# Machine learning: lecture 8

Tommi S. Jaakkola
MIT CSAIL
*tommi@csail.mit.edu*

# Topics

- Support vector machines
  - training, prediction
  - other kernel methods
- Kernels
  - examples, properties, construction
  - feature vectors and sparsity

# SVM summary

- **Training:** We can find the optimal setting of the Lagrange multipliers $\alpha_i$ by maximizing

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

subject to $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$.

  - larger $C$ means larger penalty for errors
  - $\hat{\alpha}_i = 0$ except for "support vectors"
  - all misclassified examples will be support vectors
  - $\hat{w}_0$ can be found based on examples for which $\hat{\alpha}_i$ is between 0 and $C$ (when classification constraints are satisfied with equality)

# SVM summary

- **Training:** We can find the optimal setting of the Lagrange multipliers $\alpha_i$ by maximizing

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

subject to $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$.

- **Prediction:** We make predictions according to the sign of the discriminant function

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

# Other kernel methods: linear regression

- A linear regression model with feature vectors:

  $$f(\mathbf{x}; \mathbf{w}) = \phi(\mathbf{x})^T \mathbf{w}_1 + w_0,$$

  where $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]^T$.

  We can train these models via regularized least squares

  $$\min \quad \frac{1}{2} \sum_{i=1} (y_i - f(\mathbf{x}; \mathbf{w}))^2 + \frac{\lambda}{2} \|\mathbf{w}_1\|^2$$

- We'd like to turn these models into kernel methods where the examples (feature vectors) appear only in inner products $K(\mathbf{x}, \mathbf{x}') = \phi^T(\mathbf{x})\phi(\mathbf{x}')$.

# Other kernel methods: linear regression

- **Training**: maximize

$$\sum_{i=1}^{n} \left(\alpha_i y_i - \lambda \alpha_i^2/2\right) - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $\alpha_i \in \mathcal{R}$ and $\sum_i \alpha_i = 0$.

The offset parameter $\hat{w}_0$ can be obtained directly from the solution:

$$\hat{w}_0 = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \hat{\alpha}_j K(\mathbf{x}_i, \mathbf{x}_j)\right)$$

# Other kernel methods: linear regression

- **Training**: maximize

$$\sum_{i=1}^{n} \left(\alpha_i y_i - \lambda \alpha_i^2/2\right) - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

  subject to $\alpha_i \in \mathcal{R}$ and $\sum_i \alpha_i = 0$.

  The offset parameter $\hat{w}_0$ can be obtained directly from the solution:

$$\hat{w}_0 = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \hat{\alpha}_j K(\mathbf{x}_i, \mathbf{x}_j)\right)$$

- **Prediction**: the predicted output for a new point $\mathbf{x}$ is

$$f(\mathbf{x}; \hat{\alpha}, \hat{w}_0) = \hat{w}_0 + \sum_{i=1}^{n} \hat{\alpha}_i K(\mathbf{x}, \mathbf{x}_i)$$

# Other kernel methods: logistic regression

- A logistic regression model with feature vectors

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\big(\, \phi(\mathbf{x})^T \mathbf{w}_1 + w_0 \,\big)$$

where $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]^T$

As before we can train these models by minimizing the following regularized empirical loss (maximizing penalized log-likelihood):

$$\min \quad \sum_{i=1}^{n} -\log P(y_i|\mathbf{x}_i, \mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}_1\|^2$$

# Other kernel methods: logistic regression

- **Training**: maximize

$$\sum_{i=1}^{n} H(\lambda\alpha_i)/\lambda - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq 1/\lambda$ and $\sum_i \alpha_i y_i = 0$.

Here $H(p) = -p\log(p) - (1-p)\log(1-p)$ is the binary entropy function. $\hat{w}_0$ has to be solved iteratively after obtaining $\hat{\alpha}$.

# Other kernel methods: logistic regression

- **Training**: maximize

$$\sum_{i=1}^{n} H(\lambda\alpha_i)/\lambda - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

  subject to $0 \leq \alpha_i \leq 1/\lambda$ and $\sum_i \alpha_i y_i = 0$.

