Support Vector Machines

Maximizing the margin for linearly separable data
Linear classifiers—Which line is better?

Pick the one with the largest margin!

\[ w \cdot x + w_0 = 0 \]

“confidence” = \( y_i (w \cdot x_i + w_0) \)
Maximize the margin

\[
\begin{align*}
\max_{\gamma, \mathbf{w}, w_0} & \quad \gamma \\
y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) & \geq \gamma, \quad \forall i \in \{1, \ldots, N\}
\end{align*}
\]

But there are many planes...
Review: Normal to a plane

\[ \mathbf{x}_i = \bar{\mathbf{x}}_i + \lambda \frac{\mathbf{w}}{||\mathbf{w}||} \]
A convention: Normalized margin
Canonical hyperplanes

\[ x_i = \bar{x}_i + \lambda \frac{\mathbf{w}}{||\mathbf{w}||} \]

 Margin maximization using canonical hyperplanes

Unnormalized problem:

\[ \max_{\gamma, \mathbf{w}, w_0} \gamma \]

\[ y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq \gamma, \forall i \in \{1, \ldots, N\} \]

Normalized Problem:

\[ \min_{\mathbf{w}, w_0} ||\mathbf{w}||^2 \]

\[ y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1, \forall i \in \{1, \ldots, N\} \]
Support vector machines (SVMs)

\[
\begin{align*}
\min_{\mathbf{w}, w_0} & \quad ||\mathbf{w}||_2^2 \\
y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) & \geq 1, \forall i \in \{1, \ldots, N\}
\end{align*}
\]

- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
  - Well-studied solution algorithms
  - Stochastic gradient descent
- Hyperplane defined by support vectors

What if the data are not linearly separable?
What if data are not linearly separable?

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

What if data are still not linearly separable?

\[
\min_{w, w_0} \|w\|^2_2
\]
\[
y_i (w \cdot x_i + w_0) \geq 1, \forall i \in \{1, \ldots, N\}
\]

• If data are not linearly separable, some points don’t satisfy margin constraint:

• How bad is the violation?

• Tradeoff margin violation with \(\|w\|\):
SVMs for non-linearly separable data, meet my friend the Perceptron...

- Perceptron was minimizing the hinge loss:
  \[ \sum_{i=1}^{N} (-y_i(w \cdot x_i + w_0))_+ \]

- SVMs minimizes the regularized hinge loss!!
  \[ \|w\|^2_2 + C \sum_{i=1}^{N} (1 - y_i(w \cdot x_i + w_0))_+ \]

Stochastic gradient descent for SVMs

- Perceptron minimization:
  \[ \sum_{i=1}^{N} (-y_i(w \cdot x_i + w_0))_+ \]

- SGD for Perceptron:
  \[ w^{(t+1)} \leftarrow w^{(t)} + \mathbb{I} \left[ y_t(w^{(t)} \cdot x_t) \leq 0 \right] y_t x_t \]

- SVMs minimization:
  \[ \|w\|^2_2 + C \sum_{i=1}^{N} (1 - y_i(w \cdot x_i + w_0))_+ \]

- SGD for SVMs:
Side note: What’s the difference between SVMs and logistic regression?

- SVM:
  $$\min_{w, w_0} \|w\|^2_2 + C \sum_{i=1}^{N} (1 - y_i (w \cdot x_i + w_0))_+$$

- Logistic regression:
  $$p(y = 1 \mid w, x) = \frac{1}{1 + e^{-(w \cdot x + w_0)}}$$

Log loss:
  $$-\log p(y = 1 \mid w, x) = \log \left( 1 + e^{-(w \cdot x + w_0)} \right)$$

Visualizing the SVM decision boundary with and without kernels
Mixture model example

\[ C = 10000 \]

Training Error: 0.270
Test Error: 0.288
Bayes Error: 0.210

\[ C = 0.01 \]

Training Error: 0.26
Test Error: 0.29
Bayes Error: 0.21

From Hastie, Tibshirani, Friedman book

Mixture model example – kernels

SVM - Degree-4 Polynomial in Feature Space

Training Error: 0.180
Test Error: 0.245
Bayes Error: 0.210

SVM - Radial Kernel in Feature Space

Training Error: 0.180
Test Error: 0.218
Bayes Error: 0.210

From Hastie, Tibshirani, Friedman book
What you need to know...

- Maximizing margin
- Derivation of SVM formulation
- Non-linearly separable case
  - Hinge loss
  - a.k.a. adding slack variables
- SVMs = Perceptron + $L_2$ regularization
- Can optimize SVMs with SGD
  - Many other approaches possible

Slack variables – Hinge loss

$$\minimize_{w, b} \ w \cdot w$$

$$\left( w \cdot x_j + b \right) y_j \geq 1 \quad , \forall j$$

- If margin $\geq 1$, don’t care
- If margin $< 1$, pay linear penalty
Side note: What’s the difference between SVMs and logistic regression?

SVM: $\min_{\beta, \beta_0} \|\beta\|^2_2 + C \sum_{i=1}^n (1 - y_i (\beta \cdot x_i + \beta_0))_+$

Logistic regression:

$p(Y = 1 \mid x, \beta) = \frac{1}{1 + e^{-(\beta \cdot x + \beta_0)}}$

Log loss:

$-\log p(Y = 1 \mid x, \beta) = \log \left(1 + e^{-(\beta \cdot x + \beta_0)} \right)$