SVMs & Good Practices for Machine Learning

Machine Learning Decal

Hosted by Machine Learning at Berkeley

Agenda

Hard Margin SVM

Soft Margin SVM

Solving the SVM

Kernels

SVMS in Practice

Debugging ML Algorithms

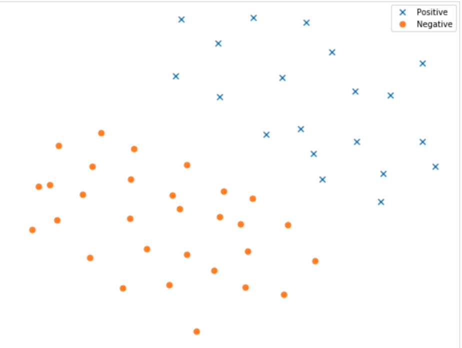
Applying ML Algorithms

Questions

Hard Margin SVM

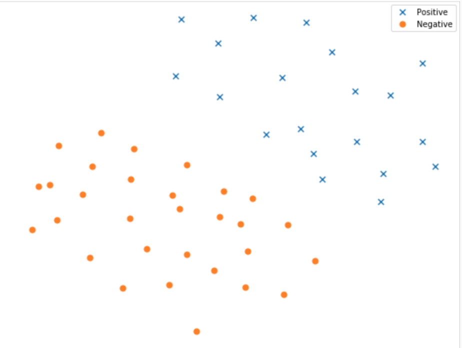
Let's talk about classification today

We want to classify data.



Let's talk about classification today

We want to classify data.



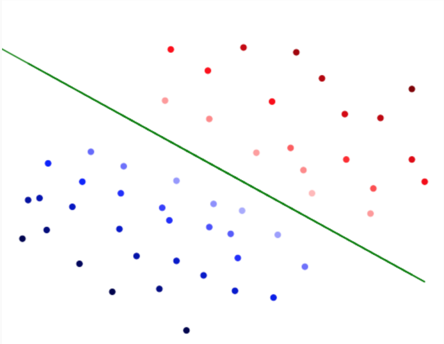
Let's take it simple. Our data is nicely divided (such as above), and we have 2 classes.

Let's talk about classification today

What can we do about it?

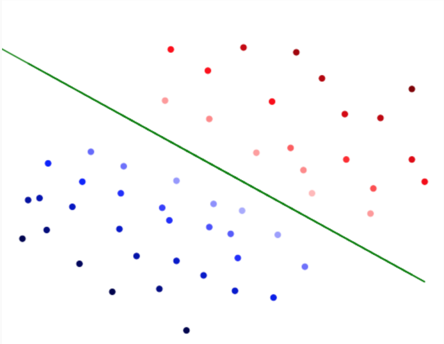
Let's talk about classification today

What can we do about it? Draw a line through it!



Let's talk about classification today

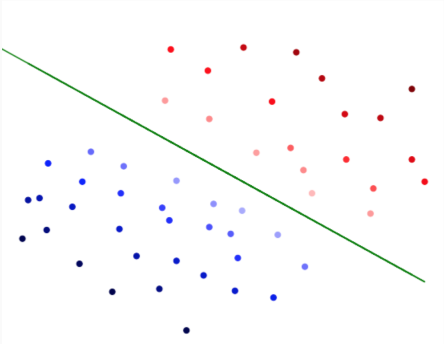
What can we do about it? Draw a line through it!



What makes this line good?

Let's talk about classification today

What can we do about it? Draw a line through it!



What makes this line good? It's far from the data points near it.

What are SVMs?

- Supervised classification algorithm

What are SVMs?

- Supervised classification algorithm
- Finds the optimal line, or "hyperplane" (line in multiple dimensions) between training points

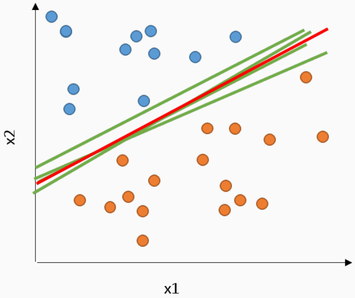
What are SVMs?

- Supervised classification algorithm
- Finds the optimal line, or "hyperplane" (line in multiple dimensions) between training points
- Hyperplane is far from nearby points; results in a nice separation

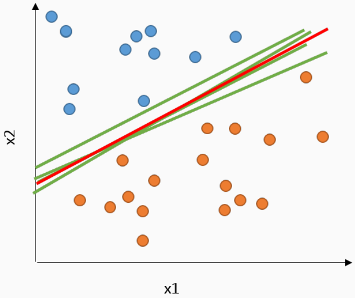
What are SVMs?

- Supervised classification algorithm
- Finds the optimal line, or "hyperplane" (line in multiple dimensions) between training points
- Hyperplane is far from nearby points; results in a nice separation
- Widely used in practice

Optimal Hyperplane

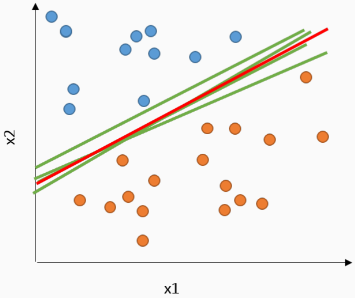


Optimal Hyperplane



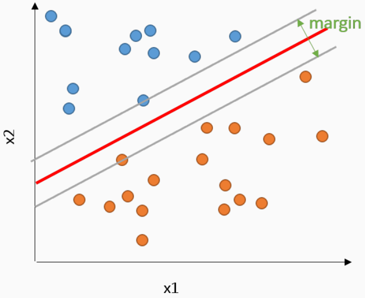
- Correctly separates all data points if possible

Optimal Hyperplane

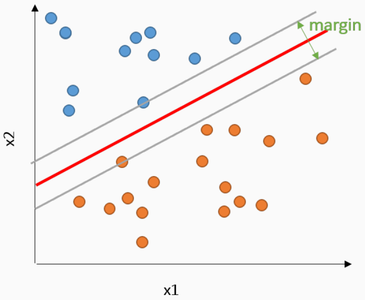


- Correctly separates all data points if possible
- Furthest away from all data points