  Here $H(p) = -p\log(p) - (1-p)\log(1-p)$ is the binary entropy function. $\hat{w}_0$ has to be solved iteratively after obtaining $\hat{\alpha}$.

- **Prediction:** the predicted probabilities over possible labels for a new point $\mathbf{x}$ are given by

$$P(y = 1|\mathbf{x}, \hat{\alpha}, \hat{w}_0) = g\big(\,\hat{w}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i)\,\big)$$

# Example kernels

- **Linear kernel**

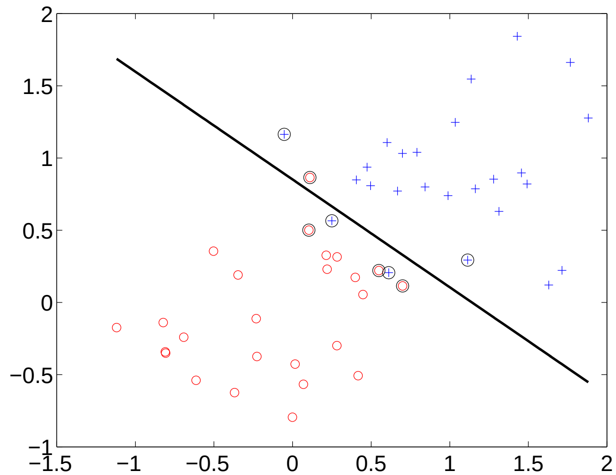
$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')$$

- **Polynomial kernel**

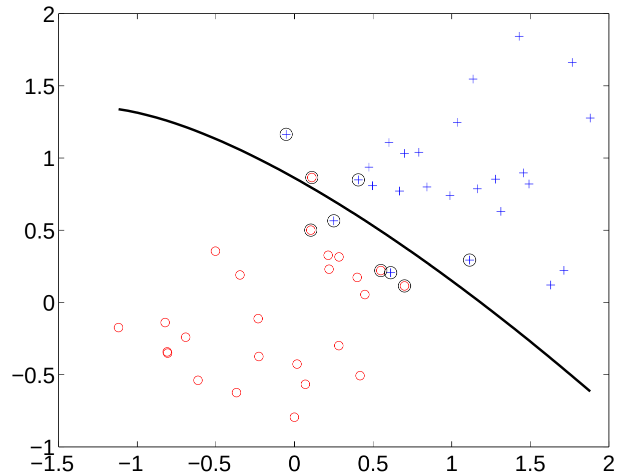
$$K(\mathbf{x}, \mathbf{x}') = \left(1 + (\mathbf{x}^T \mathbf{x}')\right)^p$$

