Support vector machines

Optimal separating hyperplanes

A main initiative in early computer science was to find separating hyperplanes among groups of data

(Rosenblatt (1958) with the perceptron algorithm)

The issue is that if there is a separating hyperplane, there is an infinite number

An optimal separating hyperplane can be generated by finding support points and bisecting them.

(Sometimes optimal separating hyperplanes are called maximum margin classifiers)

Basic linear geometry

A hyperplane in $\mathbb{R}^p$ is given by

$$\mathcal{H} = \{X \in \mathbb{R}^p : h(X) = \beta_0 + \beta^\top X = 0\}$$

(Usually it is assumed that $||\beta||_2 = 1$)

1. The vector β is normal to $\mathcal{H}$
2. For any point $X \in \mathbb{R}^p$, the (signed) length of its orthogonal complement to $\mathcal{H}$ is $h(X)$

Support vector machines (SVM)

Let $Y_i \in \{-1, 1\}$

(It is common with SVMs to code Y this way. With logistic regression, Y is commonly phrased as $\{0, 1\}$ due to the connection with Bernoulli trials)

We will generalize this to supervisors with more than 2 levels at the end

A classification rule induced by a hyperplane is

$$g(X) = \text{sgn}(X^\top \beta + \beta_0)$$

Separating hyperplanes

Our classification rule is based on a hyperplane $\mathcal{H}$

$$g(X) = \mathrm{sgn}(X^\top \beta + \beta_0)$$

A correct classification is one such that $h(X)Y > 0$ and $g(X)Y > 0$

The larger the quantity $Yh(X)$, the more "sure" the classification

Under classical separability, we can find a function such that $Y_i h(X_i) > 0$

Optimal separating hyperplane

This idea can be encoded in the following convex program

$$M \rightarrow \max_{\beta_0, \beta}, \text{ subject to}$$

$$Y_i h(X_i) \geq M \text{ for each } i \text{ and } ||\beta||_2 = 1$$

Intuition:

- We know that $Y_i h(X_i) > 0 \;\Rightarrow g(X_i) = Y_i$. Hence, larger $Y_i h(X_i) \;\Rightarrow$ "more" correct classification
- For "more" to have any meaning, we need to normalize β, thus the other constraint

Optimal separating hyperplane

Let's take the original program:

$$M \to \max_{\beta_0,\beta}, \text{ subject to}$$

$$Y_i h(X_i) \geq M \text{ for each } i \text{ and } ||\beta||_2 = 1$$

and rewrite it as

$$\min_{\beta_0,\beta} \frac{1}{2} ||\beta||_2^2 \text{ subject to}$$

$$Y_i h(X_i) \geq 1 \text{ for each } i$$

This is still a convex optimization program: quadratic criterion, linear inequality constraints

Optimal separating hyperplane

We can convert this constrained optimization problem into the Lagrangian (primal) form

$$\min_{\beta_0,\beta} \frac{1}{2} ||\beta||_2^2 - \sum_{i=1}^{n} \alpha_i [Y_i(X_i^\top \beta + \beta_0) - 1]$$

Everything is nice and smooth, so we can take derivatives..

Optimal separating hyperplane

$$\frac{1}{2}\,||\beta||_2^2 - \sum_{i=1}^{n} \alpha_i [Y_i (X_i^\top \beta + \beta_0) - 1]$$

Derivatives with respect to β and β_0:

- $\beta = \sum_{i=1}^{n} \alpha_i Y_i X_i$
- $0 = \sum_{i=1}^{n} \alpha_i Y_i$

Substituting into the Lagrangian:

$$\text{Wolfe Dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k Y_i Y_k X_i^\top X_k$$

(this is all subject to $\alpha_i \geq 0$)

We want to maximize Wolfe Dual

Optimal separating hyperplane

A side condition, known as complementary slackness states (or Karush-Kuhn-Tucker (KKT) conditions):

$$\alpha_i[1 - Y_i h(X_i)] = 0 \text{ for all } i$$

(The product of Lagrangian parameters and inequalty constraint equals 0)

This implies either:

- $\alpha_i = 0$, which happens if the constraint $Y_i h(X_i) > 1$
- $\alpha_i > 0$, which happens if the constraint $Y_i h(X_i) = 1$

Optimal separating hyperplane

Taking this relationship

$$\alpha_i[Y_i h(X_i) - 1] = 0$$

we see that, for $i = 1, \ldots, n$,

- The points (X_i, Y_i) such that $\alpha_i > 0$ are support vectors
- The points (X_i, Y_i) such that $\alpha_i = 0$ are irrelevant for classification

