Machine learning: lecture 8

Tommi S. Jaakkola
MIT AI Lab
tommi@ai.mit.edu

Topics

- Support vector machine (part 2)
  - optimization, interpretation
  - kernel function, examples
- Text classification example
  - model specification
  - model estimation with regularization
  - feature selection

Support vector machine

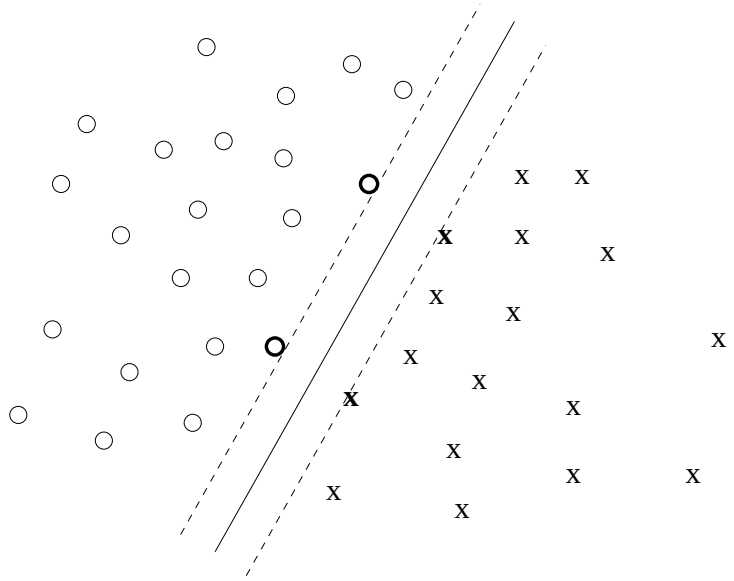
- We minimize a regularization penalty

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

  subject to the classification constraints

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

- The attained margin is now given by $1/\|\mathbf{w}\|$
- Only a few of the classification constraints are relevant
  $\Rightarrow$ support vectors



Optimization

- We have to solve the following *quadratic programming problem* to find the values for the Lagrange multipliers α_i associated with the classification constraints:

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

  where

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

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

- The resulting decision boundary has an interpretable form

$$\hat{\mathbf{w}}^T \mathbf{x} + \hat{w}_0 = \sum_{i \in SV} \hat{\alpha}_i \, y_i \, (\mathbf{x}_i^T \mathbf{x}) + \hat{w}_0$$

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 $\{\mathbf{x}_i\}_{i \in SV}$:

$$\hat{y} = \text{sign}\left(\sum_{i \in SV} \hat{\alpha}_i \, y_i \, (\mathbf{x}_i^T \mathbf{x}) + \hat{w}_0 \right)$$

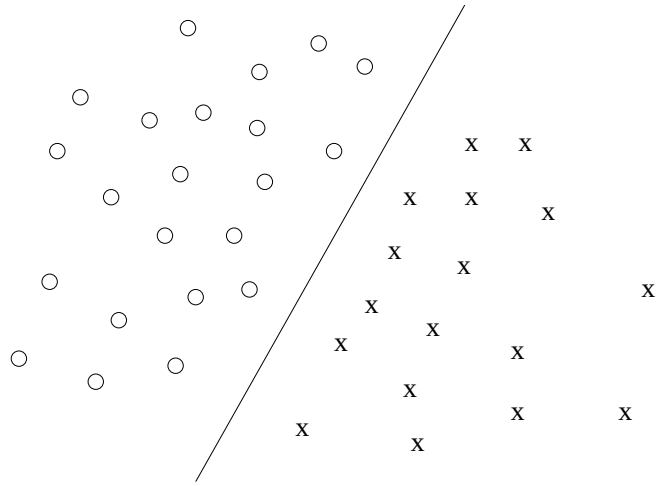
Non-linear classifier

- So far our classifier can make only linear separations
- We can easily obtain a non-linear classifier by mapping our examples $\mathbf{x} = [x_1\ x_2]$ into longer feature vectors $\Phi(\mathbf{x})$

