

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- **Naïve Bayes for text classification**
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- **Support vector machines**
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

Introduction

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are **linear classifiers** that find a hyperplane to separate **two class** of data, positive and negative.
- **Kernel functions** are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best classifier for text classification.

Basic concepts

- Let the set of **training examples** D be

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_r, y_r)\},$$

where $\mathbf{x}_i = (x_1, x_2, \dots, x_n)$ is an **input vector** in a real-valued space $X \subseteq R^n$ and y_i is its **class label** (output value), $y_i \in \{1, -1\}$.

1: positive class and -1: negative class.

- SVM finds a linear function of the form (\mathbf{w} : weight vector)

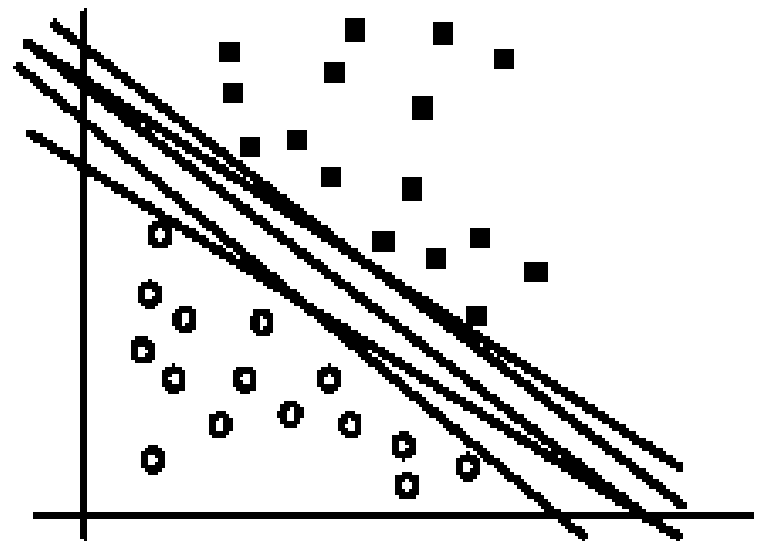
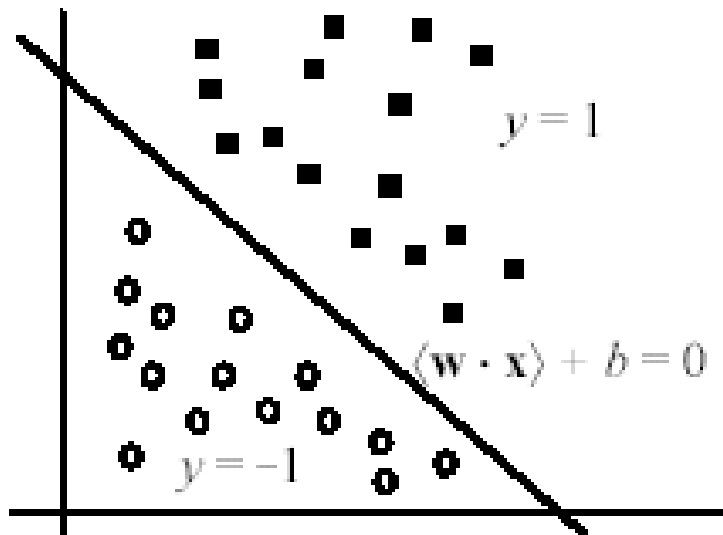
$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$
$$y_i = \begin{cases} 1 & \text{if } \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & \text{if } \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

