6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

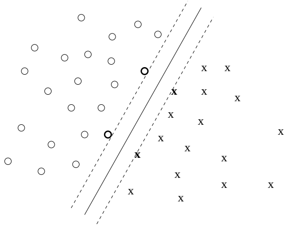
LECTURE 9 (October 5, 2000)

Topics

- Support vector machines
  - optimal hyperplane
  - kernel function
- Complexity
  - cross-validation result
  - shattering, VC-dimension

"Optimal" hyperplane

- Let's assume for simplicity that the classification problem is *linearly separable*



- Maximum margin hyperplane is maximally removed from all the training examples
- This hyperplane can be defined on the basis of only a few training examples called *support vectors*

"Optimal" hyperplane cont'd

- Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where the labels are binary ± 1
- Linear separator:

$$\begin{aligned} f(\mathbf{x}; \mathbf{w}, w_0) &= w_0 + x_1 w_1 + \ldots x_d w_d \\ &= w_0 + \mathbf{w}^T \mathbf{x} \end{aligned} \tag{1}$$

- We can try to find the "optimal" hyperplane by requiring that the sign of the decision boundary $[w_0 + \mathbf{w}^T \mathbf{x}]$ (clearly) agrees with the training labels

$$y_i \left[w_0 + \mathbf{w}^T \mathbf{x}_i \right] - 1 \geq 0, \quad i = 1, \ldots, n \tag{2}$$

BUT:
- this is just an alternative definition of linear separability
- there are multiple answers
- larger values of $\mathbf{w}, w_0$ would yield larger separation.

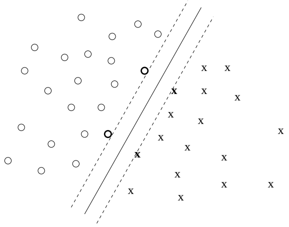
Support vector machine

- We minimize

$$\|\mathbf{w}\|^2/2 = \mathbf{w}^T\mathbf{w}/2 = \sum_{j=1}^{d} w_i^2/2 \tag{3}$$

subject to the classification constraints

$$y_i\,[w_0 + \mathbf{w}^T\mathbf{x}_i] - 1 \geq 0, \quad i = 1, \ldots, n \tag{4}$$



- Only a few of the classification constraints are relevant
$\Rightarrow$ support vectors

Support vector machine cont'd

- We find the optimal setting of $\{w_0, \mathbf{w}\}$ by introducing Lagrange multipliers $\alpha_i \geq 0$ for the inequality constraints
- We *minimize*

$$J(\mathbf{w}, w_0, \alpha) = \|\mathbf{w}\|^2/2 - \sum_{i=1}^{n} \alpha_i \left(y_i\,[w_0 + \mathbf{w}^T\mathbf{x}_i] - 1 \right) \tag{5}$$

with respect to $\mathbf{w}, w_0$. $\{\alpha_i\}$ make sure that the classification constraints are indeed satisfied.

For fixed $\{\alpha_i\}$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, w_0, \alpha) &= \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \\ \frac{\partial}{\partial w_0} J(\mathbf{w}, w_0, \alpha) &= -\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{6}$$

Solution

- Solving for $\{w_0, \mathbf{w}\}$ leaves us with the following (dual) optimization problem over the Lagrange multipliers:

We *maximize*

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \tag{7}$$

subject to the constraints

$$\alpha_i \geq 0, \quad i = 1, \ldots, n, \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{8}$$

(For non-separable problems we have to limit $\alpha_i \leq C$)

- This is a *quadratic programming problem*

Support vector machines

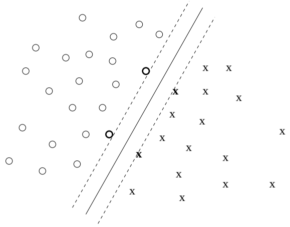
- The resulting parameter vector $\hat{\mathbf{w}}$ is a weighted combination of the training examples:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i \tag{9}$$

where the "weight" $\alpha_i = 0$ for all but the *support vectors SV*

- The decision boundary has an interpretable form

$$\hat{\mathbf{w}}^T\mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i\, y_i\, (\mathbf{x}_i^T \mathbf{x}) + w_0 \tag{10}$$



Interpretation of support vector machines

- To use support vector machines we have to specify only the inner products (or *kernel*) between the examples ($\mathbf{x}_i^T\mathbf{x}$)
- The weights $\{\alpha_i\}$ associated with the training examples are solved by enforcing the classification constraints.