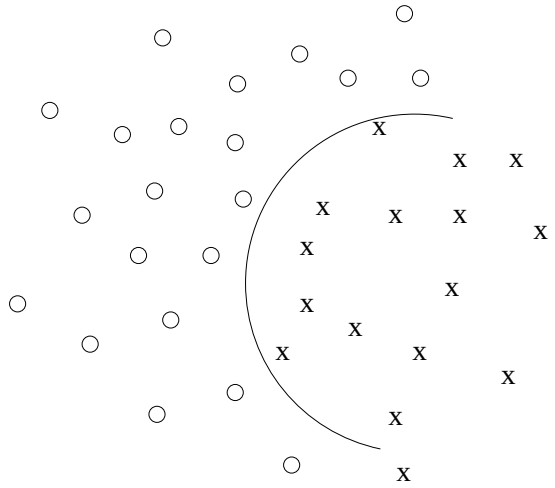
$$\Phi(\mathbf{x}) = [x_1^2\ \ x_2^2\ \ \sqrt{2}x_1x_2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2\ \ 1]$$

and applying the linear classifier to the new feature vectors $\Phi(\mathbf{x})$ instead

Non-linear classifier



Linear separator in the feature space



Non-linear separator in the original space

Feature mapping and kernels

- Let's look at the previous example in a bit more detail

$$\mathbf{x} \rightarrow \Phi(\mathbf{x}) = [x_1^2 \;\; x_2^2 \;\; \sqrt{2}x_1x_2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2 \;\; 1]$$

- The SVM classifier deals only with inner products of examples (or feature vectors). In this example,

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

  But these inner products can be evaluated without ever explicitly constructing the feature vectors $\Phi(\mathbf{x})$!

- $K(\mathbf{x}, \mathbf{x}') = \left(1 + (\mathbf{x}^T\mathbf{x}')\right)^2$ is a *kernel function* (inner product in the feature space)

Examples of kernel functions

- **Linear kernel**

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

- **Polynomial kernel**

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

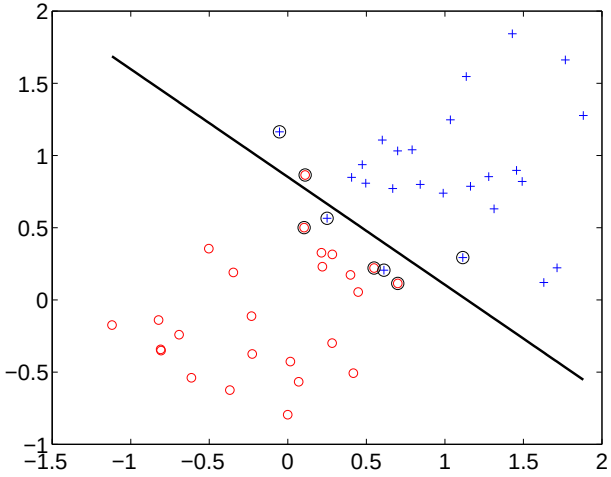
  where $p = 2, 3, \ldots$. To get the feature vectors we concatenate all p^{th} order polynomial terms of the components of $\mathbf{x}$ (weighted appropriately)

- **Radial basis kernel**

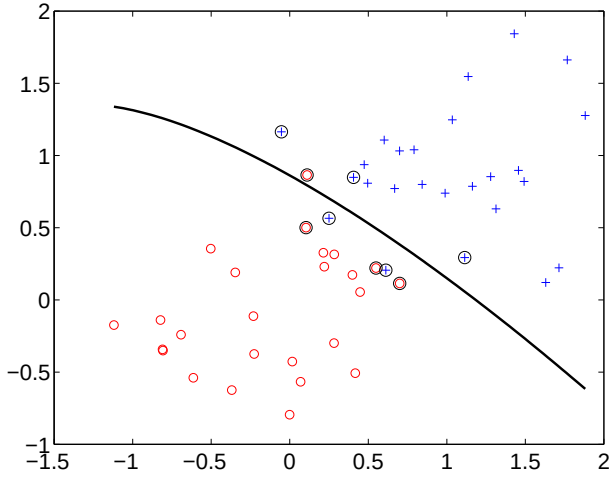
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