End result: $\hat{g}(X) = \mathrm{sgn}(X^\top \hat{\beta} + \hat{\beta}_0)$

Support vector classifier

Support vector classifier

Of course, we can't realistically assume that the data are linearly separated (even in a transformed space)

In this case, the previous program has no feasible solution

We need to introduce slack variables, ξ, that allow for overlap among the classes

These slack variables allow for us to encode training missclassifications into the optimization problem

Support vector classifier

$$M \to \max_{\beta_0, \beta, \xi_1, \ldots, \xi_n}, \quad \text{subject to}$$

$$Y_i h(X_i) \geq M \underbrace{(1 - \xi_i), \quad \xi_i \geq 0, \quad \sum \xi_i \leq t}_{\textit{new}}, \text{ for each } i$$

Note that

- t is a tuning parameter. The literature usually refers to t as a budget
- The separable case corresponds to $t = 0$

Support vector classifier

We can rewrite the problem again:

$$\min_{\beta_0,\beta,\xi} \frac{1}{2} ||\beta||_2^2, \quad \text{subject to}$$

$$Y_i h(X_i) \geq 1 \underbrace{-\xi_i, \quad \xi_i \geq 0, \quad \sum \xi_i \leq t}_{\textit{new}}, \text{ for each } i$$

Converting $\sum \xi_i \leq t$ to the Lagrangian (primal):

$$\min_{\beta_0,\beta} \frac{1}{2} ||\beta||_2^2 + \lambda \sum \xi_i \quad \text{subject to}$$

$$Y_i h(X_i) \geq 1 - \xi_i, \xi_i \geq 0, \text{ for each } i$$

SVMs: slack variables

The slack variables give us insight into the problem

- If $\xi_i = 0$, then that observation is on correct the side of the margin
- If $\xi_i = \in (0, 1]$, then that observation is on the incorrect side of the margin, but still correctly classified
- If $\xi_i > 1$, then that observation is incorrectly classified

Support vector classifier

Continuing to convert constraints to Lagrangian

$$\min_{\beta_0,\beta,\xi} \frac{1}{2}||\beta||_2^2 + \lambda \sum \xi_i \underbrace{- \sum_{i=1}^{n} \alpha_i [Y_i (X_i^\top \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{n} \gamma_i \xi_i}_{\text{remaining constraints}}$$

Necessary conditions (taking derivatives)

- $\beta = \sum_{i=1}^n \alpha_i Y_i X_i$
- $0 = \sum_{i=1}^n \alpha_i Y_i$
- $\alpha_i = \lambda - \gamma_i$

(As well as positivity constraints on Lagrangian parameters)

Support vector classifier

Substituting, we reaquire the Wolfe Dual

This, combined with the KKT conditions uniquely characterize the solution:

$$\max_{\alpha \text{ subject to: KKT + Wolfe Dual}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} Y_i Y_{i'} X_i^\top X_{i'}$$

Note: the necessary conditions $\beta = \sum_{i=1}^n \alpha_i Y_i X_i$ imply estimators of the form

- $\hat{\beta} = \sum_{i=1}^n \hat{\alpha}_i Y_i X_i$
- $\hat{\beta}^\top X = \sum_{i=1}^n \hat{\alpha}_i Y_i X_i^\top X$

SVMs: tuning parameter

We can think of t as a budget for the problem

If $t = 0$, then there is no budget and we won't tolerate any margin violations

If $t > 0$, then no more than $\lfloor t \rfloor$ observations can be misclassified

A larger t then leads to larger margins (we allow more margin violations)

SVMs: tuning parameter

Further intuition:

Like the optimal hyperplane, only observations that violate the margin determine $\mathcal{H}$

A large t allows for many violations, hence many observations factor into the fit

