Lecture 8: Support Vector Machines
STATS 202: Statistical Learning and Data Science

Linh Tran
stat202@gmail.com

Department of Statistics
Stanford University

July 21, 2025

Announcements

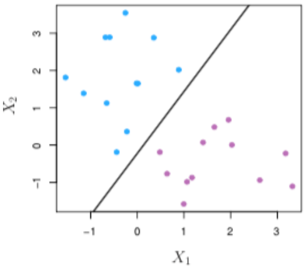
- Midterm is being graded (maybe mid-week?)
- Homework 2 grading is complete
- Homework 3 is up.
  - Due next Monday.
- Final projects due in 3 weeks.
- Final exam is on August 16 (7 PM - 10 PM)
- Survey is still up (closing this Tuesday)
- No session this Friday.

Outline

Support Vector Machines

- Maximal margin classifier
- Support vector classifier
- Support vector machine

Support Vector Machines



Support vector machines are (generally) classifiers

- Linear (like logistic regression)
- Non-probabilistic (unlike logistic regression)

Hyperplanes and normal vectors

Consider a p-dimensional space of predictors

- A *hyperplane* is an affine space which separates the space into two regions

Hyperplanes and normal vectors

Consider a p-dimensional space of predictors

- A *hyperplane* is an affine space which separates the space into two regions
- The normal vector $\beta = (\beta_1, \ldots, \beta_p)$, is a unit vector $\sum_{j=1}^{p} \beta_j^2 = \|\beta\| = 1$ which is orthogonal to the hyperplane

Hyperplanes and normal vectors

Consider a p-dimensional space of predictors

- A *hyperplane* is an affine space which separates the space into two regions
- The normal vector $\beta = (\beta_1, \ldots, \beta_p)$, is a unit vector $\sum_{j=1}^{p} \beta_j^2 = \|\beta\| = 1$ which is orthogonal to the hyperplane
- The deviation between a point $(x_1, \ldots, x_p)$ and the hyperplane is the dot product
  - If the hyperplane goes through the origin

$$x \cdot \beta = x_1\beta_1 + \cdots + x_p\beta_p \tag{1}$$

Hyperplanes and normal vectors

Consider a p-dimensional space of predictors

- A *hyperplane* is an affine space which separates the space into two regions
- The normal vector $\beta = (\beta_1, \ldots, \beta_p)$, is a unit vector $\sum_{j=1}^{p} \beta_j^2 = \|\beta\| = 1$ which is orthogonal to the hyperplane
- The deviation between a point $(x_1, \ldots, x_p)$ and the hyperplane is the dot product
  - If the hyperplane goes through the origin

$$x \cdot \beta = x_1\beta_1 + \cdots + x_p\beta_p \tag{1}$$

  - If the hyperplane is displaced from the origin by $-\beta_0$

$$\beta_0 + x \cdot \beta = \beta_0 + x_1\beta_1 + \cdots + x_p\beta_p \tag{2}$$

Hyperplanes and normal vectors

Consider a p-dimensional space of predictors

- A *hyperplane* is an affine space which separates the space into two regions
- The normal vector $\beta = (\beta_1, \ldots, \beta_p)$, is a unit vector $\sum_{j=1}^{p} \beta_j^2 = \|\beta\| = 1$ which is orthogonal to the hyperplane
- The deviation between a point $(x_1, \ldots, x_p)$ and the hyperplane is the dot product
  - If the hyperplane goes through the origin
$$x \cdot \beta = x_1\beta_1 + \cdots + x_p\beta_p \tag{1}$$
  - If the hyperplane is displaced from the origin by $-\beta_0$
$$\beta_0 + x \cdot \beta = \beta_0 + x_1\beta_1 + \cdots + x_p\beta_p \tag{2}$$
- The sign of the dot product tells us on which side of the hyperplane the point lies

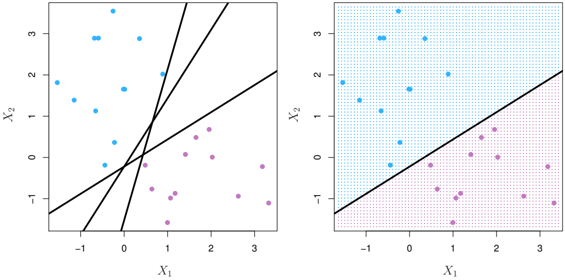
The maximal margin classifier

Suppose we have a classification problem with response $Y = -1$ or $Y = 1$.