The hyperplane

- The hyperplane that separates positive and negative training data is

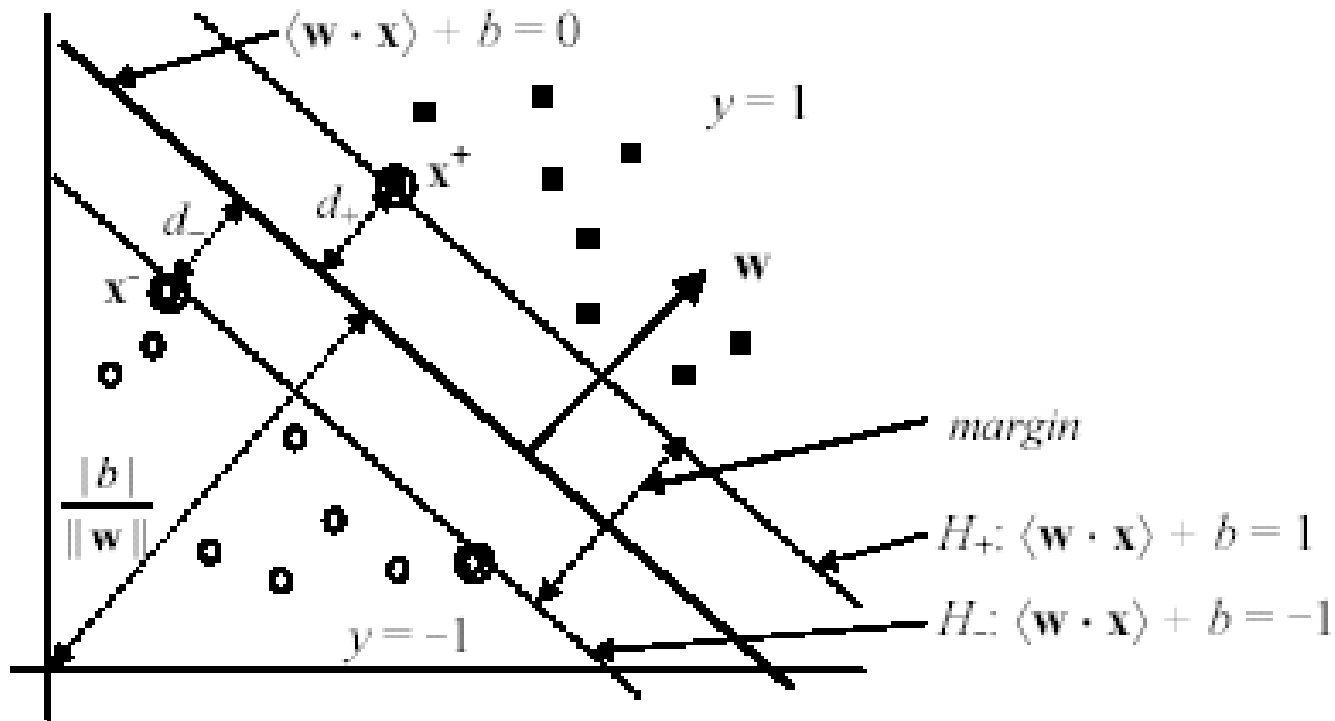
$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$$

- It is also called the **decision boundary (surface)**.
- So many possible hyperplanes, which one to choose?



Maximal margin hyperplane

- SVM looks for the separating hyperplane with the largest margin.
- Machine learning theory says this hyperplane minimizes the error bound



Linear SVM: separable case

- Assume the data are linearly separable.
- Consider a positive data point $(\mathbf{x}^+, 1)$ and a negative $(\mathbf{x}^-, -1)$ that are closest to the hyperplane

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0.$$

- We define two parallel hyperplanes, H_+ and H_- , that pass through \mathbf{x}^+ and \mathbf{x}^- respectively. H_+ and H_- are also parallel to $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$.

$$H_+: \quad \langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$$

$$H_-: \quad \langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1$$

$$\text{such that} \quad \begin{array}{ll} \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 & \text{if } y_i = 1 \\ \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 & \text{if } y_i = -1, \end{array}$$

Compute the margin

- Now let us compute the distance between the two **margin hyperplanes** H_+ and H_- . Their distance is the **margin** ($d_+ + d_-$ in the figure).
- Recall from vector space in algebra that the (perpendicular) **distance** from a point \mathbf{x}_i to the hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ is:

