Linear Models

Héctor Corrada Bravo

University of Maryland, College Park, USA

CMSC643: 2018-09-18

Support Vector Machines

State-of-the-art classification and regression method

Flexible and efficient framework to learn classifers.

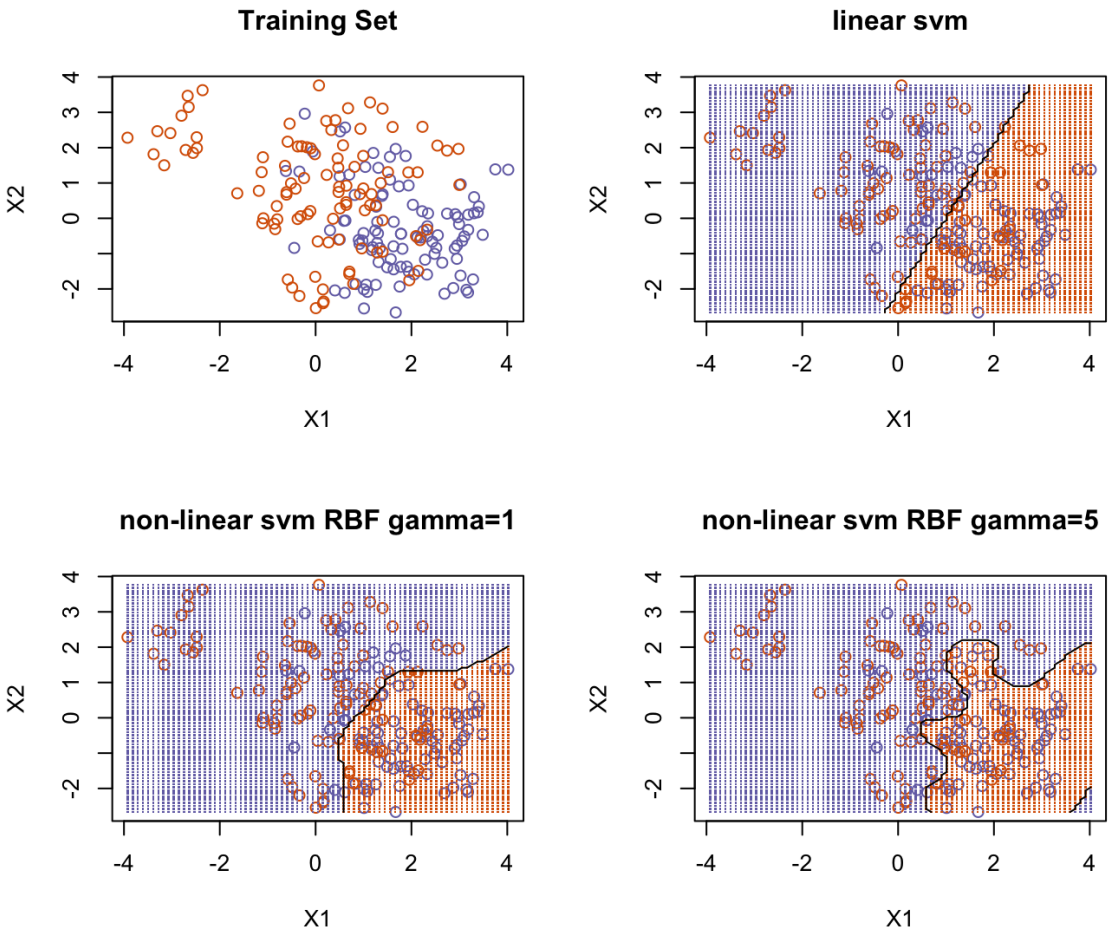
Support Vector Machines

State-of-the-art classification and regression method

Flexible and efficient framework to learn classifers.

Build upon linear methods we have discussed previously and have a nice geometric interpretation of how they are trained (based maximum margin arguments).