The maximal margin classifier

Suppose we have a classification problem with response $Y = -1$ or $Y = 1$.

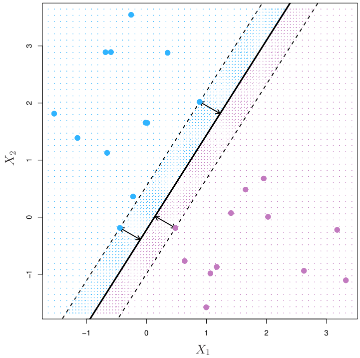
- If the classes can be separated (most likely) there will be an infinite number of hyperplanes separating the classes
- Which one should we choose?



The maximal margin classifier

Idea: Select the classifier with the *maximal margin*

- Draw the largest possible empty margin around the hyperplane
- Out of all possible hyperplanes that separate the 2 classes, choose the one with the widest margin (in both directions)



The maximal margin classifier

We can frame this as an optimization problem, i.e.

$$\max_{\beta_0,\beta_1,\ldots,\beta_p} M \tag{3}$$

subject to

- $||\beta|| = 1$
- $y_i \underbrace{\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right)}_{\text{How far } x_i \text{ is from the hyperplane}} \geq M \;\forall\; i = 1, \ldots, n$

where M is the width of the margin (in either direction)

n.b. the sign of $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$ indicates the class

This is numerically hard to optimize

Estimating the maximal margin classifier

We can reformulate the problem by *normalizing* for $\|\beta\|$, i.e.

$$\max_{\beta_0,\beta} M \tag{4}$$

subject to

- $\frac{1}{\|\beta\|} y_i(\beta_0 + x_i^\top \beta) \geq M \ \forall \ i = 1, \ldots, n$

Estimating the maximal margin classifier

We can reformulate the problem by *normalizing* for $\|\beta\|$, i.e.

$$\max_{\beta_0,\beta} M \tag{4}$$

subject to

- $\frac{1}{\|\beta\|} y_i(\beta_0 + x_i^\top \beta) \geq M \; \forall \; i = 1, \ldots, n$

or, equivalently,

- $y_i(\beta_0 + x_i^\top \beta) \geq M\|\beta\| \; \forall \; i = 1, \ldots, n$

Estimating the maximal margin classifier

Setting $||\beta|| = 1/M$, we have

$$\max_{\beta_0,\beta} \frac{1}{\|\beta\|} = \min_{\beta_0,\beta} \|\beta\| = \min_{\beta_0,\beta} \quad \frac{1}{2}\|\beta\|^2 \tag{5}$$

subject to

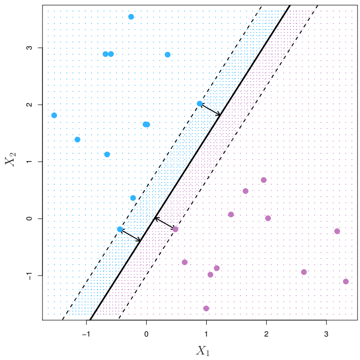
- $y_i(\beta_0 + x_i^\top \beta) \geq 1 \; \forall \; i = 1, \ldots, n$

This is a quadratic optimization problem (i.e. easier to solve)

- Typically solved using Lagrange duality

Support vectors

The vectors (i.e. observations) that fall on the margin (and determine the solution) are called *support vectors*:

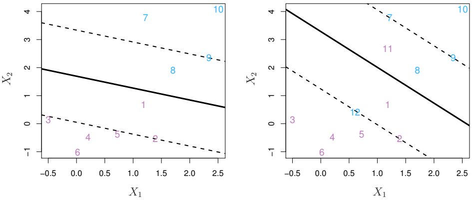


n.b. Only these points affect our estimation of the separating hyperplane.

