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

$$
\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)
$$

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

\[
\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)
\]

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

\[
\hat{y} = \text{sign}(\hat{w}_0 + \sum_{i \in SV} \hat{\alpha}_i y_i K(x, x_i))
\]
Other kernel methods: linear regression

- A linear regression model with feature vectors:

\[ f(x; w) = \phi(x)^T w_1 + w_0, \]

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

We can train these models via regularized least squares

\[
\min \quad \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x; w))^2 + \frac{\lambda}{2} \|w_1\|^2
\]

- We’d like to turn these models into kernel methods where the examples (feature vectors) appear only in inner products

\[ K(x, x') = \phi^T(x)\phi(x'). \]
Other kernel methods: linear regression

- **Training**: maximize

\[
\sum_{i=1}^{n} \left( \alpha_i y_i - \lambda \frac{\alpha_i^2}{2} \right) - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j)
\]

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:

\[
\hat{w}_0 = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j) \right)
\]
Other kernel methods: linear regression

- **Training**: maximize

\[
\sum_{i=1}^{n} (\alpha_i y_i - \lambda \alpha_i^2 / 2) - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j)
\]

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:

\[
\hat{w}_0 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j))
\]

- **Prediction**: the predicted output for a new point \(x\) is

\[
f(x; \hat{\alpha}, \hat{w}_0) = \hat{w}_0 + \sum_{i=1}^{n} \hat{\alpha}_i K(x, x_i)
\]
Other kernel methods: logistic regression

- A logistic regression model with feature vectors

\[ P(y = 1|x, w) = g(\phi(x)^T w_1 + w_0) \]

where \( \phi(x) = [\phi_1(x), \ldots, \phi_m(x)]^T \)

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

\[
\min \sum_{i=1}^{n} -\log P(y_i|x_i, w) + \frac{\lambda}{2} \| w_1 \|^2
\]
Other kernel methods: logistic regression

- **Training**: maximize

\[
\sum_{i=1}^{n} H(\lambda \alpha_i)/\lambda - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)
\]

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

\[
\sum_{i=1}^{n} H(\lambda \alpha_i)/\lambda - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)
\]

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 \(x\) are given by

\[
P(y = 1|x, \hat{\alpha}, \hat{w}_0) = g\left( \hat{w}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i K(x, x_i) \right)
\]
Example kernels

- **Linear kernel**

  \[ K(x, x') = (x^T x') \]

- **Polynomial kernel**

  \[ K(x, x') = \left(1 + (x^T x')\right)^p \]

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

- Linear
- 2nd order polynomial
- 4th order polynomial
- 8th order polynomial

Tommi Jaakkola, MIT CSAIL
Example kernels

- **Radial basis kernel**

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

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: \( K(x, x') \) is a kernel if it can be written as an inner product \( \phi(x)^T \phi(x') \) for some feature mapping \( \phi \).
  - Definition 2: \( K(x, x') \) is a kernel if for any finite set of training examples, \( x_1, \ldots, x_n \), the \( n \times n \) matrix \( K_{ij} = K(x_i, x_j) \) is positive semi-definite.
Kernels and construction

- We can build kernels from simpler ones. For example:
  - If $K_1(x, x')$ and $K_2(x, x')$ are valid kernels then
    \begin{align*}
    f(x)K_1(x, x')f(x') & \quad \text{(scaling)} \\
    K_1(x, x') + K_2(x, x') & \quad \text{(sum)} \\
    K_1(x, x')K_2(x, x') & \quad \text{(product)}
    \end{align*}
  are valid kernels.

- If $x = [x_1, \ldots, x_d]^T \in \mathcal{R}^d$ and $K_i(x_i, x'_i)$ are valid 1-dimensional kernels, then
  \begin{equation}
  K(x, x') = \prod_{i=1}^d K_i(x_i, x'_i)
  \end{equation}
  is a valid kernel in $\mathcal{R}^d$. 
Kernels and sequences

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

  \[ x = \ldots \text{my first day this term was} \ldots \]

  \[ x' = \ldots \text{Last year the midterm had} \ldots \]

  Gap-weighted subsequence kernel:

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

  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 $dim(x)$
  - radial basis kernel (infinite)
  - subsequence kernel (combinatorial)
    etc.

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

$$
\min \|w_1\|^2 \text{ subject to } y_i[w_0 + \phi(x_i)^T w_1] - 1 \geq 0, \quad \forall i
$$

Can these methods generalize?
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

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

\[
\begin{bmatrix}
\phi(x_1) \\
\phi_1(x_1) \\
\phi_2(x_1) \\
\phi_3(x_1) \\
\vdots \\
\phi_d(x_1)
\end{bmatrix}
\begin{bmatrix}
\phi(x_2) \\
\phi_1(x_2) \\
\phi_2(x_2) \\
\phi_3(x_2) \\
\vdots \\
\phi_d(x_2)
\end{bmatrix}
\ldots
\begin{bmatrix}
\phi(x_n) \\
\phi_1(x_n) \\
\phi_2(x_n) \\
\phi_3(x_n) \\
\vdots \\
\phi_d(x_n)
\end{bmatrix}
\}
\text{a few components}
\]

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