  $\Rightarrow$ sparse solution
- We make decisions by comparing each new example $\mathbf{x}$ with **only** the support vectors $\{\mathbf{x}_i\}_{i\in SV}$:

$$\hat{y} = \text{sign}\left(\sum_{i\in SV} \hat{\alpha}_i\, y_i\, (\mathbf{x}_i^T\mathbf{x}) + w_0\right) \tag{11}$$

Kernel function and feature mapping

- Both the optimization problem (setting of Lagrange multipliers) and the decision boundary depends on the examples $\mathbf{x}$ only in terms of their inner products or *kernel*:

$$K(\mathbf{x},\mathbf{x}_i) = (\mathbf{x}^T\mathbf{x}_i) \tag{12}$$

- If each example $\mathbf{x}$ is first mapped into a feature vector $\phi(\mathbf{x})$,

$$\mathbf{x} \rightarrow \phi(\mathbf{x}) \tag{13}$$

  and we find the "optimal" hyperplane in the feature space, we can express the solution in terms of the following *kernel*

$$K(\mathbf{x},\mathbf{x}_i) = \phi(\mathbf{x})^T\phi(\mathbf{x}_i) \tag{14}$$

Kernel function and feature mapping

- Example: consider a mapping from $\mathbf{x} = [x_1\ x_2]^T$ to a longer feature vector $\phi(\mathbf{x})$:

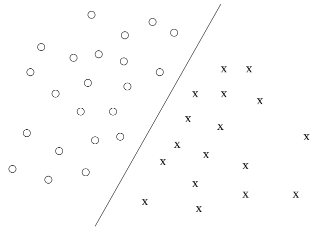
$$\mathbf{x} \rightarrow \phi(\mathbf{x}) = [x_1^2\ \ x_2^2\ \ \sqrt{2}x_1x_2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2\ \ 1]^T \tag{15}$$

  Finding the "optimal" hyperplane in the feature space gives us the following *kernel function*

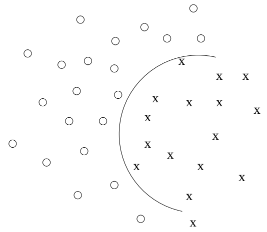
$$\begin{aligned} K(\mathbf{x},\mathbf{x}') &= \phi(\mathbf{x})^T\phi(\mathbf{x}') \\ &= x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1x_1' + 2x_2x_2' + 1 \\ &= (1 + x_1x_1' + x_2x_2')^2 \\ &= \left(1 + (\mathbf{x}^T\mathbf{x}')\right)^2 \end{aligned} \tag{16}$$

- Thus we can perform linear separation in higher dimensional feature space *implicitly* by defining a non-linear kernel function

Kernel functions



Linear separator in the *feature space*



Non-linear separator in the *input space*

Other examples of kernel functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}') \tag{17}$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + (\mathbf{x}^T \mathbf{x}')\right)^p \tag{18}$$

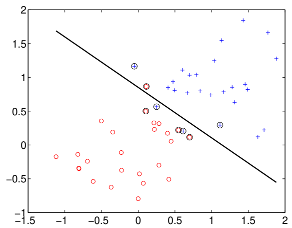
where $p = 2, 3, \ldots$. To get the feature vectors we concatenate all p^{th} order polynomial terms of the components of $\mathbf{x}$ (weighted appropriately)

- **Radial basis kernel**

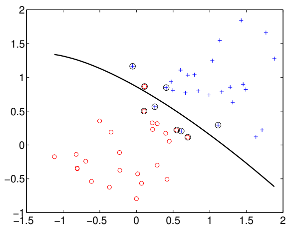
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{19}$$

In this case the feature space consists of functions and results in a *non-parametric* classifier.

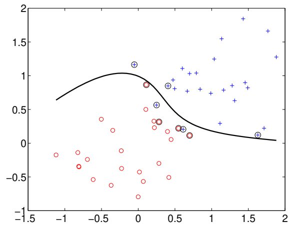
SVM examples



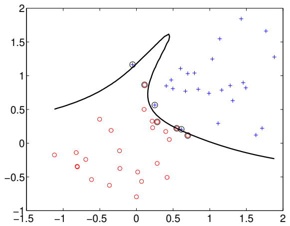
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

SVM examples

- Digit recognition example (16x16 grayscale pixel images)

Method	error %	
SVM (4^{th} order polynomial)	1.1	
LeNet 1 (neural network)	1.7	(handcrafted)
LeNet 4 (neural network)	1.1	(handcrafted)
Tangent distance (template matching)	0.7	(handcrafted)

- Document classification, etc.

Complexity

- Example: even for small values of p the polynomial kernel

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + (\mathbf{x}^T \mathbf{x}')\right)^p \tag{20}$$

corresponds to long feature vectors $\phi(\mathbf{x})$.