Support Vector Machines



SVMs follow the "predictor space partition" framework

Separating Hyperplanes

Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i$ is a vector of p predictor values for ith observation,
- y_i is the class label (we're going to use +1 and -1)

Separating Hyperplanes

Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i$ is a vector of p predictor values for ith observation,
- y_i is the class label (we're going to use +1 and -1)

Build a classifier by defining a *discriminative* function such that

$$b + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_p x_{ip} > 0 \,\text{if} y_i = +1$$

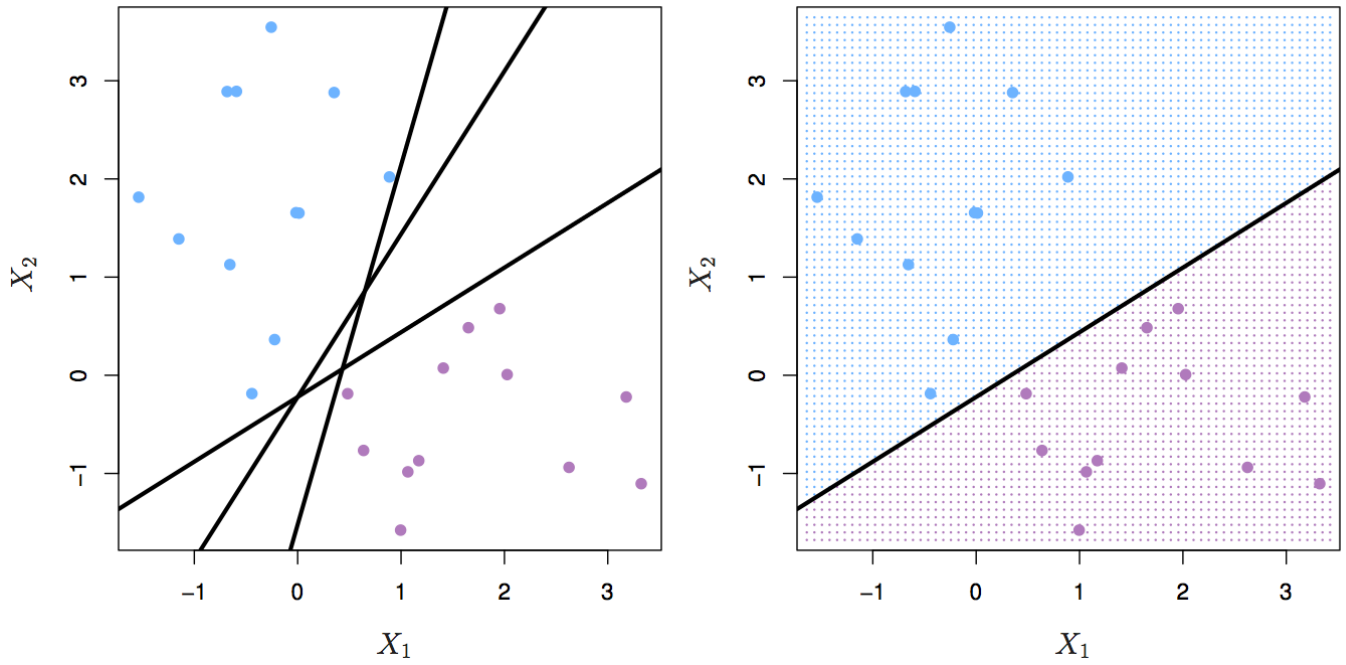
and

$$b + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_p x_{ip} < 0 \,\text{if} y_i = -1$$

Separating Hyperplanes

Points where the *discriminative* function equals 0 form a *hyper-plane* (i.e., a line in 2D)

$$\{x : b + w_1 x_1 + \cdots + w_p x_p = 0\}$$



Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote $\mathbf{w}$ as the vector $(w_1, w_2, \ldots, w_p)$

Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote $\mathbf{w}$ as the vector $(w_1, w_2, \ldots, w_p)$

Restrict estimates to those for which $\mathbf{w}'\mathbf{w} = ||\mathbf{w}||^2 = 1$

Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote $\mathbf{w}$ as the vector $(w_1, w_2, \ldots, w_p)$

Restrict estimates to those for which $\mathbf{w}'\mathbf{w} = \|\mathbf{w}\|^2 = 1$

Then, the *signed* distance of any point x to the decision boundary L is $b + \mathbf{w}'x$.

Separating Hyperplanes

With this we can easily describe the two partitions as

$$L^+ = \{x : b + \mathbf{w}'x > 0\},$$

$$L^- = \{x : b + \mathbf{w}'x < 0\}$$

Separating Hyperplanes

The $\mathbf{w}$ we want as an estimate is one that separates the training data as perfectly as possible.

Separating Hyperplanes

The $\mathbf{w}$ we want as an estimate is one that separates the training data as perfectly as possible.