A small t means only a few observations do

Hence, t calibrates a bias/variance trade-off, as expected

In practice, t gets selected via cross-validation

SVMs: tuning parameter

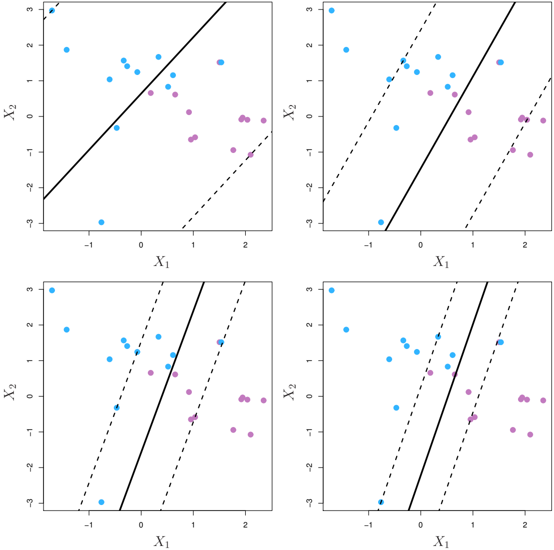

Figure: Figure 9.7 in ISL

Kernel methods

Intuition: Many methods have linear decision boundaries

We know that sometimes this isn't sufficient to represent data

Example: Sometimes we need to included a polynomial effect or a log transform in multiple regression

Sometimes, a linear boundary, but in a different space makes all the difference..

Optimal separating hyperplane

Reminder: The Wolfe dual, which gets maximized over α, produces the optimal separating hyperplane

$$\text{Wolf dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k Y_i Y_k X_i^\top X_k$$

(this is all subject to $\alpha_i \geq 0$)

A similar result holds after the introduction of slack variables

(e.g. support vector classifiers)

Important: The features only enter via

$$X^\top X' = \langle X, X' \rangle$$

Kernel Methods

Nonnegative definite matrices

Let $A \in \mathbb{R}^{p \times p}$ be a symmetric, nonnegative definite matrix:

$$z^\top A z \geq 0 \text{ for all } z \quad \text{and} \quad A^\top = A$$

Then, A has an eigenvalue expansion

$$A = UDU^\top = \sum_{j=1}^{p} d_j u_j u_j^\top$$

where $d_j \geq 0$

Observation: Each such A, generates a new inner product

$$\langle z, z' \rangle = z^\top z' = z^\top \underbrace{I}_{\text{Identity}} z'$$

$$\langle z, z' \rangle_A = z^\top A z'$$

(If we enforce A to be positive definite, then $\langle z, z \rangle_A = ||z||_A^2$ is a norm)

Nonnegative definite matrices

Suppose A_i^j is the (i, j) entry in A, and A_i is the i^{th} row

$$Az = \begin{bmatrix} A_1^\top \\ \vdots \\ A_p^\top \end{bmatrix} z = \begin{bmatrix} A_1^\top z \\ \vdots \\ A_p^\top z \end{bmatrix}$$

Note: Multiplication by A is really taking inner products with its rows.

Hence, A_i is called the (multiplication) kernel of matrix A

Kernel methods

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric, nonnegative definite kernel
Write the eigenvalue expansion of k as

$$k(X, X') = \sum_{j=1}^{\infty} \theta_j \phi_j(X) \phi_j(X')$$

with

- $\theta_j \geq 0$ (nonnegative definite)
- $\left|\left|(\theta_j)_{j=1}^{\infty}\right|\right|_2 = \sum_{j=1}^{\infty} \theta_j^2 < \infty$
- The ϕ_j are orthogonal eigenfunctions: $\int \phi_j \phi_{j'} = \delta_{j,j'}$

We can write any $f \in \mathcal{H}_k$ with two constraints

- $f(x) = \sum_{j=1}^{\infty} f_j \phi_j(x)$
- $\langle f, f \rangle_{\mathcal{H}_k} = ||f||_{\mathcal{H}_k}^2 = \sum_{j=1}^{\infty} f_j^2 / \theta_j < \infty$

Kernel: Example

Back to polynomial terms/interactions:

Form

$$k_d(X, X') = (X^\top X' + 1)^d$$

k_d has $M = \binom{p+d}{d}$ eigenfunctions

These span the space of polynomials in $\mathbb{R}^p$ with degree d

Kernel: Example

Example: Let $d = p = 2 \Rightarrow M = 6$ and

$$\begin{aligned} k(u,v) &= 1 + 2u_1v_1 + 2u_2v_2 + u_1^2v_1^2 + u_2^2v_2^2 + 2u_1u_2v_1v_2 \\ &= \sum_{k=1}^{M} \Phi_k(u)\Phi_k(v) \\ &= \Phi(u)^\top \Phi(v) \\ &= \langle \Phi(u), \Phi(v) \rangle \end{aligned}$$

where

$$\Phi(v)^\top = (1, \sqrt{2}v_1, \sqrt{2}v_2, v_1^2, v_2^2, \sqrt{2}v_1v_2)$$

Kernel: Conclusion

Let's recap:

$$\begin{aligned} k(u, v) &= 1 + 2u_1 v_1 + 2u_2 v_2 + u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 u_2 v_1 v_2 \\ &= \langle \Phi(u), \Phi(v) \rangle \end{aligned}$$