Problem: It is not always possible (or desireable) to separate the points using a hyperplane.

Support vector classifier:

- Relaxes the maximal margin classifier, using a *soft margin*
- Allows a number of points points to be on the wrong side of the margin or hyperplane



The support vector classifier

Building this into our optimization problem gives:

$$\max_{\beta_0,\beta,\epsilon} M \tag{6}$$

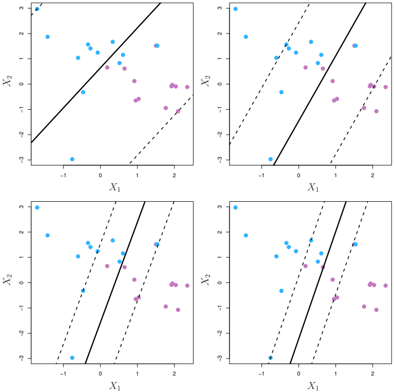
subject to

- $\|\beta\| = 1$
- $y_i(\beta_0 + x_i^\top \beta) \geq M(1 - \epsilon_i)\ \forall\ i = 1, \ldots, n$
- $\epsilon_i \geq 0\ \forall\ i = 1, \ldots, n$ and $\sum_{i=1}^n \epsilon_i \leq C$

where

- M is the width of the margin (in either direction)
- $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ are called *slack* variables
- C is called the *budget*

Tuning the budget, C



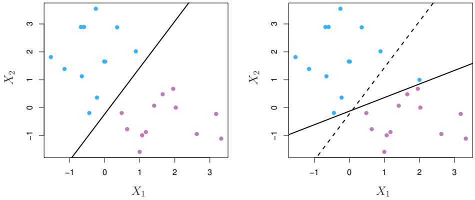
Higher C means:

- More tolerance for errors
- Larger (estimated) margins

Bias-variance trade off

C is typically chosen via cross-validation

- Larger C leads to classifers that have lower variance, but potentially have higher bias
- Smaller C leads to classifiers that are highly fit to the data, which may have low bias but high variance
  - If C is too low we can overfit
  - e.g. With the maximal margin classifier ($C = 0$), adding one observation can dramatically change the classifier



Estimating the support vector classifier

(Similar to before) we can reformulate the problem, i.e.

$$\min_{\beta_0,\beta,\epsilon} \quad \frac{1}{2}\|\beta\|^2 + D\sum_{i=1}^{n}\epsilon_i \tag{7}$$

subject to

- $y_i(\beta_0 + x_i^\top \beta) \geq (1 - \epsilon_i) \; \forall \; i = 1, \ldots, n$
- $\epsilon_i \geq 0 \; \forall \; i = 1, \ldots, n$

Estimating the support vector classifier

(Similar to before) we can reformulate the problem, i.e.

$$\min_{\beta_0, \beta, \epsilon} \quad \frac{1}{2}\|\beta\|^2 + D\sum_{i=1}^{n} \epsilon_i \tag{7}$$

subject to

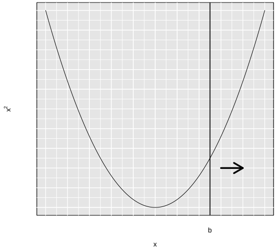
- $y_i(\beta_0 + x_i^\top \beta) \geq (1 - \epsilon_i) \; \forall \; i = 1, \ldots, n$
- $\epsilon_i \geq 0 \; \forall \; i = 1, \ldots, n$

The penalty $D \geq 0$ is inversely related to C, i.e.

