6.867 Machine learning and neural networks

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

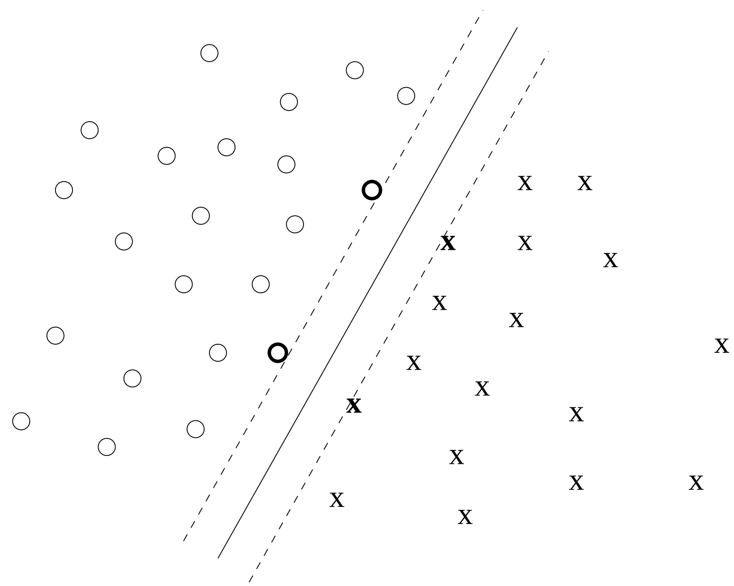
Lecture 9: support vector machine

Topics

- Support vector machines
  - "optimal" hyperplane
  - kernel function

“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*

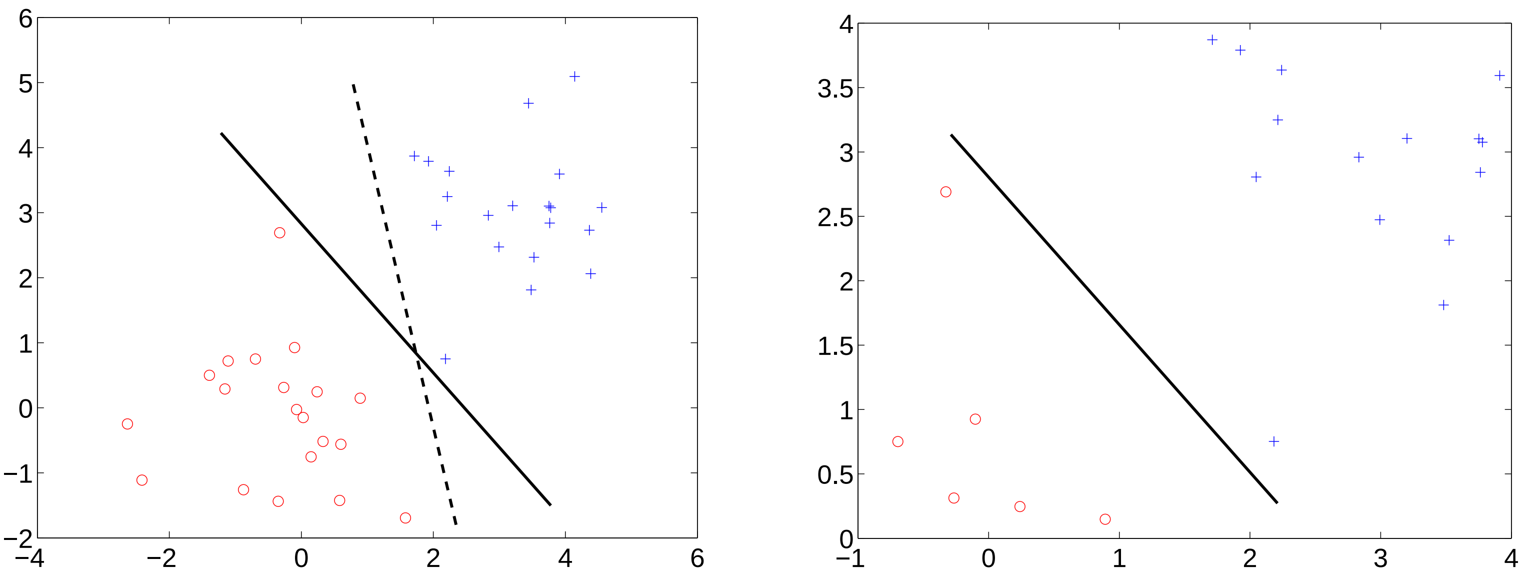
“Optimal” hyperplane cont’d

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

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

- 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

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



"Optimal" hyperplane cont'd

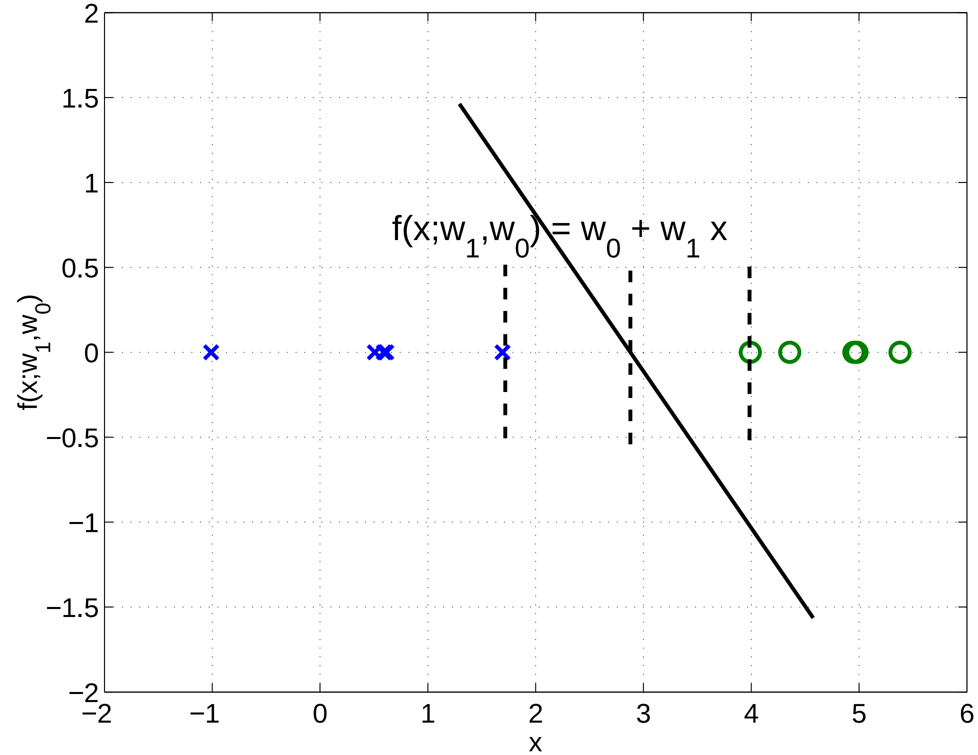
- One dimensional example: $f(x; w_1, w_0) = w_0 + w_1 x$.

  Relevant classification constraints are

  $$\begin{cases} 1\,(w_0 + w_1 x^+) - 1 \geq 0 \\ -1\,(w_0 + w_1 x^-) - 1 \leq 0 \end{cases}$$

  which lead to