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

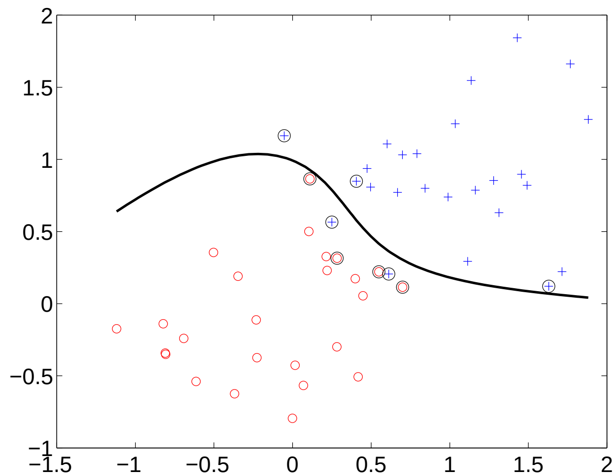
# Polynomial kernels with SVMs



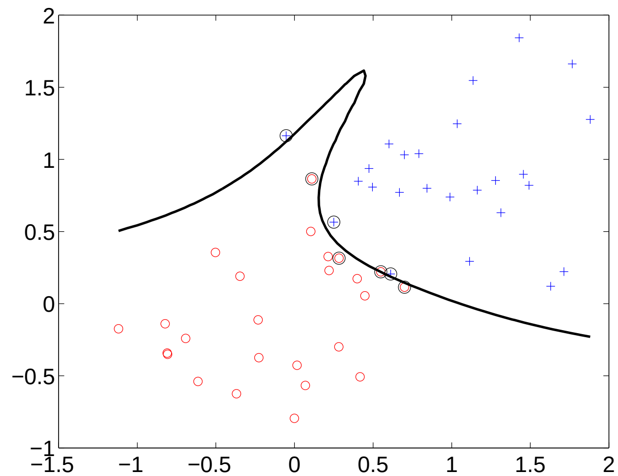

linear




$2^{nd}$ order polynomial




$4^{th}$ order polynomial



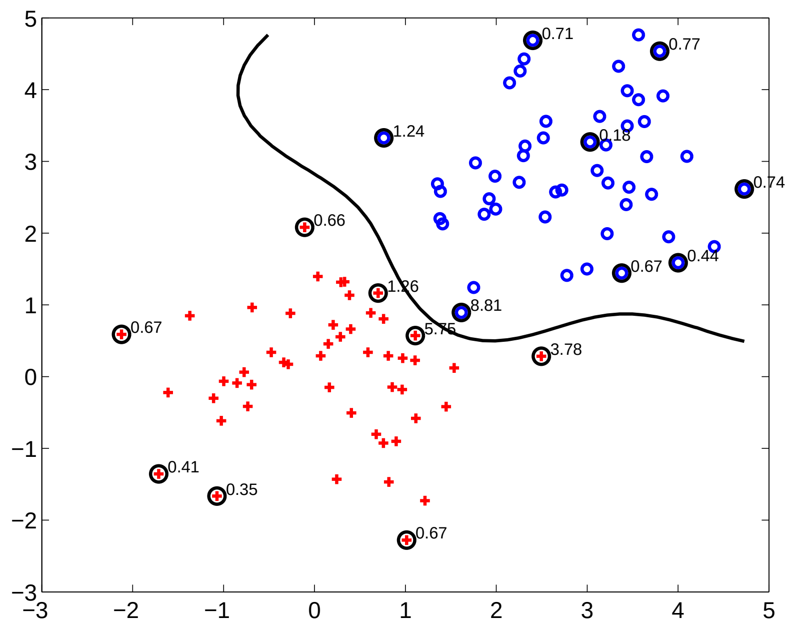

$8^{th}$ order polynomial

# Example kernels

- **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 is infinite dimensional function space (use of the kernel results in a *non-parametric* classifier).




– support vectors need not appear close to the boundary in the input space, only in the feature space

# Definition of kernels

- We can think of kernels in terms of explicit or implicit feature mappings
  - Definition 1: $K(\mathbf{x}, \mathbf{x}')$ is a kernel if it can be written as an inner product $\phi(\mathbf{x})^T \phi(\mathbf{x}')$ for some feature mapping $\phi$.
  - Definition 2: $K(\mathbf{x}, \mathbf{x}')$ is a kernel if for any finite set of training examples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the $n \times n$ matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is positive semi-definite.

# Kernels and construction

- We can build kernels from simpler ones. For example:
  - If $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$ are valid kernels then

$$
\begin{aligned}
f(\mathbf{x})K_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') & \qquad \text{(scaling)} \\
K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}') & \qquad \text{(sum)} \\
K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}') & \qquad \text{(product)}
\end{aligned}
$$

    are valid kernels.
  - If $\mathbf{x} = [x_1, \ldots, x_d]^T \in \mathcal{R}^d$ and $K_i(x_i, x_i')$ are valid 1-dimensional kernels, then

$$K(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{d} K_i(x_i, x_i')$$

    is a valid kernel in $\mathcal{R}^d$.

