Classification Applet

http://www.cs.technion.ac.il/~rani/LocBoost/

Multiclass classification

1. Build a classifier for each class, where the training set consists of the set of documents in the class (positive labels) and its complement (negative labels).
2. Given the test document, apply each classifier separately.
3. Assign the document to the class with
   - the maximum score,
   - the maximum confidence value,
   - or the maximum probability.

Confusion matrix

assigned class / true class	*money-fx*	*trade*	*interest*	*wheat*	*corn*	*grain*
money-fx	95	0	10	0	0	0
trade	1	1	90	0	1	0
interest	13	0	0	0	0	0
wheat	0	0	1	34	3	7
corn	1	0	2	13	26	5
grain	0	0	2	14	5	10

▶ **Table 14.5** A confusion matrix for Reuters-21578. For example, 14 documents from *grain* were incorrectly assigned to *wheat*. Adapted from Picca et al. (2006).

Precision of classifier?

Recall of classifier?

Support Vector Machines

Textbook, Chapter 15

April 27, 2010

Example

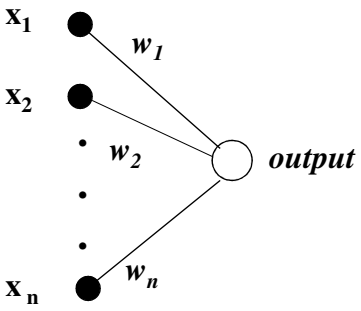
- UCI Spam data: http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/

"A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree. Neither can generalize well." (C. Burges, *A tutorial on support vector machines for pattern recognition).*

6

Perceptrons

- Invented by Rosenblatt, 1950s
- Single-layer feedforward network: $y = \sum_{i=1}^{n} w_i x_i + b$



$$o = \begin{cases} 1 \text{ if } y > 0 \\ 0 \text{ if } y = 0 \\ -1 \text{ otherwise} \end{cases}$$

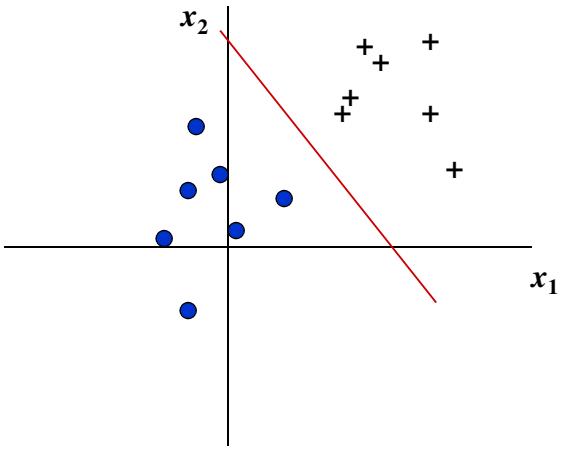
This can be used for classification:

+1 = positive, -1 = negative,

0 = "uncertain"

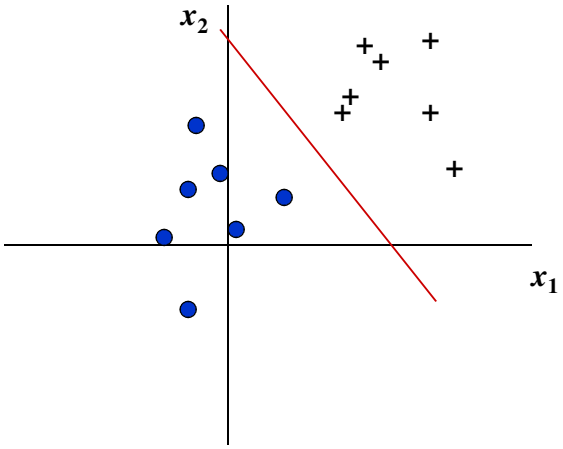
- Consider example with two inputs, x_1, x_2.

Can view trained network as defining "separation line". What is its equation?