Describe this requirement as

$$y_i(b + \mathbf{w}'x_i) > 0, i = 1, \ldots, N$$

Perceptron Algorithm

Algorithm to find vector $\mathbf{w}$ that satisfies the separation requirement as much as possible.

Penalize $\mathbf{w}$ by how far into the wrong side misclassified points are:

$$D(b, \mathbf{w}) = -\sum_{i \in \mathcal{M}} y_i (b + \mathbf{w}' x_i)$$

$\mathcal{M}$: set of points misclassified by $\mathbf{w}$ (on the wrong side of the hyper-plane).

Perceptron Algorithm

Estimate $\mathbf{w}$ by minimizing D.

How do we do this??

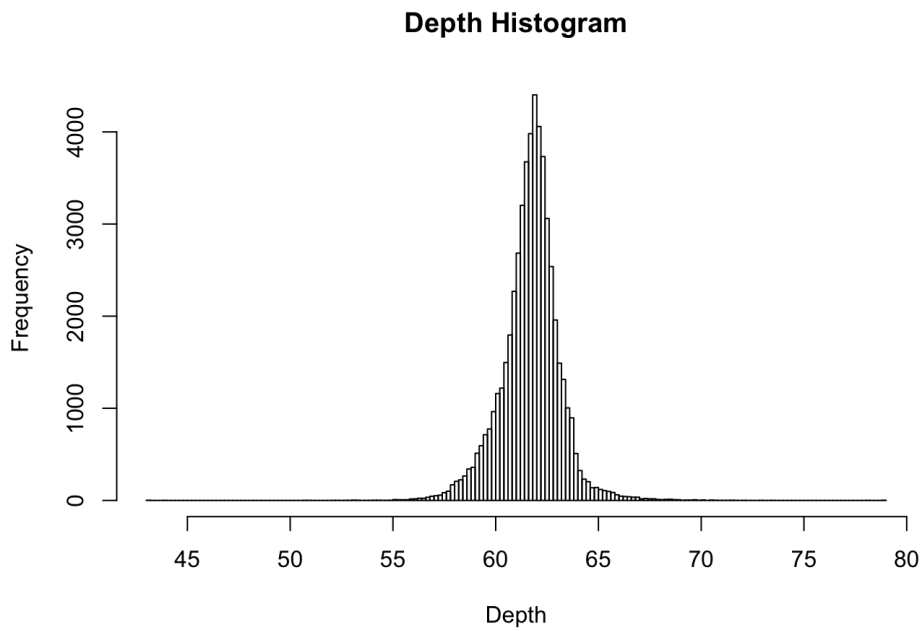
Tangent: Why do we use means so much in Data Science?

Suppose I get some dataset for analysis, the first thing I do is to use Exploratory Data Analysis to get a sense of what this data looks like.

One purpose of EDA is to spot problems in data (as part of data wrangling) and understand variable properties like:

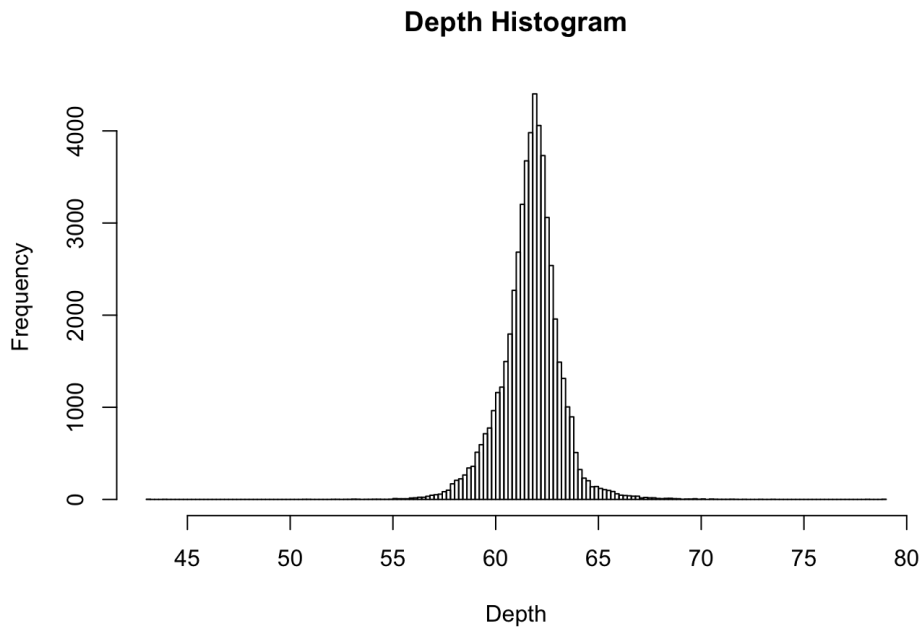
- central trends
- spread
- skew
- suggest possible modeling strategies (e.g., probability distributions)

Tangent: Why do we use means so much in Data Science?



