

Consider $H = \{h_1, h_2, h_3\}$. Assume all have equal *a priori* probability.

Suppose $P(h_1|D) = 0.4$, $P(h_2|D) = 0.3$, and $P(h_3|D) = 0.3$.

$h_{\text{MAP}} = h_1$. However, ...

Let x be a new instance, with

$$h_1(x) = +$$

$$h_2(x) = -$$

$$h_3(x) = -$$

We have

$$P(x \text{ is a } + \text{ instance} \mid D) = 0.4$$

$$P(x \text{ is a } - \text{ instance} \mid D) = 0.6$$

- In general:
 - Let the possible classifications of x be
 $V = \{v_1, v_2, \dots, v_n\}$.
 - Let $P(v_j|D)$ be the probability that x is classified as v_j given training data D . Then

$$P(v_j|D) = \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

- Optimal classification of x is value v_j for which $P(v_j|D)$ is maximum:

$$\text{class}(x) = \operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

This is called a “Bayes optimal classifier”

- **Back to example:**

$$V = \{+, -\}$$

$$P(h_1 | D) = 0.4 \quad P(- | h_1) = 0 \quad P(+ | h_1) = 1$$

$$P(h_2 | D) = 0.3 \quad P(- | h_2) = 1 \quad P(+ | h_2) = 0$$

$$P(h_3 | D) = 0.3 \quad P(- | h_3) = 1 \quad P(+ | h_3) = 0$$

$$\text{class}(x) = \operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = ?$$

Bayes Optimal Classifier, continued

- “No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average.”
- However, method is almost always impractical — why?

- Three popular methods:

- **Voting:**

- Train classifier on n different training sets S_i to obtain n different classifiers h_i .
- For a new instance x , define $H(x)$ as:

$$H(x) = \sum_{i=1}^n w_i h_i(x)$$

– **Bagging (Breiman, 1990s):**

- To create S_i , create “bootstrap replicates” of original training set S

– **Boosting (Schapire & Freund, 1990s)**

- To create S_i , reweight examples in original training set S as a function of whether or not they were misclassified on the previous round.

Bagging
 (“Bootstrap Aggregation”)
L. Breiman, 1994

- **Problem with voting:** Usually don’t have enough data for N independent data sets S_i .
- **Solution:** Use “bootstrapping” as way to simulate this from single data set S .

Bootstrapping

- Create N training sets from original training set S as follows:
 - For each S_i , select m examples at random with replacement.
- Each of these new training sets has approximately the same distribution as that underlying S .

Stable versus unstable learning procedures

- Note that bagging will be most effective when small changes in S produce significant changes in h . (“Unstable learning procedures”.)
- “The evidence, both experimental and theoretical, is that bagging can push a good but unstable procedure a significant step towards optimality. On the other hand, it can slightly degrade the performance of stable procedures.” (Breiman, 1994)

Some empirical results on bagging (E. Bauer & R. Kohavi, 1999)

- Used 14 large data sets from UCI machine learning repository
- Algorithms:
 - (1) Decision tree learner
 - (2) Naive Bayes classifier

- **Results:**

1. Decision trees:

- Bagging reduced generalization error on all data sets.
- On some data sets, the generalization error was decreased dramatically.
- Error reduction was almost completely due to variance reduction.

2. Naive Bayes classifier:

- Also reduced generalization error, but by a smaller amount
- This is because naive Bayes classifier is more stable than decision tree learner —less variance in results among different training sets, so less error to reduce

Why does bagging work?

Two different explanations:

1. Reduces variance (Breiman, 1996)
 - Error is due to noise, bias, and variance
 - Noise: error in targets
 - Bias: systematic error in learner (average error over all training sets)
 - Variance: how sampling variation affects algorithm
 - Bootstrap sampling simulates sampling variance
 - Averaging over these samples can reduce error from variance, especially for unstable learning procedures

More formally (adapted from Tom Diettrich's slides):

Let Z be a random variable with probability distribution $P(Z)$

Let $\underline{Z} = E_p[Z]$ be the average value of Z .

Lemma: $E[(Z - \underline{Z})^2] = E[Z^2] - \underline{Z}^2$

$$\begin{aligned} E[(Z - \underline{Z})^2] &= E[Z^2 - 2 Z \underline{Z} + \underline{Z}^2] \\ &= E[Z^2] - 2 E[Z] \underline{Z} + \underline{Z}^2 \\ &= E[Z^2] - 2 \underline{Z}^2 + \underline{Z}^2 \\ &= E[Z^2] - \underline{Z}^2 \end{aligned}$$

Application to Bagging:

Let $\mathbf{x}_i \in \mathfrak{X}^m, y_i \in \mathfrak{Y}$.

Let h_1, \dots, h_N be the base hypotheses.

Let H be the aggregate hypothesis (average over h_i).

Let the error of hypothesis h be:

$$e(h, x_i) = (y_i - h(x_i))^2$$

Then:

$$e(H, \mathbf{x}_i) = \frac{\sum_{i=1}^N e(h_i, \mathbf{x}_i)}{N} - \frac{\sum_{i=1}^N (h_i(\mathbf{x}) - H(\mathbf{x}))^2}{N}$$

That is, squared error of average prediction equals average squared error of base hypotheses minus variance of the base hypotheses.

When will bagging reduce error?

- If $h_i \approx H$ (stable algorithm) for most i , won't get benefit of error reduction, since second term will be close to zero.
- The more highly variable the h_i are, the more improvement bagging can provide.
- "Bagging unstable classifiers usually improves them. Bagging stable classifiers is not a good idea."

2. Bayesian interpretation (P. Domingos, 1997)

Two hypothesis:

- a. Bagging reduces a classification learner's error rate because it more closely approximates an optimal Bayes classifier than a single hypothesis output by the learner.

Bayes optimal classifier:

$$v_{\text{MAP}}(\mathbf{x}) = \underset{\{v=+,-\}}{\operatorname{argmax}} \sum_{h_i \in H} P(v | h_i(\mathbf{x})) P(h_i(\mathbf{x}) | D)$$

Note that this is like voting, except we weight h_i 's value by its posterior probability.

In contrast, bagging assigns uniform weight to all h_i 's

- b. Bagging reduces a classification learner's error rate because it changes the learner's hypothesis space and/or prior distribution to one that better fits the domain.

(I.e., searches new hypothesis space of aggregate models.)

E.g., Bagging is less biased in favor of *simple* models. It thus changes "prior" probability over space of models.

Domingos finds empirical evidence for (b) but not (a).