- Some methods only involve features via inner products $X^\top X' = \langle X, X' \rangle$

  (We've explicitly seen two: ridge regression and support vector classifiers)
- If we make transformations of X to $\Phi(X)$, the procedure depends on $\Phi(X)^\top \Phi(X') = \langle \Phi(X), \Phi(X') \rangle$
- We can compute this inner product via the kernel:

$$k(X, X') = \langle \Phi(X), \Phi(X') \rangle$$

(Kernel) SVMs

Kernel SVM

Recall:

$$\frac{1}{2}\,||\beta||_2^2 - \sum_{i=1}^{n} \alpha_i [Y_i(X_i^\top \beta + \beta_0) - 1]$$

Derivatives with respect to β and β_0 imply:

- $\beta = \sum_{i=1}^{n} \alpha_i Y_i X_i$
- $0 = \sum_{i=1}^{n} \alpha_i Y_i$

Write the solution function

$$h(X) = \beta_0 + \beta^\top X = \beta_0 + \sum_{i=1}^{n} \alpha_i Y_i X_i^\top X$$

Kernelize the support vector classifier $\Rightarrow$ support vector machine (SVM):

$$h(X) = \beta_0 + \sum_{i=1}^{n} \alpha_i Y_i k(X_i, X)$$

General kernel machines

After specifying a kernel function, it can be shown that many procedures have a solution of the form

$$\hat{f}(X) = \sum_{i=1}^{n} \gamma_i k(X, X_i)$$

For some $\gamma_1, \ldots, \gamma_n$

Also, this is equivalent to performing the method in the space given by the eigenfunctions of k

$$k(u, v) = \sum_{j=1}^{\infty} \theta_j \phi_j(u) \phi_j(v)$$

Also, (the) feature map is

$$\Phi = [\phi_1, \ldots, \phi_p, \ldots]$$

Kernel SVM: A reminder

The dual Lagrangian is:

$$\ell_D(\gamma) = \sum_i \gamma_i - \frac{1}{2} \sum_i \sum_{i'} \gamma_i \gamma_{i'} Y_i Y_{i'} X_i^\top X_{i'}$$

with side conditions: $\gamma_i \in [0, C]$ and $\gamma^\top Y = 0$

Let's replace the term $X_i^\top X_{i'} = \langle X_i, X_{i'} \rangle$ with $\langle \Phi(X_i), \Phi(X_{i'}) \rangle$

Kernel SVMs

Hence (and luckily) specifying Φ itself unnecessary,

(Luckily, as many kernels have difficult to compute eigenfunctions)

We need only define the kernel that is symmetric, positive definite

Some common choices for SVMs:

- Polynomial: $k(x, y) = (1 + x^\top y)^d$
- Radial basis: $k(x, y) = e^{-\tau ||x-y||_b^b}$

  (For example, $b = 2$ and $\tau = 1/(2\sigma^2)$ is (proportional to) the Gaussian density)

Kernel SVMs: Summary

Reminder: the solution form for SVM is

$$\beta = \sum_{i=1}^{n} \alpha_i Y_i X_i$$

Kernelized, this is

$$\beta = \sum_{i=1}^{n} \alpha_i Y_i \Phi(X_i)$$

Therefore, the induced hyperplane is:

$$\begin{aligned} h(X) = \Phi(X)^\top \beta + \beta_0 &= \sum_{i=1}^{n} \alpha_i Y_i \langle \Phi(X), \Phi(X_i) \rangle + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i Y_i k(X, X_i) + \beta_0 \end{aligned}$$

The final classification is still $\hat{g}(X) = \mathrm{sgn}(\hat{h}(X))$

SVMs via penalization

SVMs via penalization

NOTE: SVMs can be derived from penalized loss methods

The support vector classifier optimization problem:

$$\min_{\beta_0,\beta} \frac{1}{2} ||\beta||_2^2 + \lambda \sum \xi_i \text{ subject to}$$

$$Y_i h(X_i) \geq 1 - \xi_i, \xi_i \geq 0, , \text{ for each } i$$

Writing $h(X) = \Phi(X)^\top \beta + \beta_0$, consider

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} [1 - Y_i h(X_i)]_+ + \tau ||\beta||_2^2$$

These optimization problems are the same!