- Consider example with two inputs, x_1, x_2.

Can view trained network as defining "separation line". What is its equation?



We have:

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

- This is a "linear discriminant function" or "linear decision surface".
- Weights determine slope and bias determines offset.
- Can generalize to n-dimensions:

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = 0$$

  - Decision surface is an n-1 dimensional "hyperplane".

In-class Exercises:

(a) Find weights for a two-input perceptron that computes $f(x_1, x_2) = x_1 \wedge x_2$. Sketch the separation line defined by this perceptron.

(b) Do the same, but for $x_1 \vee x_2$.

Support Vector Machines

- Assume a linearly separable binary classification problem.
  - Instances are represented by vector $\mathbf{x} \in \Re^n$.
  - Training examples:

$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m) \mid (\mathbf{x}_i, y_i) \in \Re^n \times \{+1, -1\}$

  - Hypothesis: A function h: $\Re^n \rightarrow \{+1, -1\}$.

$$h(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) = \text{sgn}\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

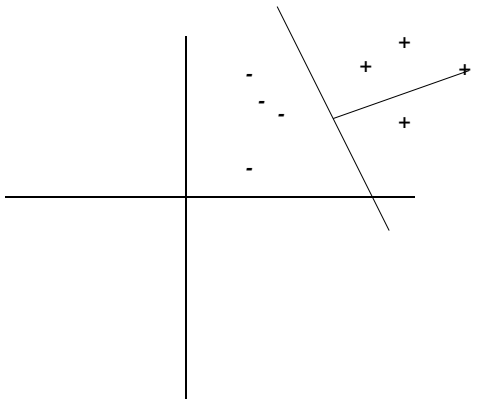
- Positive and negative instances are to be separated by the hyperplane

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

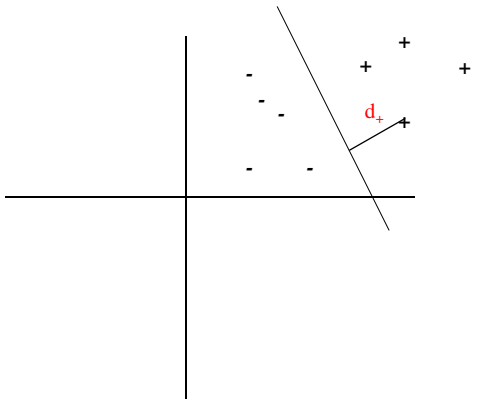
- But: Which separating hyperplane is optimal?

Definition of Margin

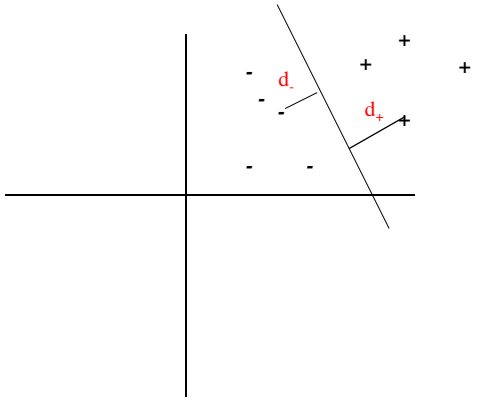
- The margin of an instance with respect to a hyperplane is the distance from that point to the hyperplane:



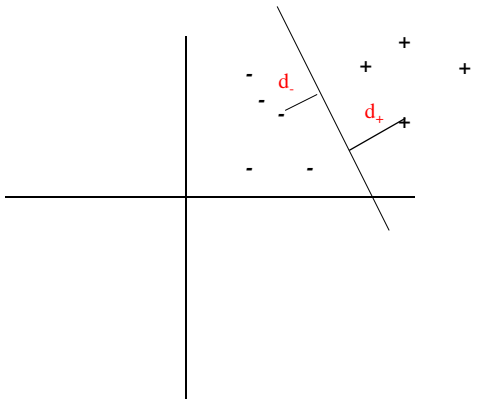
- The margin of the positive examples, d_+ with respect to that hyperplane, is the shortest distance from a positive example to the hyperplane:



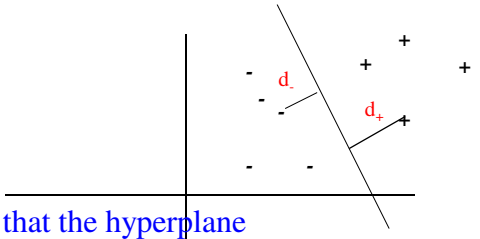
- The margin of the negative examples, d_- with respect to that hyperplane, is the shortest distance from a negative example to the hyperplane:



- The margin of the training set S with respect to the hyperplane is $d_+ + d_-$.



- The margin of the training set S with respect to the hyperplane is $d_+ + d_-$.



Vapnik showed that the hyperplane maximizing the margin of S will have minimal "VC dimension" (complexity) in the set of all consistent hyperplanes, and will thus be optimal.

This is an optimization problem!

- Note that the hyperplane is defined as

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

- To make the math easier, we will put a restriction on the set of such hyperplanes, by also requiring that:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ for positive instances } (y_i = +1)$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ for negative instances } (y_i = -1)$$

- Consider the points $\mathbf{x}_i$ for which these are equalities:

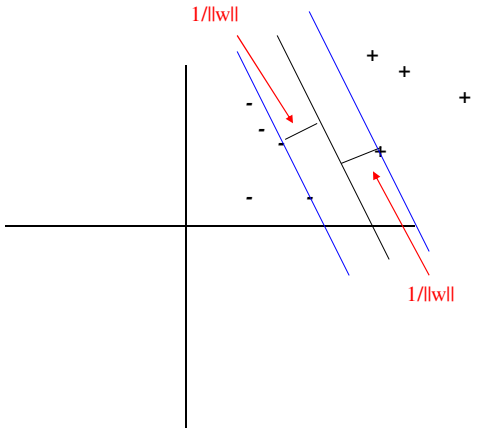
$$\mathbf{x}_i \cdot \mathbf{w} + b = +1 \text{ for positive instances } (y_i = +1)$$
$$\mathbf{x}_i \cdot \mathbf{w} + b = -1 \text{ for negative instances } (y_i = -1) \quad \textbf{(1)}$$

- What is the margin of these points?

$$d_+ = d_- = \frac{1}{\|\mathbf{w}\|} = \frac{1}{\sqrt{w_1^2 + ... + w_n^2}}$$

- Margin γ of training set:

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$



- The points $\mathbf{x}_i$ for which **(1)** holds are called *support vectors*.

- Goal is to maximize margin γ:

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$

- To do this, we need to minimize $\|\mathbf{w}\|^2$, subject to constraints

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ for positive instances } (y_i = +1)$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ for negative instances } (y_i = -1)$$

- Goal is to find hyperplane that gives the maximum margin of $2 / \|\mathbf{w}\|$, by minimizing $\|\mathbf{w}\|^2$, subject to the joint constraints given above, plus the constraint that the training error should be as small as possible.

- Note that this is a constrained optimization problem, that can be solved via a "constrained quadratic programming" method.

- Support vector machines are a way of finding a ($\mathbf{w}$, b) to accomplish this.

- It turns out that **w** can be expressed as a linear combination of a small subset of the training examples: those that lie exactly on margin (minimum distance to hyperplane).

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$$

- These training examples are called "support vectors". They carry all relevant information about the classification problem.

- The result of the SVM training algorithm (involving solving a quadratic programming problem), is the α_i's and the $\mathbf{x}_i$'s.