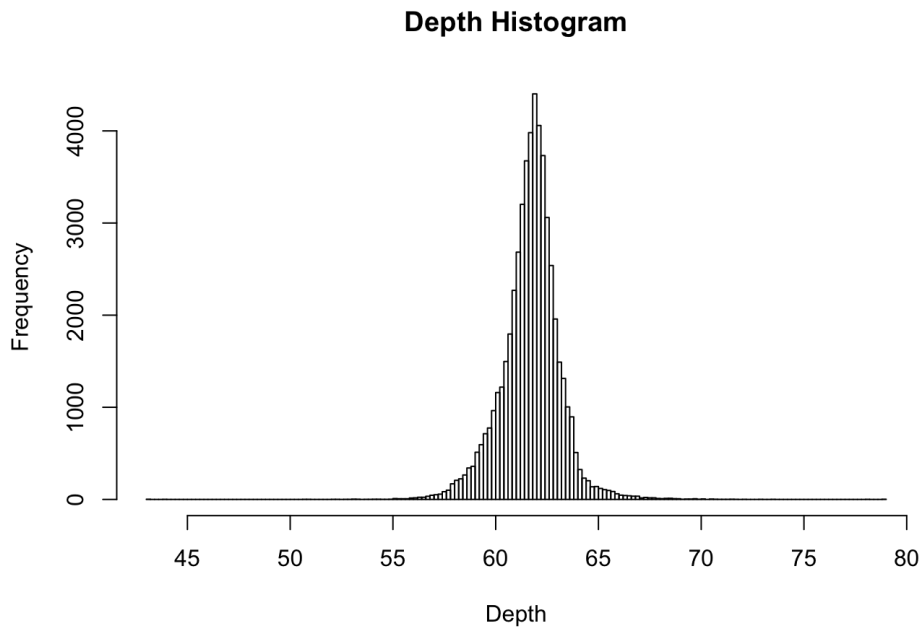
This is a dataset about diamond characteristics, one of which is a diamonds *depth* which we plot the distribution of here.

Tangent: Why do we use means so much in Data Science?



An obvious question to ask is *what is the central tendency of depth* in this dataset?

Tangent: Why do we use means so much in Data Science?



The best known statistic for central tendency is the *mean* of the data:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Tangent: Why do we use means so much in Data Science?

Turns out we can be a bit more formal about why the mean of the data makes sense as an estimate of central tendency

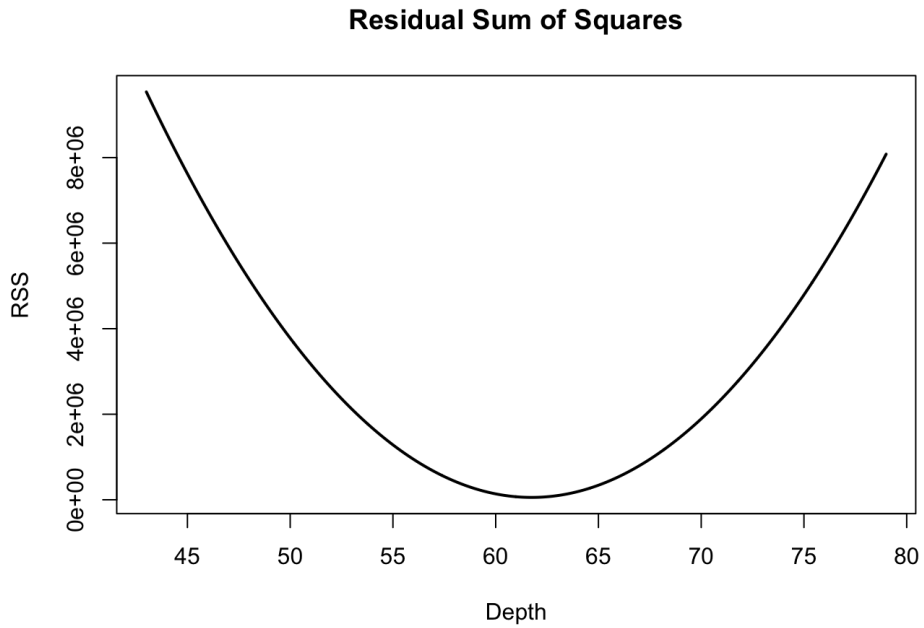
Tangent: Why do we use means so much in Data Science?