$$\frac{|\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} \quad (36)$$

where $\|\mathbf{w}\|$ is the norm of \mathbf{w} ,

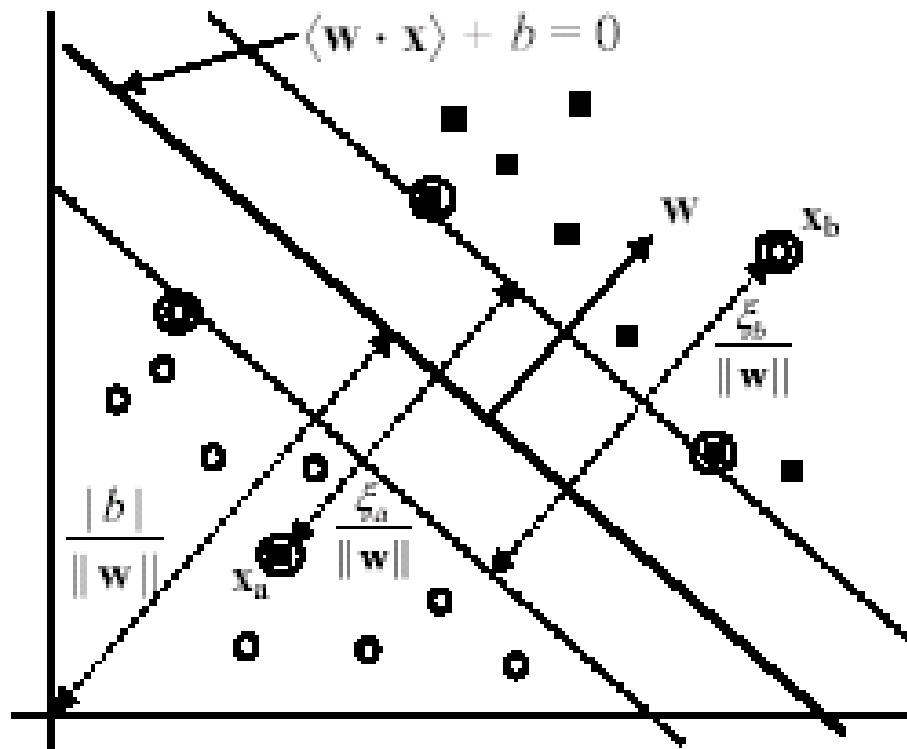
$$\|\mathbf{w}\| = \sqrt{\langle \mathbf{w} \cdot \mathbf{w} \rangle} = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2} \quad (37)$$

Linear SVM: Non-separable case

- Linear separable case is the ideal situation.
- Real-life data may have noise or errors.
 - Class label incorrect or randomness in the application domain.
- With noisy data, the constraints may not be satisfied. Then, no solution!

Geometric interpretation

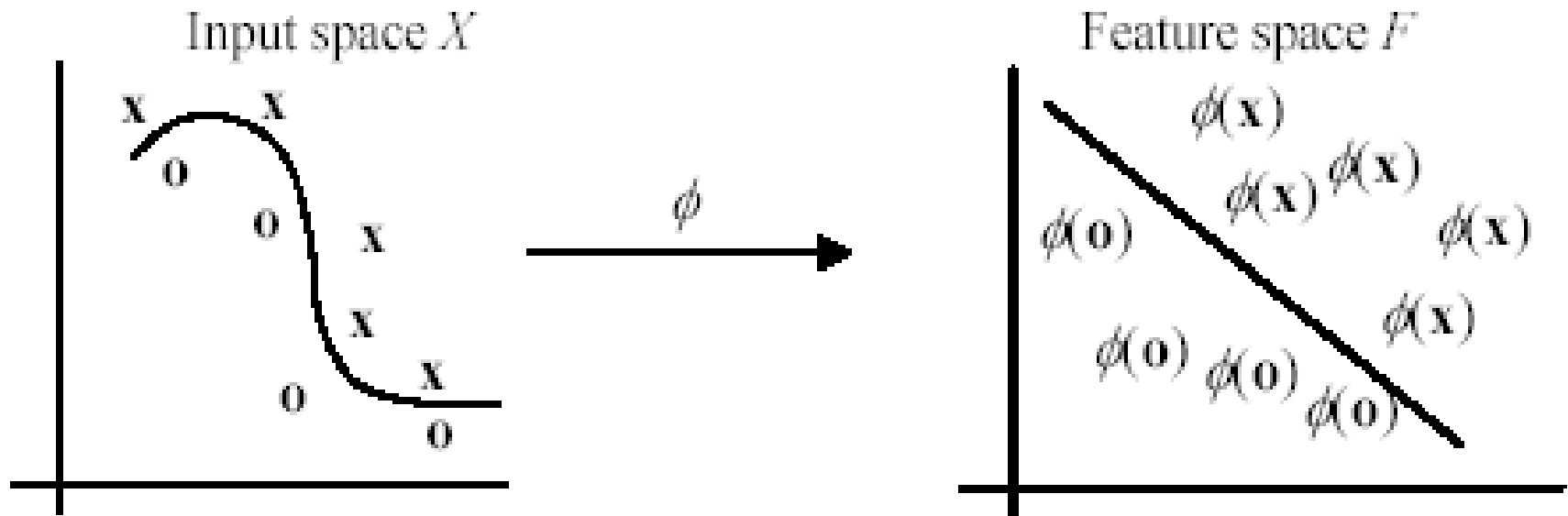
- Two error data points \mathbf{x}_a and \mathbf{x}_b (circled) in wrong regions