In two dimensions:

degree p	# of features
2	6
3	10
4	15
5	21

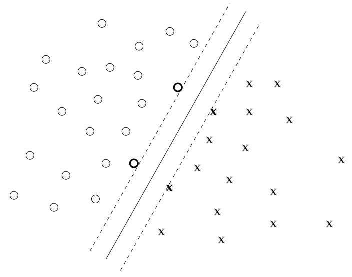
In three dimensions

degree p	# of features
2	10
3	20
4	35
5	56

(it gets much worse in higher dimensions)

- The dimensionality of the feature space does not tell the whole story

Cross-validation error



- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{CV error} \leq \frac{\#\ \text{support vectors}}{\#\ \text{of training examples}} \tag{21}$$

Worst case analysis

- How complex a classifier can we estimate on the basis of only a small number of training examples?

  We need to *define* what exactly we mean by complexity

  This is often but not always equal to the number of parameters (degrees of freedom) in the model.

- We will define *Vapnik-Chervonenkis dimension* or *VC-dimension* for a set of classifiers that we are considering

VC-dimension: preliminaries

- **A set of classifiers F:**
  For example, this could be the set of all possible linear separators, where $h \in F$ means that

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + w_0\right) \tag{22}$$

  for some values of the parameters $\mathbf{w}, w_0$.

- **Complexity:** how many different ways can we label n training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with classifiers $h \in F$?

  In other words, how many distinct binary vectors

$$[h(\mathbf{x}_1)\ h(\mathbf{x}_2)\ \ldots\ h(\mathbf{x}_n)] \tag{23}$$

  do we get by trying all $h \in F$?

$$\begin{array}{ll} [\ \text{-1} \quad 1 \quad \ldots \quad 1\ ] & h_1 \\ [\ \ 1 \quad \text{-1} \quad \ldots \quad 1\ ] & h_2 \\ & \ldots \end{array}$$

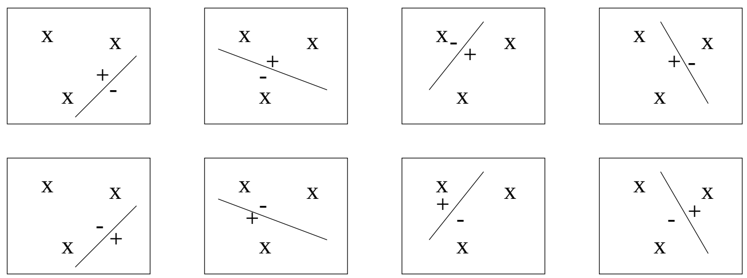
VC-dimension: shattering

- A set of classifiers F *shatters* n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ if

$$[h(\mathbf{x}_1)\ h(\mathbf{x}_2)\ \ldots\ h(\mathbf{x}_n)],\ \ h \in F \tag{24}$$

  generates all 2^n distinct labelings.

- Example: linear decision boundaries shatter (any) 3 points in 2D

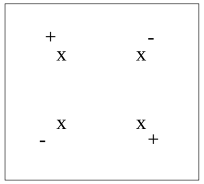


  but not any 4 points...

VC-dimension: shattering cont'd

- We cannot shatter 4 points in 2D with linear separators

  For example, the following labeling



  cannot be produced with a linear separator

- More generally: the set of all d-dimensional linear separators can shatter exactly $d+1$ points

VC-dimension

- *The VC-dimension d_{VC} of a set of classifiers F is the largest number of points that F can shatter*

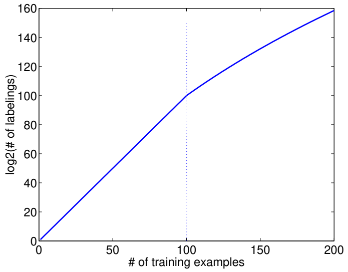
  This is a combinatorial concept and doesn't depend on what type of classifiers we use, only how "flexible" the set of classifiers is

- Example: Let F be a set of classifiers defined in terms of linear combinations of m **fixed** basis functions

$$h(\mathbf{x}) = \text{sign}\,(\, w_0 + w_1\phi_1(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x})\,) \qquad (25)$$

  d_{VC} is at most $m+1$ regardless of the form of the fixed basis functions.

Learning and VC-dimension



- The number of labelings that the set of classifiers can generate over n points increases sub-exponentially after $n > d_{VC}$ (in this case $d_{VC} = 100$)

Learning and VC-dimension

- *Finite VC-dimension is necessary and sufficient for (exponentially) fast convergence of a learning method*

  By convergence we mean:

$$\overbrace{\frac{1}{n}\sum_{i=1}^{n}\text{Loss}(y_i, h(\mathbf{x}_i))}^{\text{Empirical loss}} - \overbrace{E\{\,\text{Loss}(y, h(\mathbf{x}))\,\}}^{\text{Expected loss}} \to 0$$

  uniformly for all $h \in F$. Here $\text{Loss}(y, h(\mathbf{x})) = 1$ if $y \neq h(\mathbf{x})$ and zero otherwise (so called zero-one loss)

- This result holds for **any** underlying probability distribution from which the examples and the labels are generated