Support vector machines

COMS 4771

1. Recap: linearly separable data

Linear classifiers (with $Y = \{-1, +1\}$)

**Linear separability assumption**
Assume there is a linear classifier that perfectly classifies the training data $S$: for some $w_* \in \mathbb{R}^d$,
\[
\min_{(x, y) \in S} yx^T w_* > 0.
\]

**Linear programming**
Can find some linear separator in polynomial time by solving linear feasibility problem (a linear program).

**Perceptron**
Also finds some linear separator quickly if there is one with a large margin.

Support vector machines (SVMs)

**Motivation**
- Ambiguity and potential instability in what LP and Perceptron returns.
- What to do when $S$ is not linearly separable?
  (Some options: logistic regression and Online Perceptron.)

**Support vector machines (Vapnik and Chervonenkis, 1963)**
- Characterize a stable solution for linearly separable problems—the *maximum margin solution*.
- Solution specified via a convex optimization problem that can be solved via efficient algorithms.
- Convex dual of the optimization problem reveals useful structure.
- Minor alteration to optimization problem gives natural way to handle non-separable cases via convex surrogate losses.
2. Maximum margin solution

**Maximum margin solution**

**Best linear classifier on population**

**Possible Perceptron or LP solution on training data**

**"Maximum margin" solution on training data**

Why use the “maximum margin” solution?
(i) Uniquely determined by \( S \), unlike LP’s/Perceptron’s.
(ii) It is a particular “inductive bias”—i.e., an assumption about the problem—that seems to be commonly useful.

**Key insight:** can write as solution to a mathematical optimization problem.

### Distance to decision boundary

Suppose \( w \in \mathbb{R}^d \) satisfies

\[
\min_{(x,y) \in S} yx^T w > 0.
\]

Then so does, e.g., \( w/100 \). Let’s fix a particular scaling of \( w \).

Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

- Rescale \( w \) so that \( \tilde{y} \tilde{x}^T w = 1 \).
  
  (Now scaling is fixed.)
  
- Now distance from \( \tilde{y} \tilde{x} \) to \( H \) is \( \frac{1}{\|w\|_2} \).
  
  This distance is called the **margin**.

Therefore, the shortest vector \( w \) satisfying

\[
\min_{(x,y) \in S} yx^T w = 1,
\]

corresponds to the linear separator with largest margin on examples in \( S \).

### Maximum margin linear classifier

The solution \( \hat{w} \) to the following mathematical optimization problem:

\[
\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|_2^2 \\
\text{s.t.} \quad yx^T w \geq 1 \text{ for all } (x, y) \in S
\]

gives the linear classifier with the largest minimum margin on \( S \)—i.e., the **maximum margin linear classifier** or **support vector machine (SVM) classifier**.

Problem is a **convex optimization problem**; can be solved in polynomial time.

If there is a solution (i.e., \( S \) is linearly separable), then the solution is unique. (Compare to LP’s and Perceptron’s lack of determinism from \( S \).)

**Note:** Can also explicitly include affine expansion, so decision boundary need not pass through origin. In this class, we’ll skip the affine expansion unless stated explicitly otherwise.
Two SVM optimization problems

3. SVM dual problem

\[ \text{SVM (primal) problem} \]
\[
\begin{align*}
\min_{w \in \mathbb{R}^d} & \quad \frac{1}{2} \|w\|_2^2 \\
\text{s.t.} & \quad y_i x_i^T w \geq 1 \quad \text{for all } i = 1, \ldots, n.
\end{align*}
\]

\[ \text{SVM dual problem} \]
\[
\max_{\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j.
\]

We’ll derive SVM dual problem using Lagrangian duality.

Lagrange multipliers

Move constraints to objective using method of Lagrange multipliers.

Original problem: \(\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|_2^2\)
\[\text{s.t.} \quad 1 - y_i x_i^T w \leq 0 \quad \text{for all } i = 1, \ldots, n.\]

- For each (inequality) constraint \(1 - y_i x_i^T w \leq 0\), associate a (non-negative) dual variable \(\alpha_i\) (a.k.a. Lagrange multiplier).

- Move constraints to objective by adding \(\sum_{i=1}^{n} \alpha_i (1 - y_i x_i^T w)\) and maximizing over \(\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n\) s.t. \(\alpha \geq 0\) (i.e., \(\alpha_i \geq 0\) for all \(i\)).