Margin

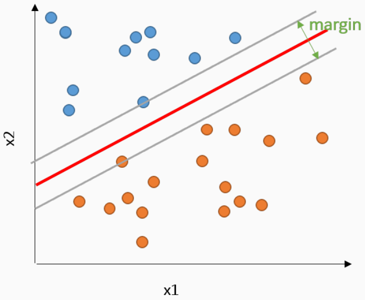


Margin



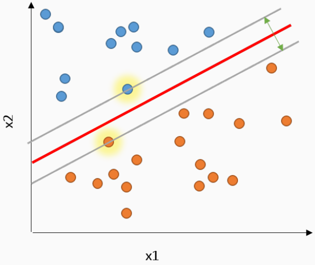
- Empty region with no data points

Margin



- Empty region with no data points
- Twice the distance from the hyperplane to the closest data point

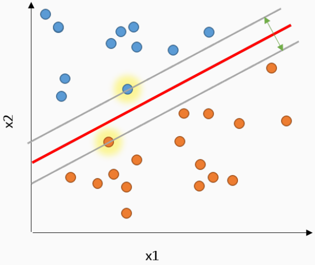
Support Vectors



(a) Linearly Separable

- Data points that lie on margin

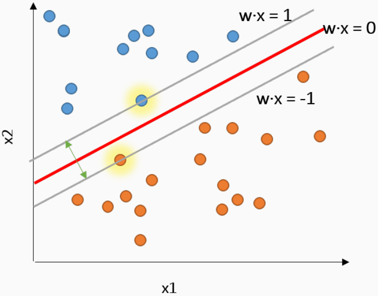
Support Vectors



(a) Linearly Separable

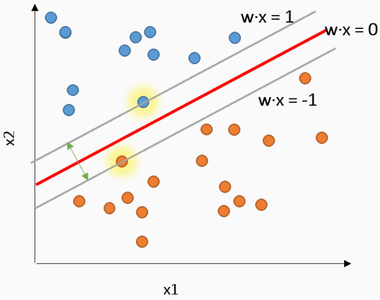
- Data points that lie on margin
- Changing support vectors changes the decision boundary

SVM Constraints



- Decision Boundary: $w \cdot x + b = 0$

SVM Constraints



- Decision Boundary: $w \cdot x + b = 0$
- Edge of Margin: $w \cdot x + b = 1$ and $w \cdot x + b = -1$

SVM Constraints

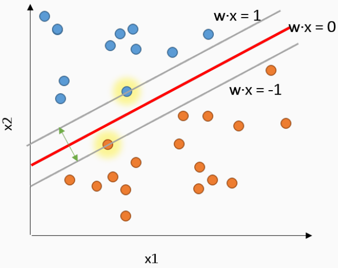


Figure 2: $y = 1$ for blue dots; $y = -1$ for orange dots

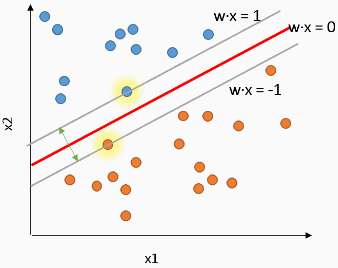


Figure 2: $y = 1$ for blue dots; $y = -1$ for orange dots

Constraint:

- For all $y_i = 1$, $w \cdot x_i + b \geqslant 1$

SVM Constraints

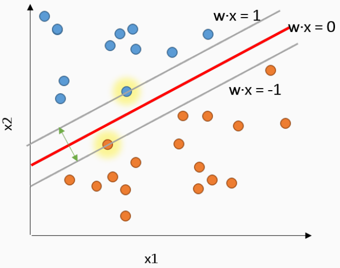


Figure 2: $y = 1$ for blue dots; $y = -1$ for orange dots

Constraint:

- For all $y_i = 1$, $w \cdot x_i + b \geqslant 1$
- For all $y_i = -1$, $w \cdot x_i + b \leqslant -1$

SVM Constraints

Constraints:

- For all $y_i = 1$, $w \cdot x_i + b \geqslant 1$
- For all $y_i = -1$, $w \cdot x_i + b \leqslant -1$

SVM Constraints

Constraints:

- For all $y_i = 1$, $w \cdot x_i + b \geqslant 1$
- For all $y_i = -1$, $w \cdot x_i + b \leqslant -1$

If $y_i = 1$, multiply both sides of constraint by y_i:

- $y_i(w \cdot x_i + b) \geqslant 1(y_i) = 1$

SVM Constraints

Constraints:

- For all $y_i = 1$, $w \cdot x_i + b \geqslant 1$
- For all $y_i = -1$, $w \cdot x_i + b \leqslant -1$

If $y_i = 1$, multiply both sides of constraint by y_i:

- $y_i(w \cdot x_i + b) \geqslant 1(y_i) = 1$

If $y_i = -1$, multiply both sides of constraint by y_i:

- $y_i(w \cdot x_i + b) \geqslant -1(y_i) = 1$

SVM Constraints

The decision boundary is

$$w \cdot x + b = 0$$

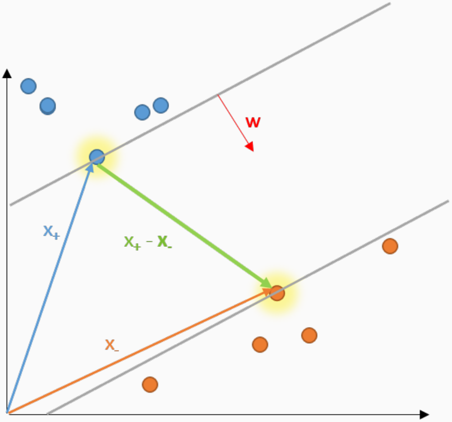
Margin is bounded by hyperplanes where

$$w \cdot x + b = 1 \quad \text{and} \quad w \cdot x + b = -1$$

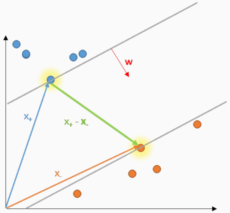
such that for all data points

$$y_i(w \cdot x_i + b) \geqslant 1$$

Width of Margin



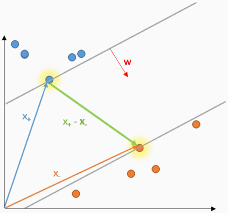
Width of Margin



- Width of margin is $\|proj_w(x_+ - x_-)\|$

SVM Constraints

Width of Margin



- Width of margin is $\|proj_w(x_+ - x_-)\|$
- width $= (x_+ - x_-) \cdot \frac{w}{\|w\|}$

SVM Constraints

Width of Margin

- For support vectors, $y_i(w \cdot x_i + b) - 1 = 0$

SVM Constraints

Width of Margin

- For support vectors, $y_i(w \cdot x_i + b) - 1 = 0$
- When $y_i = 1$, $w \cdot x_i = 1 - b$