# Kernels and sequences

- We can also derive kernels for variable length sequences. For example:

  $\mathbf{x} = \ldots$ `my first day this term was ...`
  $\mathbf{x}' = \ldots$ `Last year the midterm had ...`

  Gap-weighted subsequence kernel:

  $$K(\mathbf{x}, \mathbf{x}') = \sum_{u \in \Sigma^d} \sum_{\vec{i}: u = \mathbf{x}[\vec{i}]} \sum_{\vec{j}: u = \mathbf{x}[\vec{j}]} \lambda^{(i_d - i_1)} \lambda^{(j_d - j_1)}$$

  where $\lambda \in (0, 1)$ and $\Sigma^d$ is the set of all sequences of length $d$. The kernel reflects the degree to which the sequences have common subsequences penalizing non-contiguous subsequences.

# Dimensionality and complexity

- Many of these kernels correspond to very high dimensional feature spaces
  - polynomial kernel for large $p$ or $dim(\mathbf{x})$
  - radial basis kernel (infinite)
  - subsequence kernel (combinatorial)

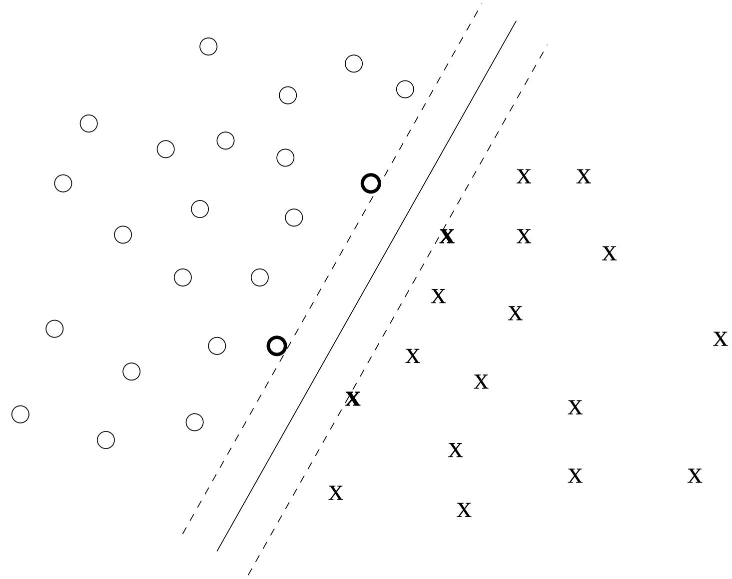
  etc.

- The dimensionality of the feature space determines the number of parameters in the primal formulation

$$\min \; \|\mathbf{w}_1\|^2 \text{ subject to } y_i[w_0 + \phi(\mathbf{x}_i)^T \mathbf{w}_1] - 1 \geq 0, \;\; \forall i$$

Can these methods generalize?

# Cross-validation



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

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

(similar results exist for kernel logistic regression)

# Kernels, examples, sparsity

- High dimensional feature vectors (many basis functions) can still permit a sparse solution in terms of the number of training examples

$$
\underbrace{
\overset{\phi(\mathbf{x}_1)}{\begin{bmatrix} \phi_1(\mathbf{x}_1) \\ \phi_2(\mathbf{x}_1) \\ \phi_3(\mathbf{x}_1) \\ \cdots \\ \phi_d(\mathbf{x}_1) \end{bmatrix}}
\quad
\overset{\phi(\mathbf{x}_2)}{\begin{bmatrix} \phi_1(\mathbf{x}_2) \\ \phi_2(\mathbf{x}_2) \\ \phi_3(\mathbf{x}_2) \\ \cdots \\ \phi_d(\mathbf{x}_2) \end{bmatrix}}
}_{\text{a few examples)}}
\quad \cdots \quad
\overset{\phi(\mathbf{x}_n)}{\begin{bmatrix} \phi_1(\mathbf{x}_n) \\ \phi_2(\mathbf{x}_n) \\ \phi_3(\mathbf{x}_n) \\ \cdots \\ \phi_d(\mathbf{x}_n) \end{bmatrix}}
\left.\begin{matrix} \\ \\ \end{matrix}\right\} \begin{matrix}\text{a few} \\ \text{components}\end{matrix}
$$

- Alternatively, we could try to find a few basis functions (components) that solve the classification/regression task