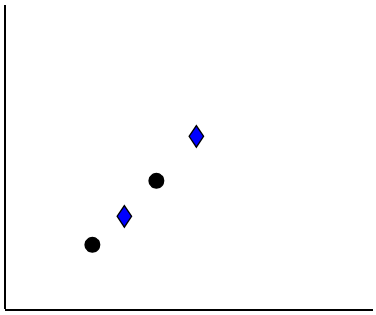
- For a new example **x**, We can now classify **x** using the support vectors:

$$\text{class}(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i (\mathbf{x} \cdot \mathbf{x}_i) + b\right)$$

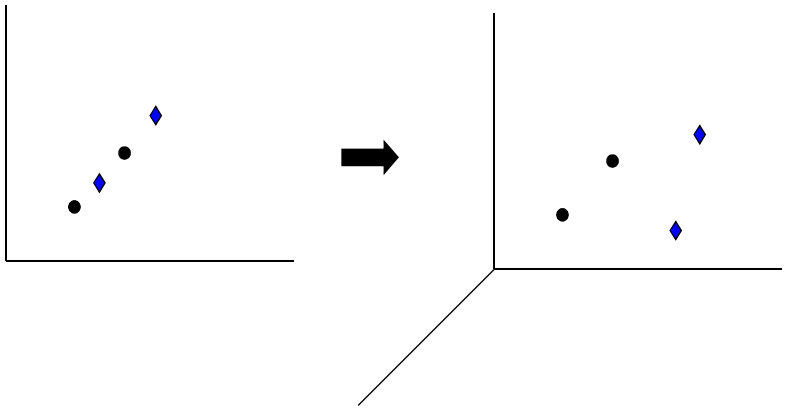
- This is the resulting SVM classifier.

Non-linearly separable training examples

- What if the training examples are not linearly separable?



- Use old trick: find a function that maps points to a higher dimensional space ("feature space") in which they are linearly separable, and do the classification in that higher-dimensional space.



Need to find a function Φ that will perform such a mapping:

$$\Phi: \Re^n \rightarrow F$$

Then can find hyperplane in higher dimensional feature space F, and do classification using that hyperplane in higher dimensional space.

- **Problem**:
  - Recall that classification of instance **x** is expressed in terms of dot products of **x** and support vectors.

$$\text{Class}(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i (\mathbf{x} \cdot \mathbf{x}_i) + b\right)$$

  - The quadratic programming problem of finding the support vectors and coefficients also depends only on dot products between training examples, rather than on the training examples outside of dot products.

- So if each $\mathbf{x}_i$ is replaced by $\Phi(\mathbf{x}_i)$ in these procedures, we will have to calculate a lot of dot products, $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$

- But in general, if the feature space F is high dimensional, $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ will be expensive to compute.

- Also $\Phi(\mathbf{x})$ can be expensive to compute

- **Second trick:**

  - Suppose that there were some magic function,

    $$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

    such that k is cheap to compute even though $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ is expensive to compute.

  - Then we wouldn't need to compute the dot product directly; we'd just need to compute k during both the training and testing phases.

  - The good news is: such k functions exist! They are called "kernel functions", and come from the theory of integral operators.

Example: polynomial kernel:

Suppose $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$.

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^2$$

Let $\Phi(\mathbf{x}) = \left(x_1^{\,2}, \sqrt{2}x_1x_2, x_2^2\right)$.

Then :

$$\Phi(\mathbf{x})\cdot\Phi(\mathbf{y}) = \left(\begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}\cdot\begin{pmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{pmatrix}\right)$$

$$= \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\cdot\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right)^2 = (\mathbf{x}\cdot\mathbf{y})^2 = k(\mathbf{x},\mathbf{y})$$

Recap of SVM algorithm

Given training set

$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m) \mid (\mathbf{x}_i, y_i) \in \Re^n \times \{+1, -1\}$

1. Choose a map $\Phi: \Re^n \rightarrow F$, which maps $\mathbf{x}_i$ to a higher dimensional feature space. (Solves problem that X might not be linearly separable in original space.)

2. Choose a cheap-to-compute kernel function

   $$k(\mathbf{x},\mathbf{z}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$$

   (Solves problem that in high dimensional spaces, dot products are very expensive to compute.)