SVM Constraints

Width of Margin

- For support vectors, $y_i(w \cdot x_i + b) - 1 = 0$
- When $y_i = 1$, $w \cdot x_i = 1 - b$
- When $y_i = -1$, $w \cdot x_i = -1 - b$

ML@B

Width of Margin

- For support vectors, $y_i(w \cdot x_i + b) - 1 = 0$
- When $y_i = 1$, $w \cdot x_i = 1 - b$
- When $y_i = -1$, $w \cdot x_i = -1 - b$
- width $= (x_+ - x_-) \cdot \frac{w}{\|w\|} = \frac{(x_+ - x_-) \cdot w}{\|w\|}$

SVM Constraints

Width of Margin

- For support vectors, $y_i(w \cdot x_i + b) - 1 = 0$
- When $y_i = 1$, $w \cdot x_i = 1 - b$
- When $y_i = -1$, $w \cdot x_i = -1 - b$
- width $= (x_+ - x_-) \cdot \frac{w}{\|w\|} = \frac{(x_+ - x_-) \cdot w}{\|w\|}$
- width $= \frac{1 - b - (-1 - b)}{\|w\|} = \frac{2}{\|w\|}$

To maximize the margin, minimize $\|w\|$

Hard Margin SVM

So now we have our problem formulation!

Problem:

$$\text{minimize } \frac{1}{2}\|w\|^2$$

such that

$$y_i(w \cdot x_i + b) \geqslant 1 \text{ for all points}$$

Hard Margin SVM

So now we have our problem formulation!

Problem:

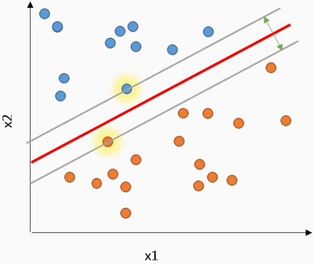
$$\text{minimize } \frac{1}{2}\|w\|^2$$

such that

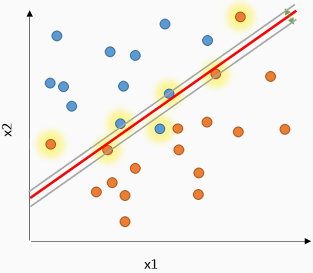
$$y_i(w \cdot x_i + b) \geqslant 1 \text{ for all points}$$

This is great! But let's take a step back. What's one key assumption we made before doing this derivation?

Support Vectors

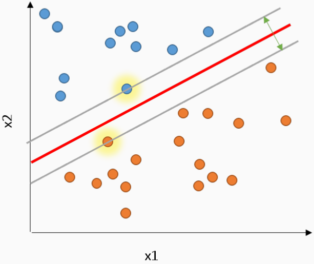


(a) Linearly Separable

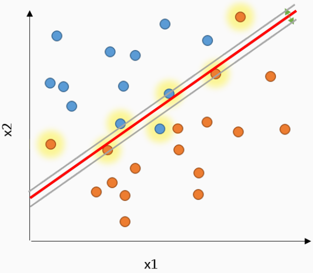


(b) Nonlinearly Separable

Support Vectors



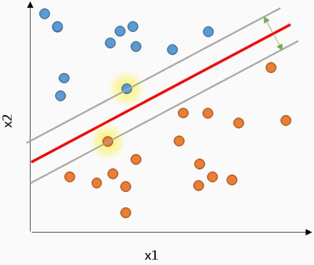
(a) Linearly Separable



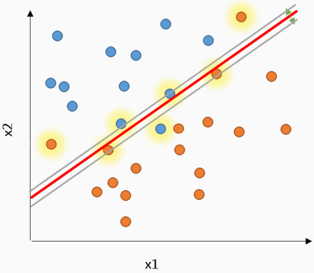
(b) Nonlinearly Separable

- Now they are data points that lie on margin OR violate margin

Support Vectors



(a) Linearly Separable



(b) Nonlinearly Separable

- Now they are data points that lie on margin OR violate margin
- Changing support vectors changes the decision boundary

Soft Margin SVM

We can no longer say

$$y_i(w \cdot x_i + b) \geqslant 1 \text{ for all points}$$

because this is impossible. No such w, b exist. This problem has no solution.

We can no longer say

$$y_i(w \cdot x_i + b) \geqslant 1 \text{ for all points}$$

because this is impossible. No such w, b exist. This problem has no solution.

How can we fix this?

Soft Margin SVM

We can no longer say

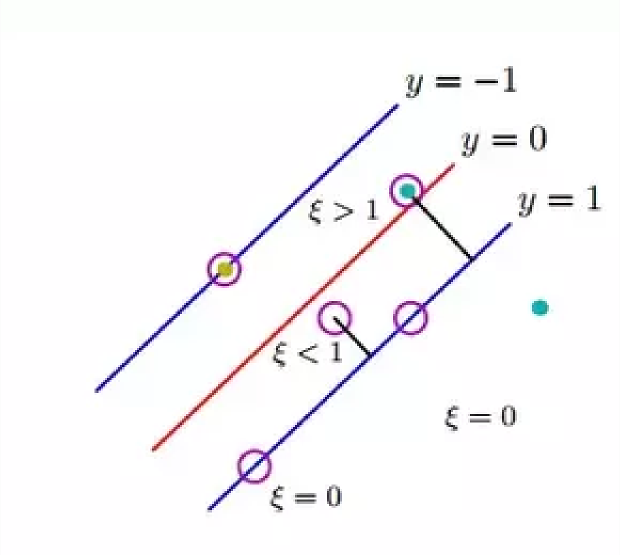
$$y_i(w \cdot x_i + b) \geqslant 1 \text{ for all points}$$

because this is impossible. No such w, b exist. This problem has no solution.

How can we fix this?

Let's add a "slack" to each datapoint, so each datapoint is free to cross the margin for as much as its slack is.