(With the relation: $2\lambda = 1/\tau$)

SVMs via penalization

The loss part is the hinge loss function

$$\ell(X, Y) = [1 - Yh(X)]_+$$

The hinge loss approximates the zero-one loss function underlying classification

It has one major advantage, however: convexity

Surrogate losses: convex relaxation

Looking at

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} [1 - Y_i h(X_i)]_+ + \tau \left|\left|\beta\right|\right|_2^2$$

It is tempting to minimize (analogous to linear regression)

$$\sum_{i=1}^{n} \mathbf{1}(Y_i \neq \hat{g}(X_i)) + \tau \left|\left|\beta\right|\right|_2^2$$

However, this is nonconvex (in $u = h(X)Y$)

A common trick is to approximate the nonconvex objective with a convex one

(This is known as convex relaxation with a surrogate loss function)

Surrogate losses

Idea: We can use a surrogate loss that mimics this function while still being convex

It turns out we have already done that! (twice)

- Hinge: $[1 - Yh(X)]_+$
- Logistic: $\log(1 + e^{-Yh(X)})$

Multiclass classification

Multiclass SVMs

Sometimes, it becomes necessary to do multiclass classification

There are two main approaches:

- One-versus-one
- One-vesus-all

Multiclass SVMs: One-versus-one

Here, for G possible classes, we run $G(G-1)/2$ possible pairwise classifications

For a given test point X, we find $\hat{g}_k(X)$ for $k = 1, \ldots, G(G-1)/2$ fits

The result is a vector $\hat{G} \in \mathbb{R}^G$ with the total number of times X was assigned to each class

We report $\hat{g}(X) = \arg\max_g \hat{G}$

This approach uses all the class information, but can be slow

Multiclass SVMs: One-vesus-all

Here, we fit only G SVMs by respectively collapsing over all size $G - 1$ subsets of $\{1, \ldots, G\}$

(This is compared with $G(G-1)/2$ comparisons for one-versus-one)

Take all $\hat{h}_g(X)$ for $g = 1, \ldots, G$, where class g is coded 1 and "the rest" is coded -1

Assign $\hat{g}(X) = \arg\max_g \hat{h}_g(X)$

Background: Structural Risk Minimization

Capacity and Generalization

- Generalization: Figure out similarities between already-seen data and new data
  - Too much: "Square piece of paper? That's a $100 bill"
- Capacity: Ability to allocate new categories for data
  - Too much: "#L26118670? It's a fake; all $100 bills I've seen had other serial numbers"
- They are competitive with one another
- How to strike the right balance?

Empirical Risk

- We are given n observations $(\mathbf{x}_i, y_i)$
  - $\mathbf{x}_i \in \mathbb{R}^p$
  - $y_i \in \{-1, 1\}$
- Learn $y = f(\mathbf{x}, \alpha)$ by tuning α
- Expected test error (risk) and empirical risk:

$$R(\alpha) = \frac{1}{2} \int |y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y)$$

$$R_{emp}(\alpha) = \frac{1}{2l} \sum |y_i - f(\mathbf{x}_i, \alpha)|$$

Risk Bound

- For 0/1 loss and with probability $1 - \eta$, $0 < \eta < 1$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(1 + \log \frac{2n}{h}) - \log \frac{\eta}{4}}{n}}$$

  where $h \in \mathbb{N}$ is the Vapnik-Chervonenkis (VC) dimension
- Second term: "VC confidence"

Importance of Risk Bound

1. Not dependent on $P(\mathbf{x}, y)$
2. lhs not computable
3. rhs computable if we know h

- For a given task, choose the machine that minimizes the risk bound!
- Even when bound not tight, we can contrast "tightness" of various families of machines

THE VC DIMENSION

- For a family of functions $f(\alpha)$:
  - Choose a set of n points
  - Label them in any way
  - $\exists\alpha$ s.t. $f(\alpha)$ can recognize ("shatter") them
- Then $f(\alpha)$ has VC at least n

Example: Hyperplanes in $\mathbb{R}^n$

- Choosing 4 planar points:
  - they can't be separated by one line for all of their possible labelings (one labeling will be inseparable)
- Similarly, $p + 1$ points in $\mathbb{R}^p$ can't be separated for all labelings
- So the VC dimension of hyperplanes in $\mathbb{R}^p$ is $p + 1$