Support Vector Machines

Lili Yao

2016/04/26
Support Vector Machines

Here we approach the two-class classification problem in a direct way:

We try and find a plane that separates the classes in feature space.

If we cannot, we get creative in two ways:

- We soften what we mean by “separates”, and
- We enrich and enlarge the feature space so that separation is possible.
What is a Hyperplane?

- A hyperplane in $p$ dimensions is a flat affine subspace of dimension $p - 1$.
- In general the equation for a hyperplane has the form
  \[
  \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0
  \]
- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.
Hyperplane in 2 Dimensions

\[ \beta = (\beta_1, \beta_2) \]

\[ \beta_1 X_1 + \beta_2 X_2 - 6 = 0 \]

\[ \beta_1 X_1 + \beta_2 X_2 - 6 = 1.6 \]

\[ \beta_1 X_1 + \beta_2 X_2 - 6 = -4 \]

\[ \beta_1 = 0.8 \]
\[ \beta_2 = 0.6 \]
Separating Hyperplanes

- If \( f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \), then \( f(X) > 0 \) for points on one side of the hyperplane, and \( f(X) < 0 \) for points on the other.

- If we code the colored points as \( Y_i = +1 \) for blue, say, and \( Y_i = -1 \) for mauve, then if \( Y_i \cdot f(X_i) > 0 \) for all \( i \), \( f(X) = 0 \) defines a **separating hyperplane**.
Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.

Constrained optimization problem

\[
\begin{align*}
\text{maximize} & \quad M \\
\text{subject to} & \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\
& \quad y_i (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \\
& \quad \text{for all } i = 1, \ldots, N.
\end{align*}
\]
Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.

Constrained optimization problem

\[
\begin{align*}
\text{maximize} & \quad M \\
\text{subject to} & \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\
& \quad y_i (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \\
& \quad \text{for all } i = 1, \ldots, N.
\end{align*}
\]

This can be rephrased as a convex quadratic program, and solved efficiently. The function `svm()` in package `e1071` solves this problem efficiently.
Non-separable Data

The data on the left are not separable by a linear boundary.

This is often the case, unless \( N < p \).
Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.
Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

The *support vector classifier* maximizes a *soft* margin.
Support Vector Classifier

\[
\begin{align*}
\text{maximize} & \quad M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\
& \quad y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \\
& \quad \epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,
\end{align*}
\]
$C$ is a regularization parameter
Sometime a linear boundary simply won’t work, no matter what value of $C$.

The example on the left is such a case.

What to do?
Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2$, $X_1^3$, $X_1X_2$, $X_1X_2^2$, ... Hence go from a $p$-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.
Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2$, $X_1^3$, $X_1X_2$, $X_1X_2^2$, ... Hence go from a $p$-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just $(X_1, X_2)$. Then the decision boundary would be of the form

$$
\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1X_2 = 0
$$

This leads to nonlinear decision boundaries in the original space (quadratic conic sections).
Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space
Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space

\[ \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0 \]
Nonlinearities and Kernels

• Polynomials (especially high-dimensional ones) get wild rather fast.

• There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of kernels.

• Before we discuss these, we must understand the role of inner products in support-vector classifiers.
Inner products and support vectors

\[ \langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— inner product between vectors} \]
Inner products and support vectors

\[ \langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— inner product between vectors} \]

- The linear support vector classifier can be represented as

\[ f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \text{— } n \text{ parameters} \]
Inner products and support vectors

\[ \langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— inner product between vectors} \]

- The linear support vector classifier can be represented as

\[ f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \text{— } n \text{ parameters} \]

- To estimate the parameters \( \alpha_1, \ldots, \alpha_n \) and \( \beta_0 \), all we need are the \( \binom{n}{2} \) inner products \( \langle x_i, x_i' \rangle \) between all pairs of training observations.
Inner products and support vectors

\[ \langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad — \text{inner product between vectors} \]

- The linear support vector classifier can be represented as

\[ f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad — \text{n parameters} \]

- To estimate the parameters \( \alpha_1, \ldots, \alpha_n \) and \( \beta_0 \), all we need are the \( \binom{n}{2} \) inner products \( \langle x_i, x_i' \rangle \) between all pairs of training observations.

It turns out that most of the \( \hat{\alpha}_i \) can be zero:

\[ f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle \]

\( S \) is the support set of indices \( i \) such that \( \hat{\alpha}_i > 0 \). [see slide 8]
Kernels and Support Vector Machines

• If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!

\[
K(x_i, x_i') = \left( 1 + \sum_{j=1}^{p} x_{ij} x_{i'j} \right)^{d}
\]

computes the inner-products needed for \(d\) dimensional polynomials — \((p+d)\) basis functions!

Try it for \(p=2\) and \(d=2\).

• The solution has the form

\[
f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)
\]
Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

\[
K(x_i, x_i') = \left( 1 + \sum_{j=1}^{p} x_{ij} x_{i'j} \right)^d
\]

computes the inner-products needed for \(d\) dimensional polynomials — \(\binom{p+d}{d}\) basis functions!
Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special *kernel functions* can do this for us. E.g.

\[
K(x_i, x_{i'}) = \left( 1 + \sum_{j=1}^{p} x_{ij}x_{i'j} \right)^d
\]

computes the inner-products needed for \( d \) dimensional polynomials — \( \binom{p+d}{d} \) basis functions!

*Try it for \( p = 2 \) and \( d = 2 \).*
Kernels and Support Vector Machines

• If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!

• Some special *kernel functions* can do this for us. E.g.

\[
K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d
\]

computes the inner-products needed for \(d\) dimensional polynomials — \(\binom{p+d}{d}\) basis functions!

*Try it for \(p = 2\) and \(d = 2\).*

• The solution has the form

\[
f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i).
\]
Radial Kernel

\[ K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2). \]

\[ f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i) \]

Implicit feature space; very high dimensional.

Controls variance by squashing down most dimensions severely.
Example: Heart Data

ROC curve is obtained by changing the threshold $0$ to threshold $t$ in $\hat{f}(X) > t$, and recording *false positive* and *true positive* rates as $t$ varies. Here we see ROC curves on training data.
Example continued: Heart Test Data

![ROC curve for different classifiers and regularization parameters.](image)

- Support Vector Classifier
- SVM: $\gamma = 10^{-3}$
- SVM: $\gamma = 10^{-2}$
- SVM: $\gamma = 10^{-1}$

- False positive rate
- True positive rate

- LDA
SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?
SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.
SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x), k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.

**OVO** One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{k\ell}(x)$. Classify $x^*$ to the class that wins the most pairwise competitions.
SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

**OVA** One versus All. Fit $K$ different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify $x^*$ to the class for which $\hat{f}_k(x^*)$ is largest.

**OVO** One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{k\ell}(x)$. Classify $x^*$ to the class that wins the most pairwise competitions.

Which to choose? If $K$ is not too large, use OVO.
Support Vector versus Logistic Regression?

With \( f(X) = \beta_0 + \beta_1X_1 + \ldots + \beta_pX_p \) can rephrase support-vector classifier optimization as

\[
\minimize \left\{ \sum_{i=1}^{n} \max[0, 1 - y_if(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}
\]

This has the form \textit{loss plus penalty}. The loss is known as the \textit{hinge loss}. Very similar to “loss” in logistic regression (negative log-likelihood).
Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (with ridge penalty) and SVM very similar.
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.
Thank you!