Turns out we can be a bit more formal about why the mean of the data makes sense as an estimate of central tendency

Define cost function RSS of some parameter μ as

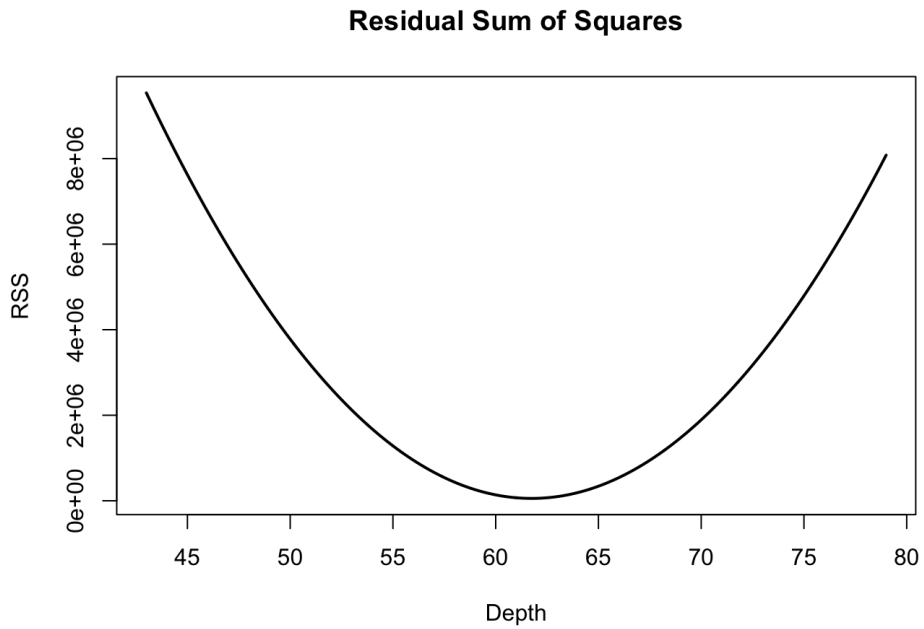
$$RSS(\mu) = \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^2$$

Tangent: Why do we use means so much in Data Science?



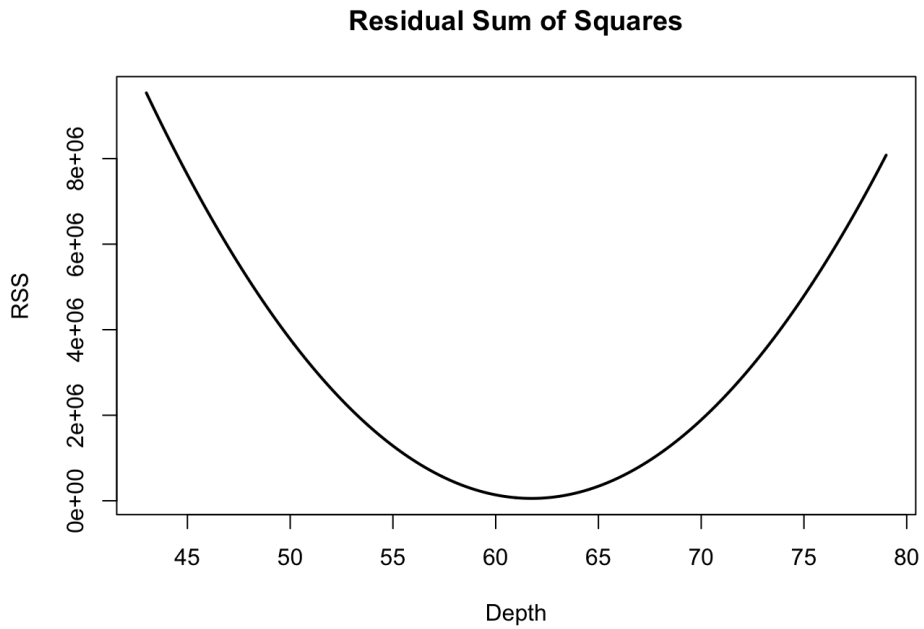
We can plot RSS for different values of μ

Tangent: Why do we use means so much in Data Science?