Soft Margin SVM



Soft Margin SVM

Constraints: $y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$ for all data points

Constraints: $y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$ for all data points

- Support vectors that violate the margin have $\xi_i > 0$, other points have $\xi_i = 0$

Soft Margin SVM

Constraints: $y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$ for all data points

- Support vectors that violate the margin have $\xi_i > 0$, other points have $\xi_i = 0$
- Support vectors on the margin have $y_i(w \cdot x_i + b) = 1$

Soft Margin SVM

Constraints: $y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$ for all data points

- Support vectors that violate the margin have $\xi_i > 0$, other points have $\xi_i = 0$
- Support vectors on the margin have $y_i(w \cdot x_i + b) = 1$
- Must penalize slack variables in some way

Soft Margin SVM

Optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^n \xi_i$$

such that

$$y_i(w \cdot x_i + b) \geqslant 1 - \xi_i \text{ and } \xi_i \geqslant 0$$

Soft Margin SVM

Optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n}\xi_i$$

such that

$$y_i(w \cdot x_i + b) \geqslant 1 - \xi_i \text{ and } \xi_i \geqslant 0$$

C is regularization hyperparameter (how much we penalize slack).

Soft Margin SVM

Optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n}\xi_i$$

such that

$$y_i(w \cdot x_i + b) \geqslant 1 - \xi_i \text{ and } \xi_i \geqslant 0$$

C is regularization hyperparameter (how much we penalize slack).
Since $\xi_i \geqslant 0$:

$$\xi_i = max(0, 1 - y_i(w \cdot x_i + b))$$

Soft Margin SVM

Optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n}\xi_i$$

such that

$$y_i(w \cdot x_i + b) \geqslant 1 - \xi_i \text{ and } \xi_i \geqslant 0$$

C is regularization hyperparameter (how much we penalize slack).
Since $\xi_i \geqslant 0$:

$$\xi_i = max(0, 1 - y_i(w \cdot x_i + b))$$

Which gives us a final objective of:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

Hinge loss

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

Hinge loss

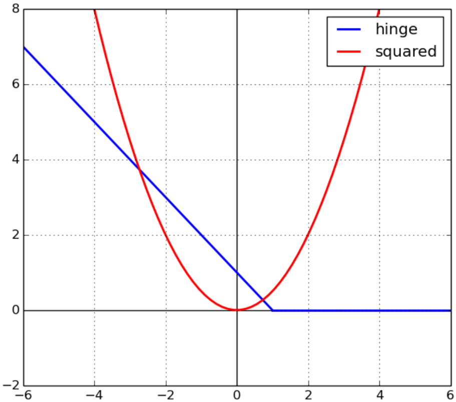
$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

First term is regularization. Second term is hinge loss.

Hinge loss

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

First term is regularization. Second term is hinge loss.



Solving the SVM

Solving SVM

We have

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

We can solve this via gradient descent.

Solving SVM

We have

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

We can solve this via gradient descent.

But this problem has a dual.

Solving SVM

We have

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

We can solve this via gradient descent.

But this problem has a dual.
From 170: every problem (called the primal) has a dual.
Maximizing the primal corresponds to minimizing the dual.
Optimums of both equal each other.

Solving SVM

We have

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

We can solve this via gradient descent.

But this problem has a dual.
From 170: every problem (called the primal) has a dual.
Maximizing the primal corresponds to minimizing the dual.
Optimums of both equal each other.

Primal: maximizing the margin.

Solving SVM

We have

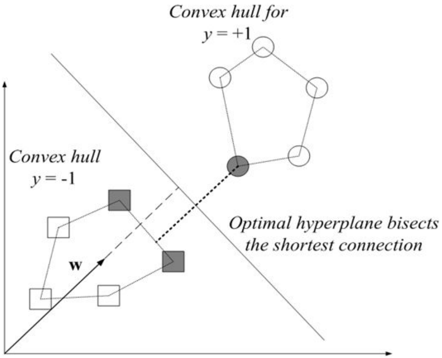
$$\text{minimize } \frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i + b))$$

We can solve this via gradient descent.

But this problem has a dual.
From 170: every problem (called the primal) has a dual.
Maximizing the primal corresponds to minimizing the dual.
Optimums of both equal each other.

Primal: maximizing the margin. Dual: minimizing the distance between convex hulls.

Convex Hull Interpretation of Dual



Find convex hulls for each class. The closest points to an optimal hyperplane are support vectors

30

Solution to SVM Optimization Using Lagrange Multipliers

We're going to do some derivations. Let's assume points are linearly separable for ease.

Solution to SVM Optimization Using Lagrange Multipliers

We're going to do some derivations. Let's assume points are linearly separable for ease.

Problem: minimize $\frac{1}{2}\|w\|^2$ such that $y_i(w \cdot x_i + b) - 1 = 0$ for support vectors

Take Derivative to Find Extremum

- $L = \frac{1}{2}\|w\|^2 - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$ where $\alpha_i = 0$ for non support vectors

Solution to SVM Optimization Using Lagrange Multipliers

We're going to do some derivations. Let's assume points are linearly separable for ease.

Problem: minimize $\frac{1}{2}\|w\|^2$ such that $y_i(w \cdot x_i + b) - 1 = 0$ for support vectors

Take Derivative to Find Extremum

- $L = \frac{1}{2}\|w\|^2 - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$ where $\alpha_i = 0$ for non support vectors
- $\frac{\partial L}{\partial w} = w - \Sigma\alpha_i y_i x_i = 0$, so $w = \Sigma\alpha_i y_i x_i$

We're going to do some derivations. Let's assume points are linearly separable for ease.

Problem: minimize $\frac{1}{2}\|w\|^2$ such that $y_i(w \cdot x_i + b) - 1 = 0$ for support vectors

Take Derivative to Find Extremum

- $L = \frac{1}{2}\|w\|^2 - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$ where $\alpha_i = 0$ for non support vectors
- $\frac{\partial L}{\partial w} = w - \Sigma\alpha_i y_i x_i = 0$, so $w = \Sigma\alpha_i y_i x_i$
- $\frac{\partial L}{\partial b} = -\Sigma\alpha_i y_i = 0$, so $\Sigma\alpha_i y_i = 0$

We know there is an extremum at $w = \Sigma \alpha_i y_i x_i$

Solution to SVM Optimization Using Lagrange Multipliers

We know there is an extremum at $w = \Sigma \alpha_i y_i x_i$

- $L = \frac{1}{2}\|w\|^2 - \Sigma \alpha_i [y_i(w \cdot x_i + b) - 1]$

Solution to SVM Optimization Using Lagrange Multipliers

We know there is an extremum at $w = \Sigma \alpha_i y_i x_i$

- $L = \frac{1}{2}\|w\|^2 - \Sigma \alpha_i [y_i(w \cdot x_i + b) - 1]$
- $L = \frac{1}{2}\Sigma \alpha_i y_i x_i \Sigma \alpha_j y_j x_j - \Sigma \alpha_i [y_i(w \cdot x_i + b) - 1]$