How to deal with nonlinear separation?

- The SVM formulations require linear separation.
- Real-life data sets may need nonlinear separation.
- To deal with nonlinear separation, the same formulation and techniques as for the linear case are still used.
- We only transform the input data into another space (usually of a much higher dimension) so that
 - a linear decision boundary can separate positive and negative examples in the transformed space,
- The transformed space is called the **feature space**. The original data space is called the **input space**.

Geometric interpretation



- In this example, the transformed space is also 2-D. But usually, the number of dimensions in the feature space is much higher than that in the input space

Some other issues in SVM

- SVM works only in a real-valued space.
 - For a categorical attribute, we need to convert its categorical values to numeric values.
- SVM does only two-class classification.
 - For multi-class problems, some strategies can be applied, e.g., one-against-rest, and error-correcting output coding.
- The hyperplane produced by SVM is hard to understand by human users.
 - SVM is commonly used in applications that do not required human understanding.

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- **K-nearest neighbor**
- Ensemble methods: Bagging and Boosting
- Summary

k-Nearest Neighbor Classification

- Unlike all the previous learning methods, **kNN does not build model from the training data.**
- To classify a test instance d , define k -neighborhood P as k nearest neighbors of d
- Count number n of training instances in P that belong to class c_j
- Estimate $\Pr(c_j|d)$ as n/k
- No training is needed. Classification time is linear in training set size for each test case.