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

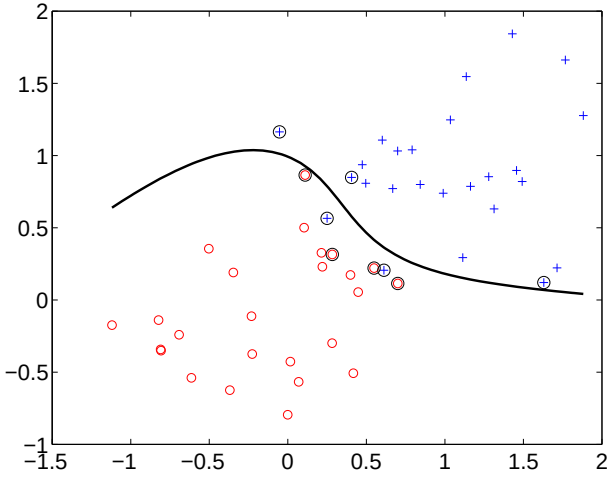
SVM examples



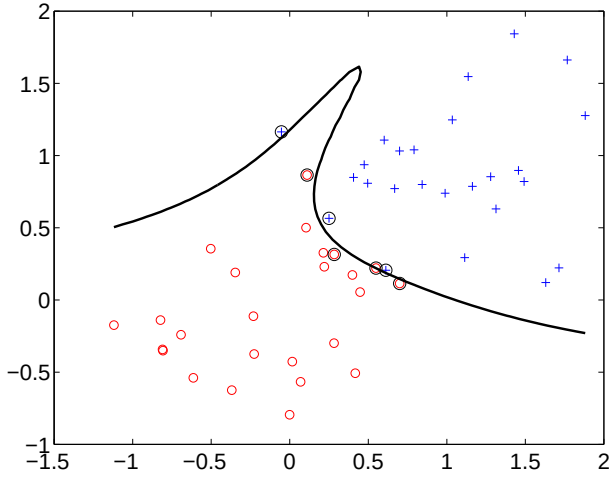
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

Dimensionality and complexity

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

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

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

In two dimensions:

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

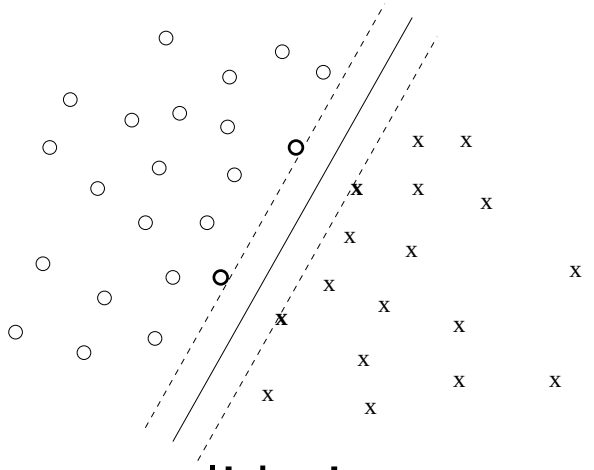
In three dimensions

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!

$$\text{Leave-one-out CV error} \leq \frac{\#\text{ support vectors}}{\#\text{ of training examples}}$$

SVM examples

- Digit recognition example (16x16 grayscale pixel images)

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

- Document classification, etc.

Topics

- Text classification example
  - model specification
  - model estimation with regularization
  - feature selection

Example problem

- Text classification (information retrieval)
  - a large number of documents $\mathbf{x}$ in a database
  - only a few labeled documents $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- We wish to build a classifier on the basis of the few labeled training examples (documents).
  - we assume for simplicity that the labels are binary (1/0)
- Several steps we need to take:
  1. feature transformation
  2. model/classifier specification
  3. model/classifier estimation with regularization

Feature transformation

- The presence/absence of specific words in a document carries information about what the document is about
- We can construct m (about $10,000$) indicator features (basis functions) $\{\phi_k(\mathbf{x})\}$ for whether a word appears in the document

  $\phi_k(\mathbf{x}) = 1$, if word k appears in document $\mathbf{x}$; zero otherwise

  $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]^T$ is the resulting *feature vector*

Classifiers

- Discriminative (support vector machine)
  - need to choose the kernel and evaluate inner products between the original feature vectors
- Generative model
  - need to define class conditional distributions

Model specification: "Naive Bayes" model

- We can treat each word detector $\phi_i(\mathbf{x})$ as an independent expert that provides some information about the classification
- We combine these "expert opinions" by modeling their decisions given the labels:

$$P(\Phi(\mathbf{x})|y,\theta) = \left[\prod_{k=1}^{m} P(\phi_k(\mathbf{x})|y,\theta_k)\right]$$

  where $P(\phi_k(\mathbf{x})|y,\theta_k)$ is the conditional probability that the k^{th} word appears in a document labeled y. θ_k are the parameters associated with this conditional probability.

- Classification via Bayes rule:

$$P(y|\Phi(\mathbf{x}),\theta) = \frac{P(\Phi(\mathbf{x})|y,\theta)P(y)}{\sum_{y'=0,1} P(\Phi(\mathbf{x})|y',\theta)P(y')}$$

Naive Bayes estimation

- We can write the conditional probabilities of a single feature as

$$P(\phi_k(\mathbf{x})|y, \theta_k) = \theta_{k|y}^{\phi_k(\mathbf{x})} (1 - \theta_{k|y})^{1-\phi_k(\mathbf{x})}$$

  where $\theta_{k|y}$ is the probability that the word k appears in a document labeled y and $\theta_k = \{\theta_{k|1}, \theta_{k|0}\}$.