Solution to SVM Optimization Using Lagrange Multipliers

We know there is an extremum at $w = \Sigma\alpha_i y_i x_i$

- $L = \frac{1}{2}\|w\|^2 - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$
- $L = \frac{1}{2}\Sigma\alpha_i y_i x_i \Sigma\alpha_j y_j x_j - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$

Solution to SVM Optimization Using Lagrange Multipliers

We know there is an extremum at $w = \Sigma\alpha_i y_i x_i$

- $L = \frac{1}{2}\|w\|^2 - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$
- $L = \frac{1}{2}\Sigma\alpha_i y_i x_i \Sigma\alpha_j y_j x_j - \Sigma\alpha_i[y_i(w \cdot x_i + b) - 1]$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$

Solution to SVM Optimization Using Lagrange Multipliers

We know there is an extremum at $w = \Sigma \alpha_i y_i x_i$

- $L = \frac{1}{2}\|w\|^2 - \Sigma \alpha_i [y_i(w \cdot x_i + b) - 1]$
- $L = \frac{1}{2}\Sigma \alpha_i y_i x_i \Sigma \alpha_j y_j x_j - \Sigma \alpha_i [y_i(w \cdot x_i + b) - 1]$
- $L = \Sigma \alpha_i - \frac{1}{2}\Sigma\Sigma \alpha_i \alpha_j y_i y_j x_i \cdot x_j$

$$L = \Sigma \alpha_i - \frac{1}{2}\Sigma\Sigma \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

Key: optimization problem depends on **dot product of data points**

Kernels

Kernel Motivation

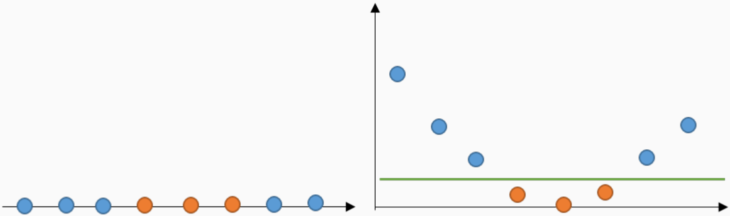
Many times our data is highly nonlinear. In this case, slack variables won't do much (slack variables are really only effective for outliers).

Kernel Motivation

Many times our data is highly nonlinear. In this case, slack variables won't do much (slack variables are really only effective for outliers).

Key idea: transform data into a higher-dimensional space

Nonlinearly separable data can be linearly separable in higher dimension



- Let $\Phi(x)$ be the transformation to a higher space

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$
- After transforming data points with $\Phi(x)$, optimization problem depends on $\Phi(x_i) \cdot \Phi(x_j)$

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$
- After transforming data points with $\Phi(x)$, optimization problem depends on $\Phi(x_i) \cdot \Phi(x_j)$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$
- After transforming data points with $\Phi(x)$, optimization problem depends on $\Phi(x_i) \cdot \Phi(x_j)$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$
- $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$
- After transforming data points with $\Phi(x)$, optimization problem depends on $\Phi(x_i) \cdot \Phi(x_j)$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$
- $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$
- Kernels allow us to compute $\Phi(x_i) \cdot \Phi(x_j)$ without computing $\Phi(x_i)$

Kernels with SVMs

- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i \cdot x_j$, so optimization problem depends on $x_i \cdot x_j$
- After transforming data points with $\Phi(x)$, optimization problem depends on $\Phi(x_i) \cdot \Phi(x_j)$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$
- $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$
- Kernels allow us to compute $\Phi(x_i) \cdot \Phi(x_j)$ without computing $\Phi(x_i)$
- $L = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(x_i, x_j)$

Example: Polynomial Kernel

- Let $x =< x_1, x_2 >$, $y =< y_1, y_2 >, p = 2$

Example: Polynomial Kernel

- Let $x =< x_1, x_2 >$, $y =< y_1, y_2 >, p = 2$
- $\Phi(x) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2 >$

Example: Polynomial Kernel

- Let $x =< x_1, x_2 >$, $y =< y_1, y_2 >, p = 2$
- $\Phi(x) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2 >$
- $\Phi(y) =< 1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2 >$

Example: Polynomial Kernel

- Let $x =< x_1, x_2 >$, $y =< y_1, y_2 >, p = 2$
- $\Phi(x) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2 >$
- $\Phi(y) =< 1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2 >$
- $\Phi(x) \cdot \Phi(y) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2 > \cdot <$ $1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2 >=$ $1 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 + 2x_1y_1x_2y_2 =$ $(1 + x_1y_1 + x_2y_2)^2 = (1 + x \cdot y)^2 = K(x, y)$

Example: Polynomial Kernel

- Let $x =< x_1, x_2 >$, $y =< y_1, y_2 >, p = 2$
- $\Phi(x) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2 >$
- $\Phi(y) =< 1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2 >$
- $\Phi(x) \cdot \Phi(y) =< 1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2 > \cdot <$
  $1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2 >=$
  $1 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 + 2x_1y_1x_2y_2 =$
  $(1 + x_1y_1 + x_2y_2)^2 = (1 + x \cdot y)^2 = K(x, y)$
- Polynomial Kernel: $K(x, y) = (1 + x \cdot y)^p$

Example: Polynomial Kernel

- Computing degree p features of d dimensional input takes $O(d^p)$ time

Example: Polynomial Kernel

- Computing degree p features of d dimensional input takes $O(d^p)$ time
- Using Polynomial Kernel, $\Phi(x_i) \cdot \Phi(x_j)$ can be computed in $O(d)$ time, even if $\Phi(x_i) \cdot \Phi(x_j)$ has length $O(d^p)$

Example: Polynomial Kernel

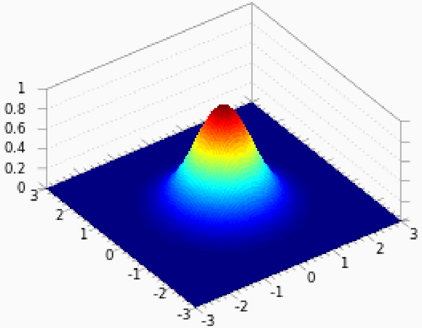
- Computing degree p features of d dimensional input takes $O(d^p)$ time
- Using Polynomial Kernel, $\Phi(x_i) \cdot \Phi(x_j)$ can be computed in $O(d)$ time, even if $\Phi(x_i) \cdot \Phi(x_j)$ has length $O(d^p)$
- Magic! $O(d^p)$ operation became $O(d)$. Exponential became linearithmic (can be shown with some work).