3. Map all the $\mathbf{x}_i$'s to feature space F by computing $\Phi(\mathbf{x}_i)$.

4. Apply quadratic programming procedure (using the kernel function k) to find a hyperplane (**w**, *b*), such that

$$\mathbf{w} = \sum_i \alpha_i \, \Phi(\boldsymbol{x}_i)$$

where the $\Phi(\mathbf{x}_i)$'s are support vectors, the α_i's are coefficients, and w_0 is a bias, such that (**w**,*b*) is the hyperplane maximizing the margin of *S* in *F*.

- Now, given a new instance, **x**, find the classification of **x** by computing

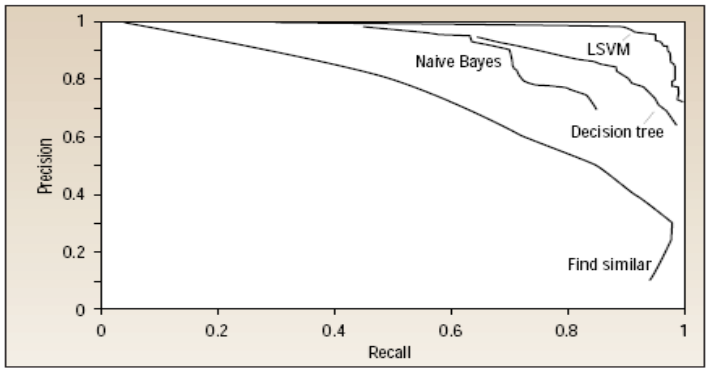
$$\text{class}(\boldsymbol{x}) = \text{sgn}\left(\sum_i \alpha_i (\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}_i)) + b) \right)$$

$$= \text{sgn}\left(\sum_i \alpha_i \, k(\boldsymbol{x}, \boldsymbol{x}_i) + b) \right)$$

Example: Applying SVMs to text classification (Dumais et al., 1998)

- Used Reuters collection of news stories.
- Question: Is a particular news story in the category "grain" (i.e., about grain, grain prices, etc.)?
- Training examples: Vectors of features such as appearance or frequency of key words. (Similar to our spam-classification task.)
- Resulting SVM: weight vector defined in terms of support vectors and coefficients, plus threshold.

Results



Precision: Correctly classified positives / Number of positive classifications

Recall: Correctly classified positives / All actual positives

Can vary decision threshold to produce higher precision or higher recall.

Generally want high values for both. SVM gives best results over all these algorithms.

	NB	Roc-chio	Dec. Trees	kNN	linear SVM $C = 0.5$	linear SVM $C = 1.0$	rbf-SVM $\sigma \approx 7$
earn	96.0	96.1	96.1	97.8	98.0	98.2	98.1
acq	90.7	92.1	85.3	91.8	95.5	95.6	94.7
money-fx	59.6	67.6	69.4	75.4	78.8	78.5	74.3
grain	69.8	79.5	89.1	82.6	91.9	93.1	93.4
crude	81.2	81.5	75.5	85.8	89.4	89.4	88.7
trade	52.2	77.4	59.2	77.9	79.2	79.2	76.6
interest	57.6	72.5	49.1	76.7	75.6	74.8	69.1
ship	80.9	83.1	80.9	79.8	87.4	86.5	85.8
wheat	63.4	79.4	85.5	72.9	86.6	86.8	82.4
corn	45.2	62.2	87.7	71.4	87.5	87.8	84.6
microavg.	72.3	79.9	79.4	82.6	86.7	87.5	86.4

► **Table 15.2** SVM classifier break-even F_1 from (Joachims 2002a, p. 114). Results are shown for the 10 largest categories and for microaveraged performance over all 90 categories on the Reuters-21578 data set.

Demo:
Text classification using *SVM Light*

Data sets in Text Classification

- Training set
- Validation set
- Test set
- Notion of “Cross Validation”

- How would you do multi-class classification with SVMs?

Spam Classification Data for Homework