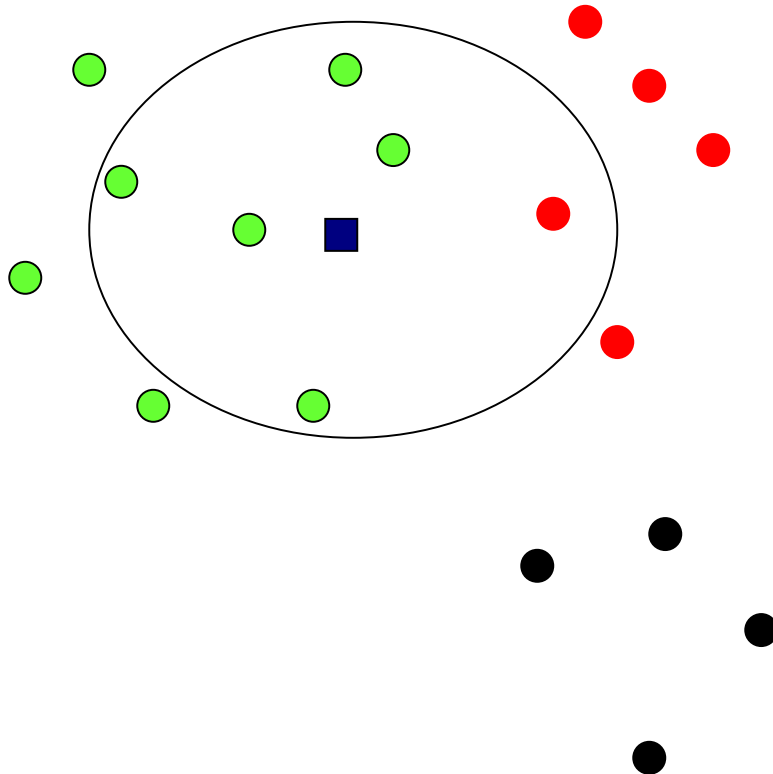
kNN Algorithm

Algorithm $kNN(D, d, k)$

- 1 Compute the distance between d and every example in D ;
- 2 Choose the k examples in D that are nearest to d , denote the set by $P (\subseteq D)$;
- 3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- **Distance function** is crucial, but depends on applications.

Example: $k=6$ (6NN)



- Government
- Science
- Arts

A new point ■
 $\text{Pr}(\text{science} | \text{■})?$

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- **Ensemble methods: Bagging and Boosting**
- Summary

Combining classifiers

- So far, we have only discussed individual classifiers, i.e., how to build them and use them.
- Can we combine multiple classifiers to produce a better classifier?
- Yes, sometimes
- We discuss two main algorithms:
 - Bagging
 - Boosting

Bagging

- Breiman, 1996
- Bootstrap Aggregating = Bagging
 - Application of **bootstrap sampling**
 - **Given:** set D containing m training examples
 - Create a sample $S[i]$ of D by drawing m examples at random *with replacement* from D
 - $S[i]$ of size m : expected to leave out 0.37 of examples from D

Bagging (cont...)

■ Training

- Create k bootstrap samples $S[1], S[2], \dots, S[k]$
- Build a distinct classifier on each $S[i]$ to produce k classifiers, using the same learning algorithm.

■ Testing

- Classify each new instance by voting of the k classifiers (equal weights)

Bagging Example

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Bagging (cont ...)

■ When does it help?

□ When learner is unstable

- Small change to training set causes large change in the output classifier
- True for decision trees, neural networks; not true for k -nearest neighbor, naïve Bayesian, class association rules

□ Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

Boosting

- A family of methods:
 - We only study **AdaBoost** (Freund & Schapire, 1996)
- **Training**
 - Produce a sequence of classifiers (the same base learner)
 - Each classifier is dependent on the previous one, and focuses on the previous one's errors
 - Examples that are incorrectly predicted in previous classifiers are given higher weights
- **Testing**
 - For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

AdaBoost

Weighted training set

(\mathbf{x}_1, y_1, w_1)

(\mathbf{x}_2, y_2, w_2)

...

(\mathbf{x}_n, y_n, w_n)

Non-negative weights
sum to 1



called a weaker classifier



- Build a classifier h_t whose accuracy on training set $> \frac{1}{2}$ (better than random)

Change weights



AdaBoost algorithm

Algorithm AdaBoost.M1

Input: sequence of m examples $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \dots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i .

Do for $t = 1, 2, \dots, T$:

1. Call **WeakLearn**, providing it with the distribution D_t .
2. Get back a hypothesis $h_t : X \rightarrow Y$.
3. Calculate the error of h_t : $\epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$.

If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.

4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$h_{\text{final}}(x) = \arg \max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}.$$

Bagging, Boosting and C4.5

C4.5's mean error rate over the 10 cross-validation.

Bagged C4.5 vs. C4.5.

Boosted C4.5 vs. C4.5.

Boosting vs. Bagging

	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
<i>average</i>	<i>15.66</i>	<i>14.11</i>		<i>.905</i>	<i>13.36</i>		<i>.847</i>		<i>.930</i>

Does AdaBoost always work?

- The actual performance of boosting depends on the data and the base learner.
 - It requires the base learner to be unstable as bagging.
- Boosting seems to be susceptible to noise.
 - When the number of outliers is very large, the emphasis placed on the hard examples can hurt the performance.

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- **Summary**

Summary

- Applications of supervised learning are in almost any field or domain.
- We studied many classification techniques.
- There are still many other methods, e.g.,
 - Bayesian networks
 - Neural networks
 - Genetic algorithms
 - Fuzzy classification

This large number of methods also show the importance of classification and its wide applicability.

- It remains an active research area.