- Smaller D means wider (estimated) margins
- Larger D means narrower (estimated) margins

This is (still) a quadratic optimization problem

When dealing with optimization constraints



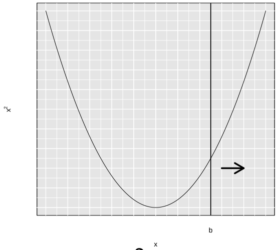
$$\min_x x^2 : x \geq b \qquad (8)$$

can be re-written as a (Lagrangian) loss function

$$L(x, \alpha) = x^2 - \alpha(x - b) \qquad (9)$$

Lagrange duality

When dealing with optimization constraints



$$\min_x x^2 : x \geq b \qquad (8)$$

can be re-written as a (Lagrangian) loss function

$$L(x, \alpha) = x^2 - \alpha(x - b) \qquad (9)$$

We then solve for it via

$$\min_x \max_\alpha L(x, \alpha) : \alpha \geq 0 \qquad (10)$$

Causes min to fight the max, i.e.

$$\begin{aligned} x < b &\Rightarrow \max_\alpha -\alpha(x - b) = \infty \\ x > b &\Rightarrow \max_\alpha -\alpha(x - b) = 0 \\ x = b &\Rightarrow L(x, \alpha) = x^2 - 0 \end{aligned}$$

Similar to the Maximal Margin Classifier:

- We can apply a Lagrange multipliers for our (constrained) optimization problem.
  - e.g. Karush-Kuhn-Tucker multipliers.
- This reduces our problem to finding $\alpha_1, \ldots, \alpha_n$ such that:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \underbrace{(x_i \cdot x_{i'})}_{\text{inner product}} \quad (11)$$

subject to

- $0 \leq \alpha_i \leq D \; \forall \; i = 1, \ldots, n$
- $\sum_i \alpha_i y_i = 0$

This only depends on the training sample inputs through the inner products $(x_i \cdot x_j)$ for every pair of points i, j

Kernel matrix

A key property of support vector classifiers:

- To *find the hyperplane* and *make predictions* all we need is the dot product between any pair of input vectors:

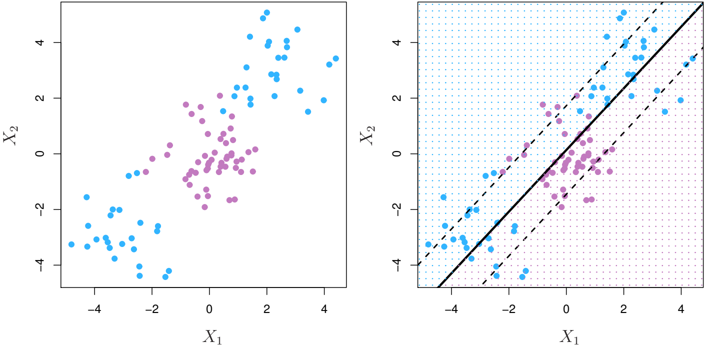
$$K(i, k) = (x_i \cdot x_k) = \langle x_i, x_k \rangle = \sum_{j=1}^{p} x_{ij} x_{kj} \qquad (12)$$

- We call this the *kernel matrix*.

Non-linear boundaries

The support vector classifier can only produce a linear boundary.

Example:



Non-linear boundaries

Recall: In *logistic regression*, we dealt with this problem by adding transformations of the predictors, e.g.

- For a linear boundary:

$$\log\left[\frac{P(Y=+1|X)}{P(Y=-1|X)}\right] = \underbrace{\beta_0 + \beta_1 X_1 + \beta_2 X_2}_{\text{set equal to 0 and solve for } X_1, X_2} \tag{13}$$

- For a quadratic boundary:

$$\log\left[\frac{P(Y=+1|X)}{P(Y=-1|X)}\right] = \underbrace{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2}_{\text{set equal to 0 and solve for } X_1, X_2} \tag{14}$$

Non-linear boundaries

For support vector classifiers: We can do the same thing, e.g.

