Support Vector Machines & Kernelization

Barna Saha

Most of the slides are made using David Sontag’s course on machine Learning at MIT
Linear Separators

- Which of these linear separators is optimal?
Support Vector Machines

- SVMs (Vapnik, 1990’s) choose the linear separator with the largest margin

- Good according to intuition, theory, practice
Support Vector Machines

- SVMs (Vapnik, 1990’s) choose the linear separator with the largest margin

- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

- Good according to intuition, theory, practice
What if the data is not linearly separable?

• General idea: the original feature space can always be mapped to a different (often some higher-dimensional feature space) where the training set is separable: \([x_1, x_2] \mapsto [\sqrt{x_1^2 + x_2^2}, \arctan(x_2/x_1)]\)

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]
What if the data is not linearly separable?

- If there is a separator which “almost” separates, find a separator that minimizes some kind of loss function.
Support vector machines: 3 key ideas

1. Use **optimization** to find solution (i.e. a hyperplane) with few errors

2. Seek **large margin** separator to improve generalization

3. Use **kernel trick** to make large feature spaces computationally efficient
Finding a perfect classifier (when one exists) using linear programming

For every data point \((x_t, y_t)\), enforce the constraint
\[
\begin{align*}
\text{for } y_t = +1, & \quad w \cdot x_t + b \geq 1 \\
\text{and for } y_t = -1, & \quad w \cdot x_t + b \leq -1
\end{align*}
\]

Equivalently, we want to satisfy all of the linear constraints
\[y_t (w \cdot x_t + b) \geq 1 \quad \forall t\]

This linear program can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

\(w\) is normal to the hyperplane \(w \cdot x + b = 0\)
Finding a perfect classifier (when one exists) using linear programming

For every data point \((x_t, y_t)\), enforce the constraint

\[
\text{for } y_t = +1, \quad w \cdot x_t + b \geq 1
\]

What happens if the data set is not linearly separable?

Equivalently, we want to satisfy all of the linear constraints

\[
y_t (w \cdot x_t + b) \geq 1 \quad \forall t
\]

This linear program can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid
Key idea #1: Allow for *slack*

\[
\min_{w, b, \xi} \sum_j \xi_j \\
(w \cdot x_j + b) y_j \geq 1 - \xi_j, \quad \forall j \quad \xi_j \geq 0
\]

“slack variables”

We now have a linear program again, and can efficiently find its optimum

For each data point:
- If functional margin $\geq 1$, don’t care
- If functional margin $< 1$, pay linear penalty
Key idea #1: Allow for *slack*

\[
\begin{align*}
\text{minimize}_{w, b, \xi} & \quad \sum_j \xi_j \\
\left( w \cdot x_j + b \right) y_j & \geq 1 - \xi_j, \quad \forall j \quad \xi_j \geq 0
\end{align*}
\]

“slack variables”

What is the optimal value \( \xi_j^* \) as a function of \( w^* \) and \( b^* \)?

If \( (w \cdot x_j + b) y_j \geq 1 \), then \( \xi_j = 0 \)

If \( (w \cdot x_j + b) y_j < 1 \), then \( \xi_j = 1 - (w \cdot x_j + b) y_j \)

Sometimes written as

\[
\left( 1 - (w \cdot x_j + b) y_j \right)_+ \quad \xi_j = \max(0, 1 - (w \cdot x_j + b) y_j)
\]
Equivalent hinge loss formulation

\[
\min_{w,b,\xi} \sum_j \xi_j \\
\left( w \cdot x_j + b \right) y_j \geq 1 - \xi_j , \forall j \quad \xi_j \geq 0
\]

Substituting \( \xi_j = \max(0, 1 - (w \cdot x_j + b) y_j) \) into the objective, we get:

\[
\min_{w,b} \sum_j \max(0, 1 - (w \cdot x_j + b) y_j)
\]

The hinge loss is defined as \( \ell_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - \hat{y}y) \)

\[
\min_{w,b} \sum_j \ell_{\text{hinge}}(y_j, w \cdot x_j + b)
\]

This is empirical risk minimization, using the hinge loss.
Key idea #2: seek large margin
What is $\gamma$ (geometric margin) as a function of $w$?