Polynomial Kernel Demo

https://www.youtube.com/watch?v=3liCbRZPrZA

Recall the gaussian: $\exp -\frac{(x-\mu)^2}{2\sigma^2}$

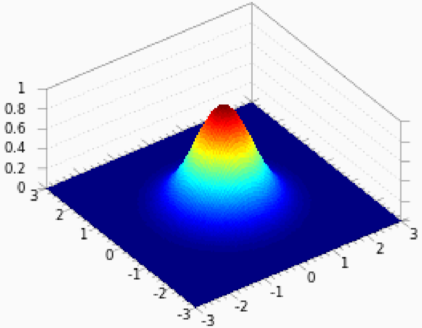
Essentially measures how close you are to the mean



Kernels with SVMs

Recall the gaussian: $\exp -\frac{(x-\mu)^2}{2\sigma^2}$

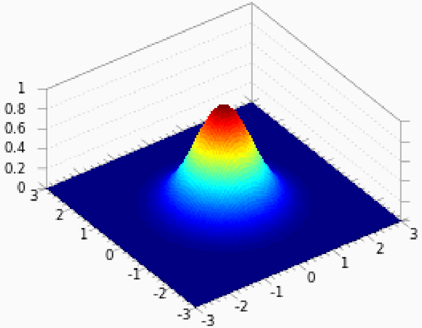
Essentially measures how close you are to the mean



- What is $\Phi(x)$ in this case?

Recall the gaussian: $\exp -\frac{(x-\mu)^2}{2\sigma^2}$

Essentially measures how close you are to the mean

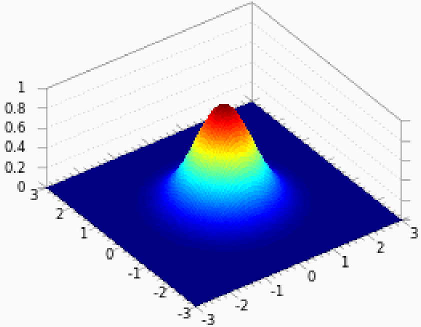


- What is $\Phi(x)$ in this case?
- We'd have to evaluate the gaussian at *infinitely many* points along the curve.

Kernels with SVMs

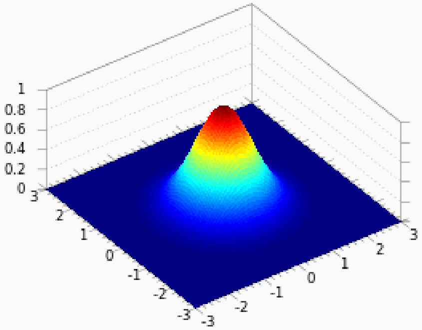
Recall the gaussian: $\exp -\frac{(x-\mu)^2}{2\sigma^2}$

Essentially measures how close you are to the mean



Recall the gaussian: $\exp -\frac{(x-\mu)^2}{2\sigma^2}$

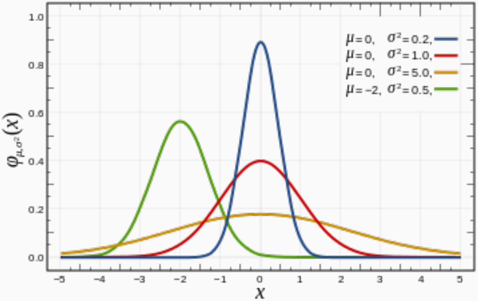
Essentially measures how close you are to the mean



- But $K(x, y)$ is just $\exp(-\frac{|x-y|^2}{2\sigma^2})$! Single computation

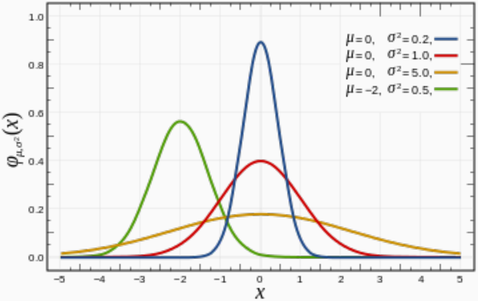
Example: Gaussian Radial Basis Kernel

- $K(x, y) = \exp(-\frac{|x-y|^2}{2\sigma^2}) = \exp(-\gamma|x-y|^2), \gamma > 0$



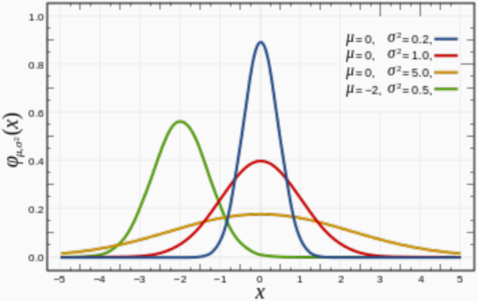
Example: Gaussian Radial Basis Kernel

- $K(x, y) = \exp(-\frac{|x-y|^2}{2\sigma^2}) = \exp(-\gamma|x-y|^2), \gamma > 0$
- $\Phi(x)$ is infinite vector



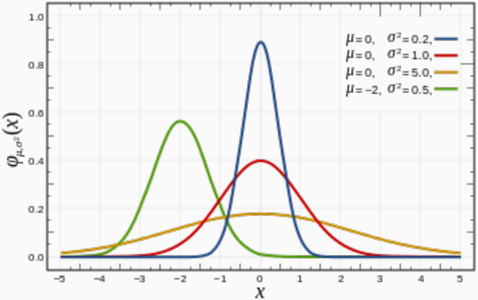
Example: Gaussian Radial Basis Kernel

- $K(x, y) = \exp(-\frac{|x-y|^2}{2\sigma^2}) = \exp(-\gamma|x-y|^2), \gamma > 0$
- $\Phi(x)$ is infinite vector
- $\Phi(x) \cdot \Phi(y)$ converges to $K(x, y)$

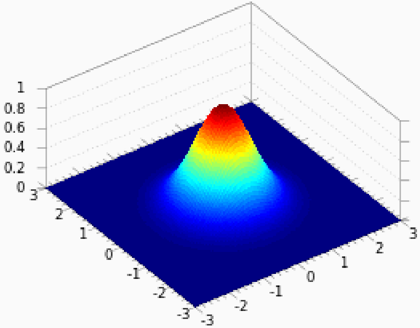


Example: Gaussian Radial Basis Kernel

- $K(x, y) = \exp(-\frac{|x-y|^2}{2\sigma^2}) = \exp(-\gamma|x-y|^2), \gamma > 0$
- $\Phi(x)$ is infinite vector
- $\Phi(x) \cdot \Phi(y)$ converges to $K(x, y)$
- Large γ = small σ, which makes Gaussian narrower $\Rightarrow$ causes high variance, lower bias



Example: Gaussian Kernel as Similarity Function



- $K(x, y)$ assigns high value for points that are near each other

Real Power of Kernels

Let's pause for a second. What did we just show?