- For a linear hyperplane:

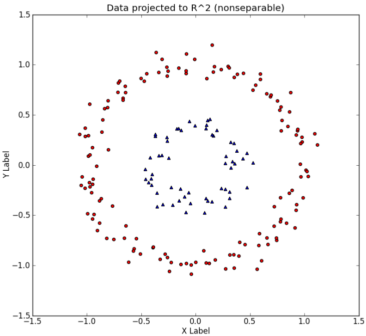
$$\underbrace{\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0}_{\text{estimate } \beta\text{'s directly}} \tag{15}$$

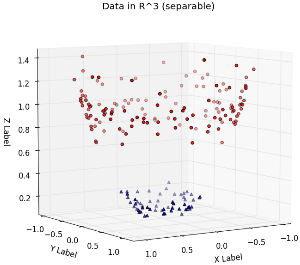
- Projecting onto a $3D$ space (X_1, X_2, X_1^2):

$$\underbrace{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 = 0}_{\text{estimate } \beta\text{'s directly}} \tag{16}$$

  - Still a linear boundary, but now in $3D$ space
  - Boundary is now quadratic in X_1

Example projection:





Left: Sample space under (X_1, X_2)
Right: Sample space under $(X_1, X_2, X_1^2 \cdot X_2^2)$

Non-linear boundaries

One approach:

- Add polynomial terms up to degree d, i.e.
  $$Z = (X_1, X_1^2, \ldots, X_1^d, X_2, X_2^2, \ldots, X_2^d, \ldots, X_p, X_p^2, \ldots, X_p^d) \quad (17)$$
- Fit a support vector classifier on the expanded set of predictors

Non-linear boundaries

One approach:

- Add polynomial terms up to degree d, i.e.
$$Z = (X_1, X_1^2, \ldots, X_1^d, X_2, X_2^2, \ldots, X_2^d, \ldots, X_p, X_p^2, \ldots, X_p^d) \quad (17)$$
- Fit a support vector classifier on the expanded set of predictors

Question: Does this make the computation more expensive?

Non-linear boundaries

One approach:

- Add polynomial terms up to degree d, i.e.

$$Z = (X_1, X_1^2, \ldots, X_1^d, X_2, X_2^2, \ldots, X_2^d, \ldots, X_p, X_p^2, \ldots, X_p^d) \quad (17)$$

- Fit a support vector classifier on the expanded set of predictors

Question: Does this make the computation more expensive?

- Recall that all we need to compute is the dot product:

$$x_i \cdot x_k = \langle x_i, x_k \rangle = \sum_{j=1}^{p} x_{ij} x_{kj} \quad (18)$$

- With the expanded set of predictors, we need:

$$z_i \cdot z_k = \langle z_i, z_k \rangle = \sum_{j=1}^{p} \sum_{\ell=1}^{d} x_{ij}^{\ell} x_{kj}^{\ell} \quad (19)$$

Kernels

Rather than expanding our predictors, we could instead use *kernels* $K(i, k)$:

- Always *positive semi-definite*, i.e. it is symmetric and has no negative eigenvalues
- Quantifies the similarity of two observations

Example:
Our support vector classifier is equivalent to using the (linear) kernel

$$K(x_i, x_i') = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{20}$$

The kernel trick

Expand predictor set:

- Find a mapping Φ which expands the original set of predictors $X_1, \ldots, X_p$. For example,

$$\Phi(X) = (X_1, X_2, X_1^2)$$

- For each pair of samples, compute:

$$K(i, k) = \langle \Phi(x_i), \Phi(x_k) \rangle.$$

The kernel trick

Expand predictor set:

- Find a mapping Φ which expands the original set of predictors $X_1, \ldots, X_p$. For example,

$$\Phi(X) = (X_1, X_2, X_1^2)$$

- For each pair of samples, compute:

$$K(i,k) = \langle \Phi(x_i), \Phi(x_k) \rangle.$$

Define a kernel:

- Prove that a function $f(\cdot,\cdot)$ is positive definite. For example:

$$f(x_i, x_k) = (1 + \langle x_i, x_k \rangle)^2 .$$

- For each pair of samples, compute:

$$K(i,k) = f(x_i, x_k).$$

The kernel trick

Expand predictor set:

- Find a mapping Φ which expands the original set of predictors $X_1, \ldots, X_p$. For example,

$$\Phi(X) = (X_1, X_2, X_1^2)$$

- For each pair of samples, compute:

$$K(i,k) = \langle \Phi(x_i), \Phi(x_k) \rangle.$$

Define a kernel:

- Prove that a function $f(\cdot, \cdot)$ is positive definite. For example:

$$f(x_i, x_k) = (1 + \langle x_i, x_k \rangle)^2 .$$

- For each pair of samples, compute:

$$K(i,k) = f(x_i, x_k).$$

Often much easier!