  $$\begin{aligned} w_1(x^+ - x^-) - 2 &\geq 0 \\ \underbrace{|x^- - x^+|/2}_{\text{max margin}} &\geq \frac{1}{|w_1|} \end{aligned}$$



- Maximum margin separation is achieved by minimizing $|w_1|$ subject to the classification constraints

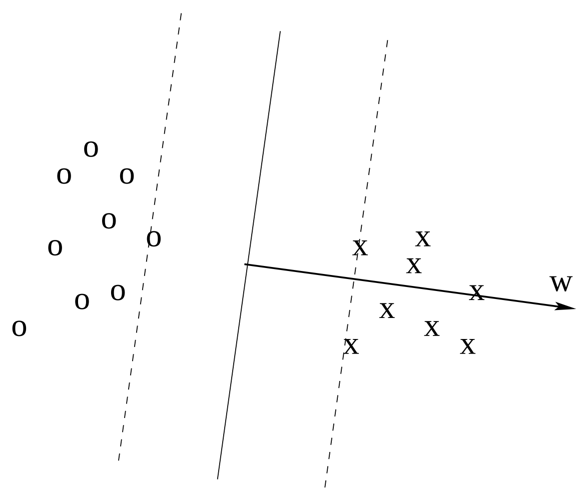
Support vector machine

- We minimize

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

subject to the classification constraints

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



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

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*

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

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

For fixed $\{\alpha_i\}$

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

Solution

- Substituting the solution $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ back into the objective leaves us with the following (dual) optimization problem over the Lagrange multipliers:

  We *maximize*

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

  subject to the constraints

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

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

- This is a *quadratic programming problem*

Support vector machines

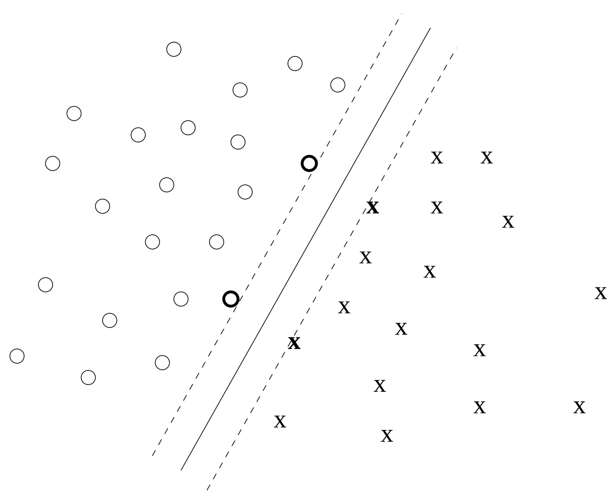
- Once we have the Lagrange multipliers $\{\hat{\alpha}_i\}$, we can reconstruct the parameter vector $\hat{\mathbf{w}}$ as a weighted combination of the training examples:

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

where the "weight" $\hat{\alpha}_i = 0$ for all but the *support vectors* (SV)

- The decision boundary has an interpretable form

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



(how did we set $\hat{w}_0$?)

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

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

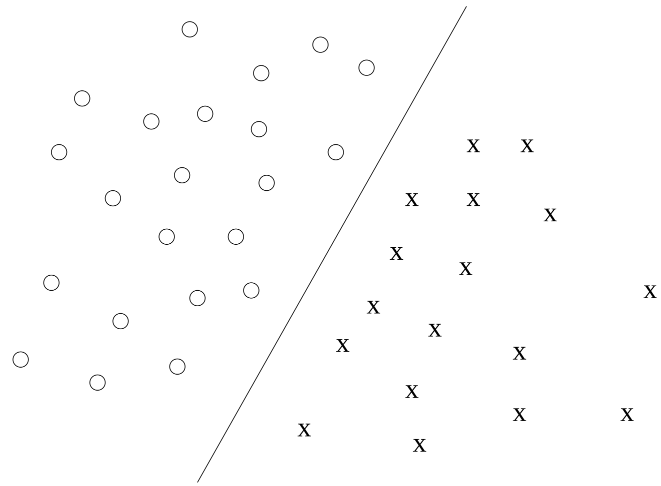
Non-linear classifier

- 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 $\Phi(\mathbf{x})$

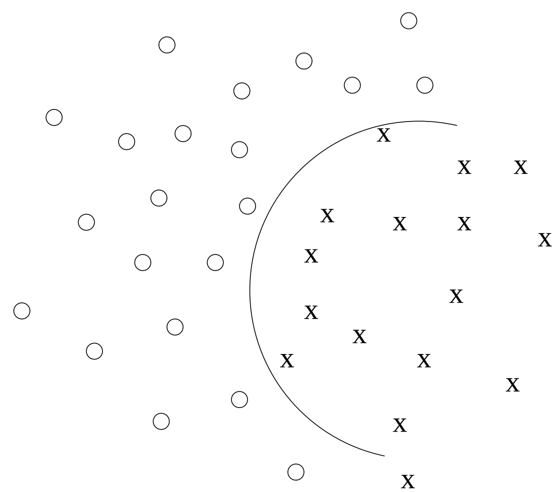
  $$\Phi(\mathbf{x}) = [x_1^2 \;\; x_2^2 \;\; \sqrt{2}x_1x_2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2 \;\; 1]$$

  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,

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

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

- $K(\mathbf{x},\mathbf{x}') = \Big(1 + (\mathbf{x}^T\mathbf{x}')\Big)^2$ 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**

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

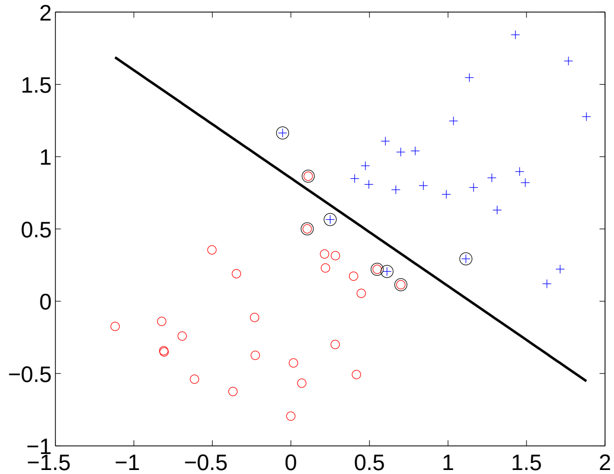
  where $p = 2, 3, \ldots$. To get the feature vectors we concatenate all p^{th} order polynomial terms of the components of $\mathbf{x}$ (weighted appropriately)