We should use the μ that minimizes RSS as our estimate of central tendency. Why?

Tangent: Why do we use means so much in Data Science?



Claim Setting $\mu = \overline{x}$ minimizes $RSS(\mu)$ over all values of μ.

Quiz Prove this claim.

Math Review: Gradients

Gradients are a generalization of the derivative for vectors

Their role in optimization is similar to what we saw in the proof to the previous claim

For example: if we want to find the minimum of a function, find a point where gradient is 0

Math Review: Gradients

Suppose function $f : \mathbb{R}^p \to \mathbb{R}$

Takes vector $x = \langle x_1, x_2, \ldots, x_p \rangle$ as input

The gradient of f is the vector where jth entry is the derivative of f with respect to x_j.

$$\nabla_x f = \left\langle \frac{\partial f}{x_1}, \ldots \frac{\partial f}{x_p} \right\rangle$$

Math Review: Gradients

Consider the following function

$$f(\mu) = \frac{1}{2} \sum_{i=1}^{N} \|x_i - \mu\|^2$$

Quiz: Write the gradient $\nabla_\mu f$

Math Review: Gradients

For now we will use two important properties of the gradient:

- $\nabla_x f(x^*) = \mathbf{0}$ is a necessary (not sufficient) condition for vector x^* to be an optimizer of f
- $\nabla_x f(x^*)$ gives the direction of **steepest ascent** of function f at point x^*

Optimization with gradients

Let's use these properties to concoct an optimization method: *gradient descent*

Optimization with gradients

Let's use these properties to concoct an optimization method: *gradient descent*

Given function f to *minimize*

- Start at some arbitrary vector z
- Repeat until done:
  - Find the direction of *steepest descent*: $-\nabla_x f(z)$
  - Update z by moving in that direction

Optimization with gradients

function GRADIENTDESCENT($f, K, \eta_1, \ldots$)
  $z^{(0)} \leftarrow \langle 0, \ldots, 0 \rangle$
  for all $k = 1, \ldots, K$ **do**
    $g^{(k)} \leftarrow \nabla_z f(z^{(k-1)})$
    $z^{(k)} \leftarrow z^{(k-1)} - \eta_k g^{(k)}$
  end for
  return $z^{(K)}$
end function

Optimization with gradients

Quiz Sketch gradient descent algorithm to minimize function

$$f(\mu) = \frac{1}{2} \sum_{i=1}^{N} \|x_i - \mu\|^2$$

Optimization with gradients

Consider the linear regression (least squares problem): Given dataset $\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle$, where outcomes y_i are *continuous*.

Suppose we want to model outcome Y as a linear function of X:

$$Y = b + \mathbf{w}' x$$

Optimization with gradients

Consider the linear regression (least squares problem): Given dataset $\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle$, where outcomes y_i are *continuous*.

Suppose we want to model outcome Y as a linear function of X:

$$Y = b + \mathbf{w}' x$$

Quiz Sketch gradient descent algorithm to minimize squared error loss

$$L(b, \mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \left[y_i - (b + \mathbf{w}' x_i) \right]^2$$

Perceptron Algorithm

Algorithm to find vector $\mathbf{w}$ that satisfies the separation requirement as much as possible.

Penalize $\mathbf{w}$ by how far into the wrong side misclassified points are:

$$D(b, \mathbf{w}) = -\sum_{i \in \mathcal{M}} y_i (b + \mathbf{w}' x_i)$$

$\mathcal{M}$: set of points misclassified by $\mathbf{w}$ (on the wrong side of the hyper-plane).

Perceptron Algorithm

Assuming $\mathcal{M}$ is fixed, the gradient of D is

$$\nabla_{\mathbf{w}} D = -\sum_{i \in \mathcal{M}} y_i x_i$$

and

$$\nabla_b D = -\sum_{i \in \mathcal{M}} y_i$$

Perceptron Algorithm

Perceptron algorithm uses *stochastic gradient descent*:

- Initialize parameters b and $\mathbf{w}$
- Cycle through training points i, if it is misclassified, update parameters as

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i x_i$$

and

$$b \leftarrow b + \eta y_i$$

- Stop when converged (or get tired of waiting)

This is gradient descent!