Resulting optimization problem (note lack of explicit constraints on \(w\)):
\[
\min_{w \in \mathbb{R}^d} \left[ \max_{\alpha \geq 0} \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i x_i^T w) \right].
\]

Equivalence: If \(w\) violates original \(i\)-th constraint (so \(1 - y_i x_i^T w > 0\)), then the “\(\max\)” will set \(\alpha_i \to \infty\) to make objective \(\to \infty\). Such \(w\) cannot be minimizer!

Weak duality: We’ll see (next slide) that \(\min \max_{\alpha \geq 0} L(w, \alpha) \geq \max \min_{\alpha \geq 0} L(w, \alpha)\) where \(L\) (the Lagrangian) is defined by
\[
L(w, \alpha) := \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i x_i^T w).
\]

Observation: \(\min_{w \in \mathbb{R}^d} L(w, \alpha)\) can be solved in closed form, since for any \(\alpha\), the function \(w \mapsto L(w, \alpha)\) is a convex quadratic function.

Solution: Quadratic achieves minimum at \(w = \sum_{i=1}^{n} \alpha_i y_i x_i\), so
\[
\min_{w \in \mathbb{R}^d} L(w, \alpha) = L \left( \sum_{i=1}^{n} \alpha_i y_i x_i, \alpha \right) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i y_i x_i \right\|_2^2.
\]

This is the same as the objective from the SVM dual problem:
\[
\max_{\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j.
\]
Weak and strong duality

Let \( P(w) = \max_{\alpha' \geq 0} L(w, \alpha') \) be the primal objective value of \( w \).
Let \( D(\alpha) = \min_{w' \in \mathbb{R}^d} L(w', \alpha) \) be the dual objective value of \( \alpha \).

- **Weak duality:**
  \[ P(w) \geq D(\alpha). \]

  *Proof.*
  \( P(w) = \max_{\alpha' \geq 0} L(w, \alpha') \), so \( P(w) \geq L(w, \alpha) \).
  \( D(\alpha) = \min_{w' \in \mathbb{R}^d} L(w', \alpha) \), so \( D(\alpha) \leq L(w, \alpha) \).

- **Strong duality:** For the optimal (feasible) solutions \( \hat{w} \) and \( \hat{\alpha} \), we have
  \[ P(\hat{w}) = D(\hat{\alpha}). \]

  Strong duality holds (primarily) because SVM optimization problem is a **convex optimization problem**. (See upcoming lecture.)

Support vectors

- **Fact:** Optimal solutions \( \hat{w} \) and \( \hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n) \) satisfy
  \[ \hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i = \sum_{i: \hat{\alpha}_i > 0} \hat{\alpha}_i y_i x_i, \]

  \[ \hat{\alpha}_i > 0 \Rightarrow y_i x_i^\top \hat{w} = 1 \text{ for all } i = 1, \ldots, n \]
  (complementary slackness).

  The \( y_i x_i \) where \( \hat{\alpha}_i > 0 \) are called **support vectors**.

  - Support vector examples satisfy "margin" constraints with equality.
  - Get same solution even if we only had support vector examples.

Proof of complementary slackness

For the optimal (feasible) solutions \( \hat{w} \) and \( \hat{\alpha} \), we have
\[
P(\hat{w}) = D(\hat{\alpha}) = \min_{w' \in \mathbb{R}^d} L(w', \hat{\alpha}) \quad \text{(by strong duality)}
\]
\[
\leq L(\hat{w}, \hat{\alpha})
\]
\[
= \frac{1}{2} \|\hat{w}\|^2 + \sum_{i=1}^{n} \hat{\alpha}_i (1 - y_i x_i^\top \hat{w})
\]
\[
\leq \frac{1}{2} \|\hat{w}\|^2 = P(\hat{w}).
\]

Therefore, every term in sum \( \sum_{i=1}^{n} \hat{\alpha}_i (1 - y_i x_i^\top \hat{w}) \) must be zero:
\[ \hat{\alpha}_i (1 - y_i x_i^\top \hat{w}) = 0 \quad \text{for all } i = 1, \ldots, n. \]

If \( \alpha_i > 0 \), then must have \( 1 - y_i x_i^\top \hat{w} = 0 \).