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**
- We can further show (through a bit of real analysis):

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**
- We can further show (through a bit of real analysis):
  - The dot product of *any* infinite-dimensional representation can be represented (very efficiently) using Kernels

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**
- We can further show (through a bit of real analysis):
  - The dot product of *any* infinite-dimensional representation can be represented (very efficiently) using Kernels
  - Any loss function operating on an infinite-dimensional space can be equivalently reduced to a loss operated on kernalized dot products

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**
- We can further show (through a bit of real analysis):
  - The dot product of *any* infinite-dimensional representation can be represented (very efficiently) using Kernels
  - Any loss function operating on an infinite-dimensional space can be equivalently reduced to a loss operated on kernalized dot products

Real Power of Kernels

Let's pause for a second. What did we just show?

- Kernels can essentially compute dot products between **infinite vectors**
- We can further show (through a bit of real analysis):
  - The dot product of *any* infinite-dimensional representation can be represented (very efficiently) using Kernels
  - Any loss function operating on an infinite-dimensional space can be equivalently reduced to a loss operated on kernalized dot products

Conclusion: kernels have *limitless* expressive power.

Why Neural Nets?

Conclusion: kernels have *limitless* expressive power.

Why Neural Nets?

Conclusion: kernels have *limitless* expressive power.

Why do we use neural nets instead of kernels?

Why Neural Nets?

Conclusion: kernels have *limitless* expressive power.

Why do we use neural nets instead of kernels?

Kernels grow on the order of the datapoints, $O(N)$, while neural networks grow on the order of the paramters $O(p)$.

N $>>$ p

Why Use Gaussian Kernels?

- Gives a smooth decision function
- Behaves like smoother k-nearest-neighbors
- Oscillates less than polynomial kernels, depending on value of σ
- Sample points closer to z have greater impact on prediction of z

SVMS in Practice

Why Use SVMs?

- SVM picks optimal hyperplane which allows model to generalize well
- Not sensitive to outliers
- Kernel functions allow for efficient computation of nonlinear features

Overview of SVMs

Pros:

- Finds optimal decision boundary between data
- Can capture complex nonlinear relationship between data with more efficiency than manually calculating features, while still maintaining simplicity of model

Cons:

- Calculating many higher dimensional features still takes long time, especially if input size is large
- Data transformation and boundary after kernel trick is hard to interpret $\Rightarrow$ SVMs are often treated like black box

SVMs in Practice

- Challenging to implement from scratch efficiently
- Better use of time to know how to use an SVM well rather than know how to code an SVM from scratch

Hyperparameter C

Soft Margin SVM: minimize $\frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n}\xi_i$ such that $y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$ and $\xi_i \geqslant 0$

- C represents how unacceptable it is to misclassify **training** data points

Hyperparameter C

Large C:

- Similar to hard margin SVM - goal is to misclassify few training points
- Often results in small margins
- Very sensitive to outliers
- Risk of overfitting

Small C:

- Maximizes margin at cost of misclassifying training data points
- Risk of underfitting

Hyperparameter γ

- γ applies for polynomial, RBF, and sigmoid kernels in sklearn

Small γ:

- Larger variance in Gaussian RBF kernel, so each support vector has a greater influence on class of points far away from it
- Leads to high bias, low variance models with risk of underfitting

Large: γ

- Leads to high variance, low bias models with risk of overfitting

SVM Demo

Debugging ML Algorithms

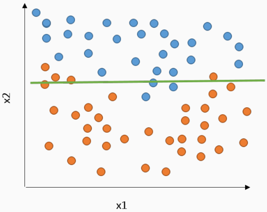
Common Issue 1 - Bias vs Variance Tradeoff

- Problem: Classifier has a test error that is too high

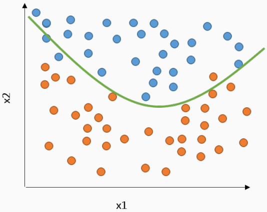
Common Issue 1 - Bias vs Variance Tradeoff

- Problem: Classifier has a test error that is too high
- Solution: Check if classifier is overfitting or underfitting

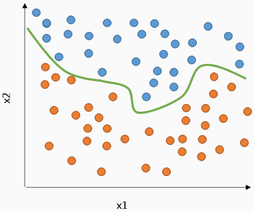
Common Issue 1 - Bias vs Variance Tradeoff



(a) High Bias

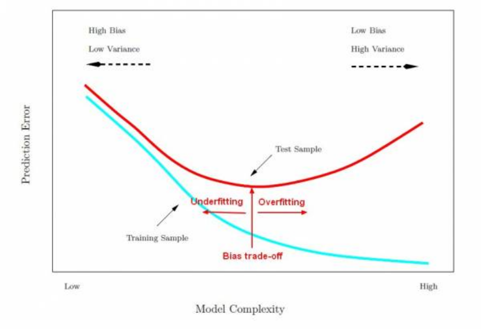


(b) Balanced

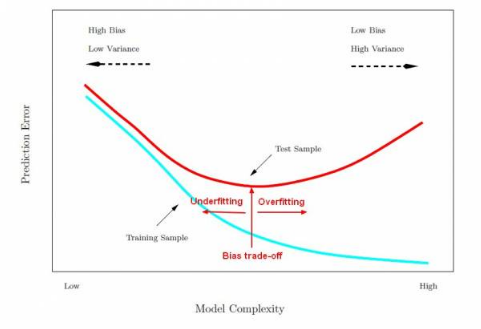


(c) High Variance

Common Issue 1 - Bias vs Variance Tradeoff

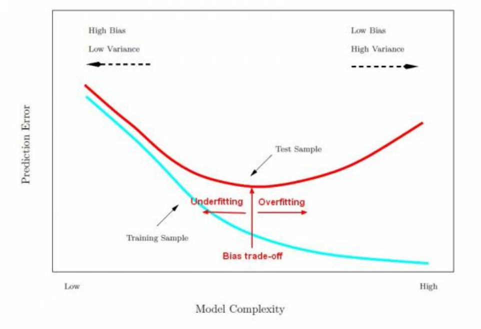


Common Issue 1 - Bias vs Variance Tradeoff



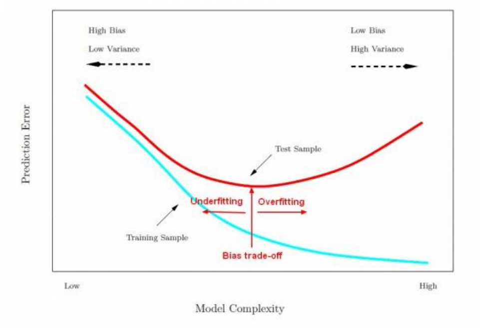
- Training error always less than test error