- **Radial basis kernel**

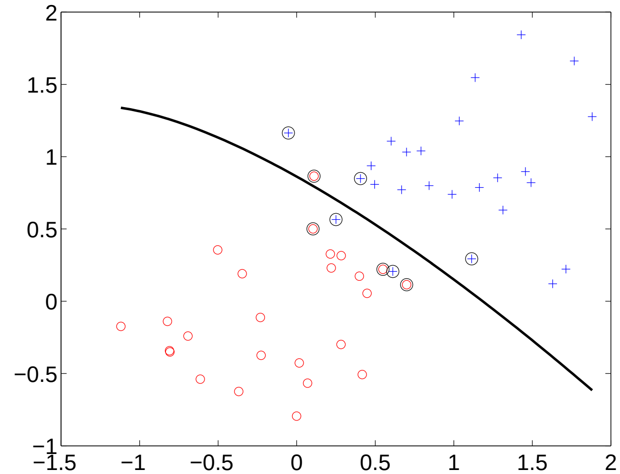
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

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

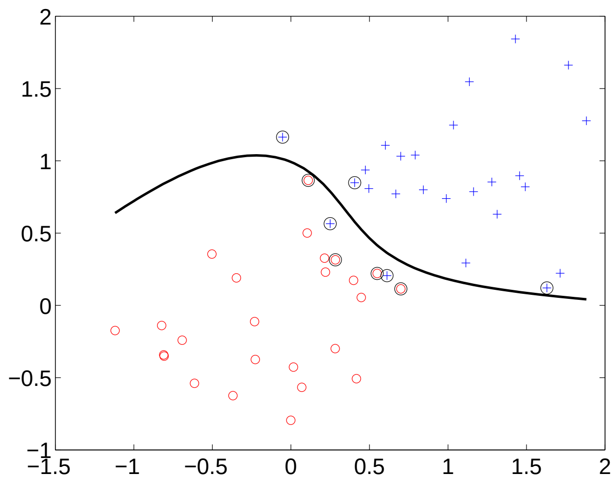
SVM examples



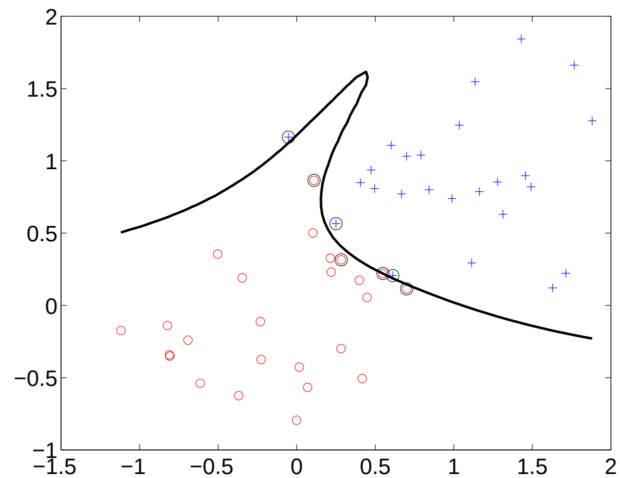
linear



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

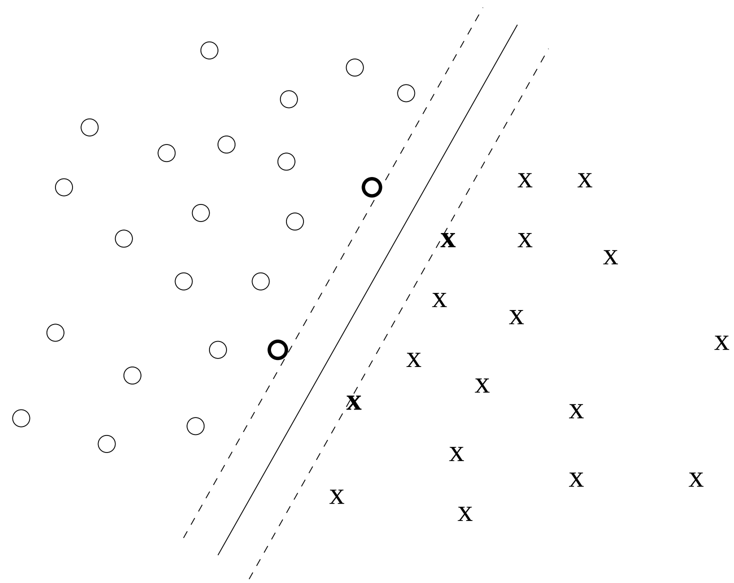
In three dimensions

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.