4. Kernels
Looking at the dual again

SVM dual problem only depends on $x_i$ through inner products $x_i^T x_j$.

$$\max_{\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j.$$  

If we use feature expansion (e.g., quadratic expansion) $x \mapsto \phi(x)$, this becomes

$$\max_{\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j).$$

Solution $\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \phi(x_i)$ is used in the following way:

$$x \mapsto \phi(x)^T \hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \phi(x)^T \phi(x_i).$$

Key insight:
Sometimes computing $\phi(x)^T \phi(x')$ is much easier than computing $\phi(x)$.

Quadratic expansion

- $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^{1+2d+\binom{d}{2}}$, where
  \[ \phi(x) = \left( 1, \sqrt{2}x_1, \ldots, \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_1x_d, \ldots, \sqrt{2}x_{d-1}x_d \right) \]
  (Don’t mind the $\sqrt{2}$’s.)
- Computing $\phi(x)^T \phi(x')$ in $O(d)$ time:
  \[ \phi(x)^T \phi(x') = (1 + x^T x')^2. \]
- Much better than $d^2$ time.

Products of all feature subsets

- $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^{2^d}$, where
  \[ \phi(x) = \left( \prod_{i \in S} x_i \right)_{S \subseteq \{1,2,\ldots,d\}} \]
- Computing $\phi(x)^T \phi(x')$ in $O(d)$ time:
  \[ \phi(x)^T \phi(x') = \prod_{i=1}^{d} (1 + x_i x'_i). \]
- Much better than $2^d$ time.

Infinite dimensional feature expansion

For any $\sigma > 0$, there is an infinite feature expansion $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^\infty$ such that

\[ \phi(x)^T \phi(x') = \exp \left( -\frac{\|x - x'\|^2}{2\sigma^2} \right), \]

which can be computed in $O(d)$ time.

(This is called the Gaussian kernel with bandwidth $\sigma$.)

12 / 32

13 / 32

14 / 32

15 / 32
### Gaussian kernel feature expansion

For simplicity, take $d = 1$, so $\phi: \mathbb{R} \to \mathbb{R}^\infty$.

What $\phi$ has $\phi(x)\phi(y) = e^{-(x-y)^2/(2\sigma^2)}$?

Reverse engineer using Taylor expansion:

$$e^{-(x-y)^2/(2\sigma^2)} = e^{-x^2/(2\sigma^2)} \cdot e^{-y^2/(2\sigma^2)} \cdot \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{xy}{\sigma^2}\right)^k$$

So let

$$\phi(x) := e^{-x^2/(2\sigma^2)} \left(1, \frac{x}{\sigma}, \frac{x^2}{\sigma^2}, \frac{x^3}{3!}, \ldots\right).$$

How to handle $d > 1$?

### Kernels

A (positive definite) kernel function $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function satisfying:

For any $x_1, x_2, \ldots, x_n \in \mathcal{X}$, the $n \times n$ matrix whose $(i,j)$-th entry is $K(x_i, x_j)$ is positive semidefinite.

For any kernel $K$, there is a feature mapping $\phi: \mathcal{X} \to \mathbb{H}$ such that $\phi(x)^T \phi(x') = K(x, x')$.

$H$ is a Hilbert space—i.e., a special kind of inner product space—called the Reproducing Kernel Hilbert Space corresponding to $K$.

### Kernel SVMs (Boser, Guyon, and Vapnik, 1992)

Solve

$$\max_{\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j).$$

Solution $\hat{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)$ is used in the following way:

$$x \mapsto \phi(x)^T \hat{w} = \sum_{i=1}^{n} \alpha_i y_i K(x, x_i).$$

- To represent classifier, need to keep support vector examples $(x_i, y_i)$ and corresponding $\alpha_i$'s.
- To compute prediction on $x$, iterate through support vector examples and compute $K(x, x_i)$ for each support vector $x_i$ . . .

Very similar to nearest neighbor classifier: predictor is represented using (a subset of) the training data.

### Making the most of the kernel trick

1. Start with a feature expansion $\phi$ that makes sense for your problem, and find an efficient way to compute $\phi(x)^T \phi(x')$.
2. Start with a similarity function $K$ that makes sense for your problem (and is efficient-to-compute), and verify that it is a (positive definite) kernel.
3. Build new kernels out of existing kernels.
Example: String kernels

- Suppose we want to define $K : \text{Strings} \times \text{Strings} \to \mathbb{R}$ such that $K(x, x') = \# \text{ substrings } x \text{ and } x' \text{ have in common}$.
- Define $\phi : \text{Strings} \to \{0, 1\}^{\text{Strings}}$, where $\phi(x) = (1 \{s \text{ appears as substring in } x\} : s \in \text{Strings})$.
  Then $K(x, x') = \phi(x)^T \phi(x')$.
- Computing $K(x, x')$:
  For each substring $s$ of $x$, check if $s$ appears in $x'$ and update total.
  Efficient algorithm: $O(|\text{Alphabet}| \times \text{length}(x) \times \text{length}(x'))$ time.