$\gamma_i = \text{Distance to } i\text{'th data point}$

$\gamma = \min_i \gamma_i$

$w \cdot x_1 + b = 1$

$w \cdot x_2 + b = 0$

$w \cdot (x_1 - x_2) = 1$

Plug in

We also know that:

$x_1 - x_2 = \gamma \frac{w}{\|w\|}$

$1 = w \cdot \left( \gamma \frac{w}{\|w\|} \right) = \frac{\gamma}{\|w\|} w \cdot w = \gamma \|w\|$

So, $\gamma = \frac{1}{\|w\|}$ (assuming there is a data point on the $w \cdot x + b = +1$ or -1 line)

Final result: can maximize $\gamma$ by minimizing $\|w\|_2$!!!
(Hard margin) support vector machines

minimize_{w,b} \quad w \cdot w \newline \left( w \cdot x_j + b \right) y_j \geq 1, \quad \forall j

- Example of a convex optimization problem
  - A quadratic program
  - Polynomial-time algorithms to solve!
- Hyperplane defined by support vectors

Non-support Vectors:
- everything else
- moving them will not change $w$

Support Vectors:
- data points on the canonical lines
Allowing for slack: “Soft margin SVM”

\[
\begin{align*}
\text{minimize}_{w,b} & \quad w \cdot w + C \sum_j \xi_j \\
& \quad (w \cdot x_j + b) y_j \geq 1 - \xi_j, \quad \forall j \quad \xi_j \geq 0
\end{align*}
\]

“slack variables”

Slack penalty: \( C > 0 \):
- \( C = \infty \rightarrow \) have to separate the data!
- \( C = 0 \rightarrow \) ignores the data entirely!

For each data point:
- If margin \( \geq 1 \), don’t care
- If margin < 1, pay linear penalty
Equivalent formulation using hinge loss

\[ \text{minimize}_{w,b} \quad w \cdot w + C \sum_j \xi_j \]
\[ (w \cdot x_j + b) y_j \geq 1 - \xi_j , \forall j \quad \xi_j \geq 0 \]

Substituting \( \xi_j = \max(0, 1 - (w \cdot x_j + b) y_j) \) into the objective, we get:

\[ \min ||w||^2 + C \sum_j \max(0, 1 - (w \cdot x_j + b) y_j) \]

Recall, the hinge loss is \( \ell_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - \hat{y} y) \)

\[ \min_{w,b} ||w||^2 + C \sum_j \ell_{\text{hinge}}(y_j, w \cdot x_j + b) \]

This is called **regularization**; used to prevent overfitting! This part is empirical risk minimization, using the hinge loss
What if the data is not linearly separable?

Use features of features of features of features....

\[
\phi(x) = \begin{pmatrix}
x^{(1)} \\
\vdots \\
x^{(n)} \\
x^{(1)}x^{(2)} \\
x^{(1)}x^{(3)} \\
\vdots \\
e^{x^{(1)}} \\
\vdots
\end{pmatrix}
\]

Feature space can get really large really quickly!
\[ \varphi(x_1, x_2) \mapsto (x_1^2, x_1^1 x_2^2, x_2^2 x_1, x_2^2) \]

Non-linear separator in the original \( x \)-space

Linear separator in the feature \( \phi \)-space

[Tommi Jaakkola]
Key idea #3: the kernel trick

- High dimensional feature spaces at no extra cost!

\[ w = \sum_{i} \alpha_i y_i x_i \]

- As a result, prediction can be performed with:

\[ \hat{y} \leftarrow \text{sign}(w \cdot \phi(x)) \]

\[ = \text{sign}\left(\left(\sum_{i} \alpha_i y_i \phi(x_i)\right) \cdot \phi(x)\right) \]

\[ = \text{sign}\left(\sum_{i} \alpha_i y_i \left(\phi(x_i) \cdot \phi(x)\right)\right) \]

\[ = \text{sign}\left(\sum_{i} \alpha_i y_i K(x_i, x)\right) \quad \text{where} \quad K(x, x') = \phi(x) \cdot \phi(x'). \]
Key idea #3: the kernel trick

- High dimensional feature spaces at no extra cost!

Kernel method enables one to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space but rather by simply computing the inner products between the images of all pairs of data in the feature space.

Often computationally cheaper than the explicit computation of the coordinates.
Polynomial kernel

\[ d=1 \]
\[ \phi(u) \cdot \phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1v_1 + u_2v_2 = u \cdot v \]

\[ d=2 \]
\[ \phi(u) \cdot \phi(v) = \begin{pmatrix} u_1^2 \\ u_1u_2 \\ u_2u_1 \\ u_2^2 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1v_2 \\ v_2v_1 \\ v_2^2 \end{pmatrix} = u_1^2v_1^2 + 2u_1v_1u_2v_2 + u_2^2v_2^2 = (u_1v_1 + u_2v_2)^2 = (u \cdot v)^2 \]

For any \( d \) (we will skip proof):
\[ \phi(u) \cdot \phi(v) = (u \cdot v)^d \]

Polynomials of degree exactly \( d \)
Common kernels

- Polynomials of degree exactly $d$
  \[ K(u, v) = (u \cdot v)^d \]

- Polynomials of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]

- Gaussian kernels
  \[ K(\vec{u}, \vec{v}) = \exp \left( -\frac{||\vec{u} - \vec{v}||^2}{2\sigma^2} \right) \]

- Sigmoid
  \[ K(u, v) = \tanh(\eta u \cdot v + \nu) \]

- And many others: very active area of research!