The kernel trick

Example: The polynomial kernel with $d = 2$ (and $p = 2$).

$$
\begin{aligned}
K(x, x') &= f(x, x') = \big(1 + \langle x, x' \rangle\big)^2 \\
&= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2' \\
&= 1 + \sqrt{2}x_1 \sqrt{2}x_1' + \sqrt{2}x_2 \sqrt{2}x_2' + x_1^2 (x_1')^2 + x_2^2 (x_2')^2 \\
&\quad + \sqrt{2}x_1 x_2 \sqrt{2}x_1' x_2'
\end{aligned} \tag{21}
$$

The kernel trick

Example: The polynomial kernel with $d = 2$ (and $p = 2$).

$$
\begin{aligned}
K(x, x') &= f(x, x') = \left(1 + \langle x, x' \rangle\right)^2 \\
&= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2' \\
&= 1 + \sqrt{2}x_1 \sqrt{2}x_1' + \sqrt{2}x_2 \sqrt{2}x_2' + x_1^2 (x_1')^2 + x_2^2 (x_2')^2 \\
&\quad + \sqrt{2}x_1 x_2 \sqrt{2}x_1' x_2'
\end{aligned} \tag{21}
$$

This is equivalent to the expansion:

$$
\Phi(X) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)
$$

giving us

$$
K(i, k) = \langle \Phi(x_i), \Phi(x_k) \rangle \tag{22}
$$

The kernel trick

Example: The polynomial kernel with $d = 2$ (and $p = 2$).

$$\begin{aligned}
K(x, x') = f(x, x') &= \left(1 + \langle x, x' \rangle\right)^2 \\
&= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + 2x_1 x_1' + 2x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' x_2 x_2' \\
&= 1 + \sqrt{2}x_1 \sqrt{2}x_1' + \sqrt{2}x_2 \sqrt{2}x_2' + x_1^2 (x_1')^2 + x_2^2 (x_2')^2 \\
&\quad + \sqrt{2}x_1 x_2 \sqrt{2}x_1' x_2'
\end{aligned} \tag{21}$$

This is equivalent to the expansion:

$$\Phi(X) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

giving us

$$K(i, k) = \langle \Phi(x_i), \Phi(x_k) \rangle \tag{22}$$

- Feature engineering is *"automated"* for us.
- Computing $K(x_i, x_k)$ directly is $O(p)$.

Defining kernels

How do we define kernels to use?

- Derive a bilinear function $f(\cdot, \cdot)$.
- Prove that $f(\cdot, \cdot)$ is positive definite (PD).

Defining kernels

How do we define kernels to use?

- Derive a bilinear function $f(\cdot, \cdot)$.
- Prove that $f(\cdot, \cdot)$ is positive definite (PD).

The common approach

- Combining PD kernels we are already familiar with.
  - e.g. sums, products, etc.
- Functions of PD kernels are PD.

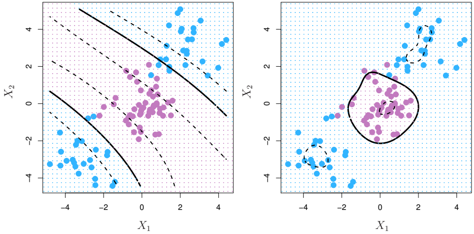
Common kernels

- The polynomial kernel:

$$K(x_i, x_k) = (1 + \langle x_i, x_k \rangle)^d$$

- The radial basis kernel:

$$K(x_i, x_k) = \exp\left(-\gamma \underbrace{\sum_{j=1}^{p} (x_{ip} - x_{kp})^2}_{\text{Euclidean } d(x_i, x_k)} \right)$$