New kernels from old kernels

Suppose $K_1$ and $K_2$ are positive definite kernel functions.

1. Does $K(x, y) := K_1(x, y) + K_2(x, y)$ define a positive definite kernel?
2. Does $K(x, y) := c \cdot K_1(x, y)$ (for $c \geq 0$) define a positive definite kernel?
3. Does $K(x, y) := K_1(x, y) \cdot K_2(x, y)$ define a positive definite kernel?

Kernelizing ridge regression

Ridge regression solution:
$$\hat{w} = (A^T A + \lambda I)^{-1} A^T b.$$  

Linear algebraic fact: $(A^T A + \lambda I)^{-1} A^T = A^T (A A^T + \lambda I)^{-1}$ for any $\lambda > 0$.

Therefore
$$\hat{w} = A^T (A A^T + \lambda I)^{-1} b = A^T \left( \frac{1}{n} K + \lambda I \right)^{-1} b =: \hat{\alpha}$$

where $K \in \mathbb{R}^{n \times n}$ is the matrix with $K_{ij} = x_i^T x_j$.

Moreover,
$$A^T \hat{\alpha} = \sum_{i=1}^{n} \hat{\alpha}_i x_i$$

so, for any $x \in \mathbb{R}^d$, 
$$x^T \hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i x^T x_i.$$  

Feature vectors only involved in inner products!
Kernelizing Perceptron

Can also kernelize specific algorithms, e.g., Perceptron.

**Perceptron**

**Input** Labeled examples $S$ from $\mathbb{R}^d \times \{-1, +1\}$.

1. **Initialize** $\hat{w}_1 := 0$.
2. for $t = 1, 2, \ldots$, do
3. if there exists $(x, y) \in S$ such that $y x^T \hat{w}_t \leq 0$ then
4. Let $(x_t, y_t)$ be any such example $(x, y)$.
5. Update: $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
6. else
7. return $\hat{w}_t$.
8. end if
9. end for

**Observation**:

$\hat{w}_t = \sum_{(x, y) \in S_t} y x$ for some $S_t \subseteq S$ of cardinality $|S_t| = t - 1$.

Therefore: Small iteration bound $\implies \hat{w}_t$ has "small" representation.

Example: OCR digits

- Binary classification problem: distinguish "9" from other digits.
- # training examples: 60000 (about 6000 are from class "9").
- Test error rates using (variant of) Kernelized Perceptron.

<table>
<thead>
<tr>
<th>Type</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>0.037</td>
</tr>
<tr>
<td>degree 2 poly</td>
<td>0.009</td>
</tr>
<tr>
<td>degree 4 poly</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Kernel approximation

**Major downside of kernel methods**: kernel matrix $K$ is of size $n \times n$, which can be computationally prohibitive to construct/store in memory when $n$ is large.

**Some alternatives**:

- **Nyström approximation**
  Find a low-rank approximation of kernel matrix:
  \[
  K \approx BB^T
  \]
  for $B \in \mathbb{R}^{n \times r}$, where $r \ll n$.
  Can somehow do this in less time than is required to form $K$ itself, and also extend to new (e.g., test data) points.

- **(Randomized) Fourier-based approximation**
  E.g., for Gaussian kernel
  \[
  K_{\sigma^2}(x, y) = \exp \left( -\frac{\|x - y\|^2}{2\sigma^2} \right).
  \]
  Leverage Fourier transform of $K_{\sigma^2}$ to construct a feature expansion $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^p$ such that $\phi(x)^T \phi(y) \approx K_{\sigma^2}(x, y)$.