Naive Bayes estimation cont'd

$$P(\phi_k(\mathbf{x})|y,\theta_k) = \theta_{k|y}^{\phi_k(\mathbf{x})}\,(1-\theta_{k|y})^{1-\phi_k(\mathbf{x})}$$

- Maximum likelihood estimation (here for a single feature)

$$\begin{aligned} J_n(\theta_k) &= \sum_{i=1}^{n} \log P(\phi_k(\mathbf{x}_i)|y_i,\theta_k) \\ &= \sum_{i=1}^{n} \left[\phi_k(\mathbf{x}_i)\log(\theta_{k|y_i}) + (1-\phi_k(\mathbf{x}_i))\log(1-\theta_{k|y_i})\right] \\ &= \sum_{y=0,1} \left[N_{ky}\log(\theta_{k|y}) + (N_y - N_{ky})\log(1-\theta_{k|y})\right] \end{aligned}$$

$N_{ky} = \#$ of documents containing word k and labeled y
$N_y = \#$ of documents with label y

Naive Bayes estimation cont'd

- We can solve for the parameters directly

$$
\begin{aligned}
J_n(\theta_k) &= \sum_{y=0,1} \left[N_{ky} \log(\theta_{k|y}) + (N_y - N_{ky}) \log(1 - \theta_{k|y}) \right] \\
\frac{\partial}{\partial \theta_{k|y}} J_n(\theta_k) &= \frac{N_{ky}}{\theta_{k|y}} - \frac{N_y - N_{ky}}{1 - \theta_{k|y}} = 0 \\
\Rightarrow \hat{\theta}_{k|y} &= \frac{N_{ky}}{N_y} \text{ (empirical fraction)}
\end{aligned}
$$

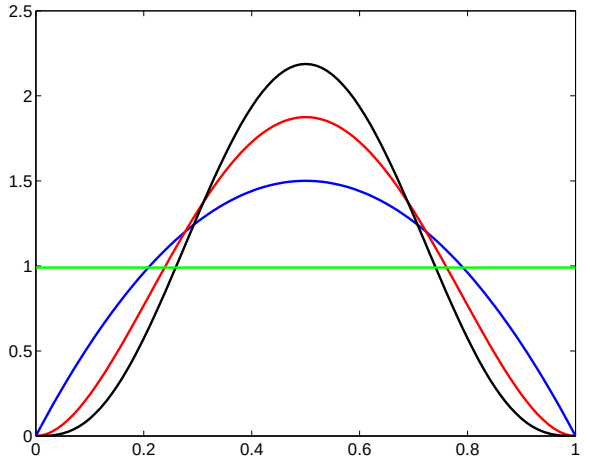
- **BUT**: we have very few documents and some words are rare; these estimates are unlikely to be good
- We need regularization...

Prior over the parameters

- Suppose we are dealing with simple coin flips (0/1), where parameter θ determines the probability of "1".
- We can construct a prior over θ on the basis of
  1. a default parameter choice p (in the absence of any data)
  2. how much we believe in the default choice (parameter n')

- Such a prior is known as the *beta distribution*:

$$P(\theta) \quad \propto \quad \theta^{n'p}\,(1-\theta)^{(n'-n'p)}$$



$p = 0.5,\ n' = 0, 1, 2, 3$

(n' and p are known as hyper-parameters)

Regularized Naive Bayes estimation

- To include the prior distribution $P(\theta_k)$ as a regularizer we maximize the penalized log-likelihood

$$\begin{aligned} J_n(\theta_k) &= \sum_{y=0,1} \left[N_{ky} \log(\theta_{k|y}) + (N_y - N_{ky}) \log(1 - \theta_{k|y}) \right] \\ &\quad + \sum_{y=0,1} \log P(\theta_{k|y}) \\ &= \sum_{y=0,1} \left[N_{ky} \log(\theta_{k|y}) + (N_y - N_{ky}) \log(1 - \theta_{k|y}) \right] \\ &\quad + \sum_{y=0,1} \left[n'p \log(\theta_{k|y}) + (n' - n'p) \log(1 - \theta_{k|y}) \right] \end{aligned}$$

- The resulting parameter estimates are

$$\hat{\theta}_{k|y} = \frac{N_{ky} + n'p}{N_y + n'}$$

Are we done ... is regularization enough?