Kernel properties

- Kernels define *similarity* between two samples, x_i and x_k.
- We can apply kernels even if we don't know what the transformations are.
- Kernels can result expansions that are an infinite number of transformations

Kernel properties

- Kernels define *similarity* between two samples, x_i and x_k.
- We can apply kernels even if we don't know what the transformations are.
- Kernels can result expansions that are an infinite number of transformations

Example: Assume $p = 1$ and $\gamma > 0$

$$
\begin{aligned}
e^{-\gamma(x_i - x_k)^2} &= e^{-\gamma x_i^2 + 2\gamma x_i x_k - \gamma x_k^2} \\
&= e^{-\gamma x_i^2 - \gamma x_k^2}\left(1 + \frac{2\gamma x_i x_k}{1!} + \frac{(2\gamma x_i x_k)^2}{2!} + \cdots\right) \\
&= e^{-\gamma x_i^2 - \gamma x_k^2}\left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}}x_i \sqrt{\frac{2\gamma}{1!}}x_k + \sqrt{\frac{(2\gamma)^2}{2!}}x_i^2 \sqrt{\frac{(2\gamma)^2}{2!}}x_k^2 + \cdots\right) \\
&= \langle \Phi(x_i), \Phi(x_k) \rangle
\end{aligned} \tag{23}
$$

$$
\text{where } \Phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \cdots\right] \tag{24}
$$

Multiclass approaches

SVM's don't generalize well to more than 2 class.

Two main approaches:

1. **One vs one**: Construct $\binom{k}{2}$ SVMs comparing every pair of classes. Apply all SVMs to a test observation and pick the class that wins the most one-on-one challenges.
2. **One vs all**: For each class k, construct an SVM β^k coding class k as 1 and all other classes as -1. Assign a test observation to class k^*, such that the distance from x_i to the hyperplace defined by β^k is the largest.

Relationship to logistic regression

In support vector classifiers: We can formulate

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \tag{25}$$

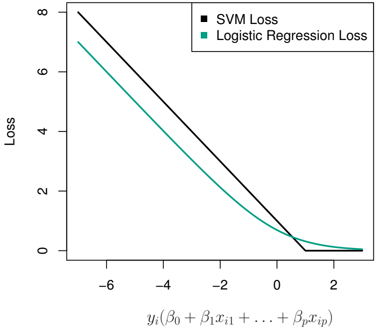
as a Loss + Penalty optimization:

$$\min_{\beta_0, \beta} \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{26}$$

In logistic regression: we optimize

$$\min_{\beta_0, \beta} \sum_{i=1}^{n} \log[1 + e^{-y_i f(x_i)}] \tag{27}$$

Comparing the losses



- When the classes are well separated, SVMs behave better
- When lots of overlap in classes, logistic regression preferred

What about the kernels?

Many previous papers indicated that kernels are unique to SVMs.

- Can logistic regression also use kernels?

What about the kernels?

Many previous papers indicated that kernels are unique to SVMs.

- Can logistic regression also use kernels?

Answer: Yes (using the *Representer theorem*)

Kernel logistic regression

$$\hat{f}(x) = \log\left[\frac{\hat{P}(Y=+1|X)}{\hat{P}(Y=-1|X)}\right] \tag{28}$$

$$= \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i K(x, x_i) \tag{29}$$

What about probabilities?

Recall: logistic regression can provide probability estimates

- Can SVMs as well?

What about probabilities?

Recall: logistic regression can provide probability estimates

- Can SVMs as well?

Answer: Yes (using logistic regression)

Let $g(x) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$

Platt scaling

$$\mathbb{P}(y = 1|x) = \frac{1}{1 + exp(Ag(x) + B)} \qquad (30)$$

n.b. Typically done via cross-validation.

This is called Platt scaling.

References

[1] ISL. Chapter 9

[2] ESL. Chapter 12.1-12.3