Fourier transform

**Characteristic function for Gaussian random vector** for any $\delta \in \mathbb{R}^d$,

\[
\exp \left( -\frac{\|\delta\|^2}{2\sigma^2} \right) = \int_{\mathbb{R}^d} \exp(i\delta^T t) \cdot \frac{1}{(2\pi/\sigma^2)^{d/2}} \exp \left( -\frac{\|t\|^2}{2} \right) dt 
\]

$\mathcal{N}(0, (1/\sigma^2) I)$ density

Therefore, if $\theta \sim \mathcal{N}(0, (1/\sigma^2) I)$, then for any $x, y \in \mathbb{R}^d$,

\[
K_{\sigma^2}(x, y) = \mathbb{E} \left[ \exp(-i(x - y)^T \theta) \right].
\]

Moreover, using Euler’s formula $e^{iz} = \cos(z) + i \sin(z)$, can write real part of $\exp(-i(x - y)^T \theta)$ as

\[
\cos(x^T \theta) \cos(y^T \theta) + \sin(x^T \theta) \sin(y^T \theta) = \phi(x; \theta)^T \phi(y; \theta)
\]

where $\phi_\theta(x) := (\cos(x^T \theta), \sin(x^T \theta)) \in [-1, 1]^2$.

Therefore, we have $\mathbb{E}[\phi_\theta(x)^T \phi_\theta(y)] = K_{\sigma^2}(x, y)$.
Randomized approximation of Gaussian kernel

Procedure to construct feature expansion $\phi$:

- Draw $\theta_1, \ldots, \theta_p \sim_{iid} N(0, (1/\sigma^2)I)$.
- Construction feature expansion $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^{2p}$, by

$$
\phi(x) := \frac{1}{\sqrt{p}} (\phi_{\theta_1}(x), \ldots, \phi_{\theta_p}(x)) \quad \text{for all } x \in \mathbb{R}^d.
$$

Theorem. Let $\phi$ be as defined above. For any $x, y \in \mathbb{R}^d$, the random variable

$$
\phi(x)^T \phi(y) = \frac{1}{p} \sum_{i=1}^{p} \left[ \cos(x^T \theta_i) \cos(y^T \theta_i) + \sin(x^T \theta_i) \sin(y^T \theta_i) \right]
$$

has expectation $K_{\sigma^2}(x,y)$ and variance $O(1/p)$.

Can just use linear methods (e.g., linear SVM, linear regression) with $\phi$.
E.g., ridge regression, $O(np^2)$ time; c.f. kernel ridge regression, $O(n^3)$ time.

5. Non-separable case

Soft-margin SVMs (Cortes and Vapnik, 1995)

When training examples are not linearly separable, the (primal) SVM optimization problem

$$
\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|_2^2
\quad \text{s.t.} \quad y_i x_i^T w \geq 1 \quad \text{for all } i = 1, 2, \ldots, n
$$

has no solution.

Introduce slack variables $\xi_1, \ldots, \xi_n \geq 0$, and a trade-off parameter $C > 0$:

$$
\min_{w \in \mathbb{R}^d, \xi_1, \ldots, \xi_n \in \mathbb{R}} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \xi_i
\quad \text{s.t.} \quad y_i x_i^T w \geq 1 - \xi_i \quad \text{for all } i = 1, 2, \ldots, n,
\quad \xi_i \geq 0 \quad \text{for all } i = 1, 2, \ldots, n.
$$

which is always feasible. This is called soft-margin SVM.
(Slack variables are auxiliary variables; not needed to form the linear classifier.)

Interpretation of slack variables

For given $w, \xi_i/\|w\|_2$ is distance that $x_i$ would have to move to satisfy $y_i x_i^T w \geq 1$. 
Another interpretation of slack variables

Constraints with non-negative slack variables:

\[
\begin{align*}
\min_{w \in \mathbb{R}^d, \xi_1, \ldots, \xi_n \in \mathbb{R}} & \quad \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} & \quad y_i x_i^T w \geq 1 - \xi_i \quad \text{for all } i = 1, 2, \ldots, n, \\
& \quad \xi_i \geq 0 \quad \text{for all } i = 1, 2, \ldots, n.
\end{align*}
\]

Equivalent unconstrained form:

\[
\begin{align*}
\min_{w \in \mathbb{R}^d} & \quad \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \left[ 1 - y_i x_i^T w \right]_+.
\end{align*}
\]

Notation: \([a]_+ := \max \{0, a\}\).

\([1 - yx^T w]_+\) is hinge loss of \(w\) on example \((x, y)\).

Can also derive dual problem for soft-margin SVM.

Key takeaways

1. Maximum margin linear classifier as solution to mathematical optimization problem defined by training data.
2. Derivation of SVM dual problem; structure of SVM solution from duality.
5. Role of slack variables in soft-margin SVMs.