Common Issue 1 - Bias vs Variance Tradeoff



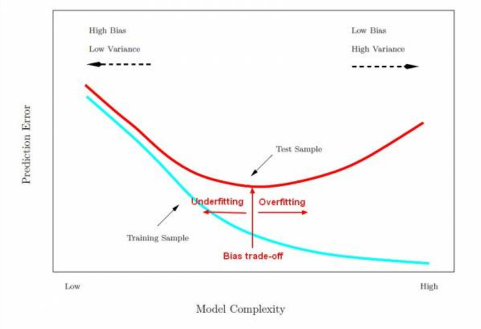
- Training error always less than test error
- Increasing model complexity reduces bias but increases variance

Common Issue 1 - Bias vs Variance Tradeoff



- Training error always less than test error
- Increasing model complexity reduces bias but increases variance
- Training error only represents bias; test error represents both

Common Issue 1 - Bias vs Variance Tradeoff



- Training error always less than test error
- Increasing model complexity reduces bias but increases variance
- Training error only represents bias; test error represents both
- There is some minimum point to the tradeoff; we use cross-validation to find this

Common Issue 1 - Bias vs Variance Tradeoff

Mini Quiz: Which of the following will fix high bias? Which will fix high variance?

- Obtain more training examples
- Reduce number of features
- Increase number of features
- Use regularization for linear or logistic regression

Common Issue 1 - Bias vs Variance Tradeoff

Mini Quiz: Which of the following will fix high bias? Which will fix high variance?

- Obtain more training examples - High Variance
- Reduce number of features
- Increase number of features
- Use regularization for linear or logistic regression

Mini Quiz: Which of the following will fix high bias? Which will fix high variance?

- Obtain more training examples - High Variance
- Reduce number of features - High Variance
- Increase number of features
- Use regularization for linear or logistic regression

Mini Quiz: Which of the following will fix high bias? Which will fix high variance?

- Obtain more training examples - High Variance
- Reduce number of features - High Variance
- Increase number of features - High Bias
- Use regularization for linear or logistic regression

Mini Quiz: Which of the following will fix high bias? Which will fix high variance?

- Obtain more training examples - High Variance
- Reduce number of features - High Variance
- Increase number of features - High Bias
- Use regularization for linear or logistic regression - High Variance

Common Issue 2 - Issues with Optimization Objective or Optimization Algorithm

Mini Quiz: Which of the following will fix a bad optimization objective? Which will fix a bad optimization algorithm?

- Run more iterations of gradient descent
- Try using Newton's method to optimize the function
- Use different values for hyperparameters
- Use a more expressive model, such as a neural network

Common Issue 2 - Issues with Optimization Objective or Optimization Algorithm

Mini Quiz: Which of the following will fix a bad optimization objective? Which will fix a bad optimization algorithm?

- Run more iterations of gradient descent - Optimization Algorithm
- Try using Newton's method to optimize the function
- Use different values for hyperparameters
- Use a more expressive model, such as a neural network

Common Issue 2 - Issues with Optimization Objective or Optimization Algorithm

Mini Quiz: Which of the following will fix a bad optimization objective? Which will fix a bad optimization algorithm?

- Run more iterations of gradient descent - Optimization Algorithm
- Try using Newton's method to optimize the function - Optimization Algorithm
- Use different values for hyperparameters
- Use a more expressive model, such as a neural network

Common Issue 2 - Issues with Optimization Objective or Optimization Algorithm

Mini Quiz: Which of the following will fix a bad optimization objective? Which will fix a bad optimization algorithm?

- Run more iterations of gradient descent - Optimization Algorithm
- Try using Newton's method to optimize the function - Optimization Algorithm
- Use different values for hyperparameters - Optimization Objective
- Use a more expressive model, such as a neural network

Common Issue 2 - Issues with Optimization Objective or Optimization Algorithm

Mini Quiz: Which of the following will fix a bad optimization objective? Which will fix a bad optimization algorithm?

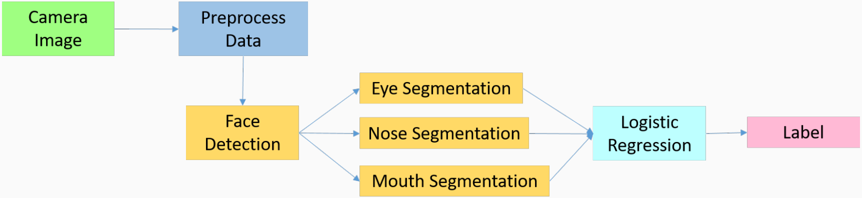
- Run more iterations of gradient descent - Optimization Algorithm
- Try using Newton's method to optimize the function - Optimization Algorithm
- Use different values for hyperparameters - Optimization Objective
- Use a more expressive model, such as a neural network - Optimization Objective

Applying ML Algorithms

When using an ML algorithm:

1. Design various components of algorithm architecture
   - Benefit: Allows for more scalable algorithm
   - Issue: Hard to predict design for each component and understand what hardest components are
2. Try to come up with a quick implementation and then optimize
   - Benefit: Often application will work more quickly - time is spent only on components that are broken

Error Analysis Example: Face Recognition



Error Analysis Example: Face Recognition

- Plug in true values as input to each component and see how each component affects accuracy

Component	Accuracy
Overall System	85%
Preprocess Data	85.1%
Eye Segmentation	95%
Nose Segmentation	96%
Mouth Segmentation	97%
Logistic Regression	100%

- Most room for improvement in eye segmentation

Questions

Questions?