Perceptron Algorithm

There are a few problems with this algorithm:

If there exists b and $\mathbf{w}$ that separates the training points perfectly,

Perceptron Algorithm

There are a few problems with this algorithm:

If there exists b and $\mathbf{w}$ that separates the training points perfectly,

then there are an infinite number of b and $\mathbf{w}$s that also separate the data perfectly

Perceptron Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

Perceptron Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

However, the number of finite steps can be *very* large

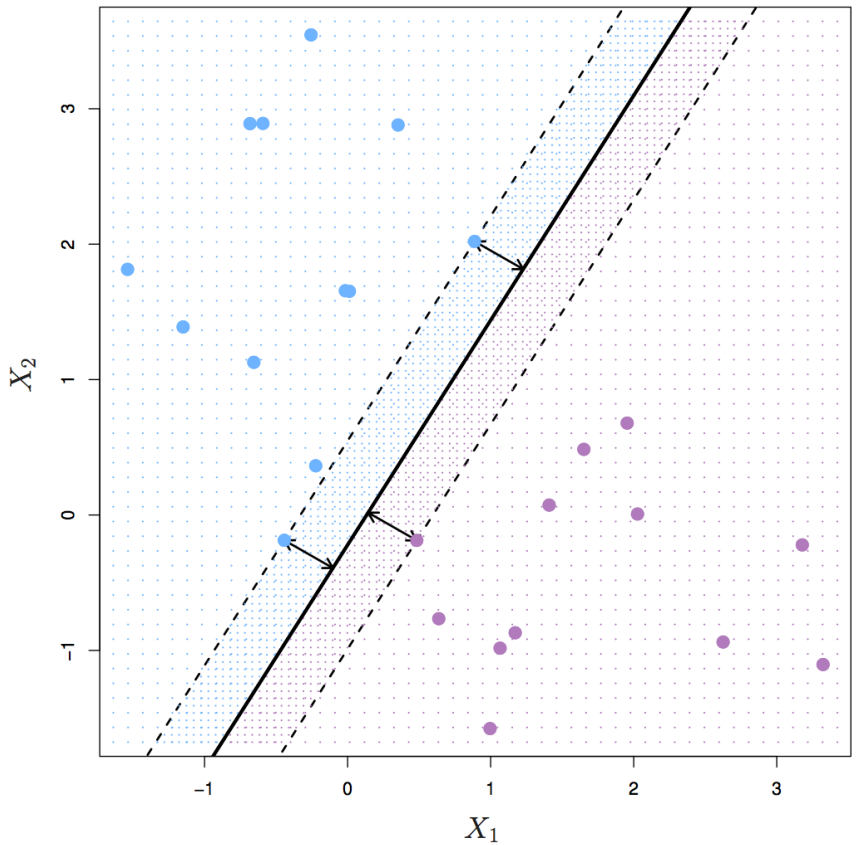
Perceptron Algorithm

When the training data is *not* separable, the algorithm will not converge.

Support Vector Machines

Support Vector Machines (SVMs) are designed to directly address these problems.

Support Vector Machines



A central concept in SVMs that we did not see in logistic regression is **the margin**: the distance between the separating plane and its nearest datapoints.

Support Vector Machines

When the data are separable, SVMs will choose the single optimal $\mathbf{w}$ that *maximizes* the distance between the decision boundary and the closest point in each class.

Support Vector Machines

When the data are separable, SVMs will choose the single optimal $\mathbf{w}$ that *maximizes* the distance between the decision boundary and the closest point in each class.

Why is this a good idea?

Support Vector Machines

Maximum margin hyper-planes

Goal: find the hyper-plane that separates training data with largest margin.

This will tend to *generalize* better since new observations have room to fall within margin and still be classified correctly.

Maximum margin hyper-planes

This can be cast as *optimization* problem:

$$\max_{b,\mathbf{w}} M$$
$$\text{s. t.} |\mathbf{w}|^2 = 1$$
$$y_i(b + \mathbf{w}' x_i) \geq M \, \forall i$$

Maximum margin hyper-planes

Rewrite optimization problem setting $M = 1/\|\mathbf{w}\|^2$ and using a little bit of algebra (see CIML):

$$\min_{b,\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$
$$\text{s.t.} y_i(b + \mathbf{w}' x_i) \geq 1 \, \forall i$$

Maximum margin hyper-planes

$$\min_{b,\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$
$$\text{s.t.} y_i(b + \mathbf{w}' x_i) \geq 1 \, \forall i$$

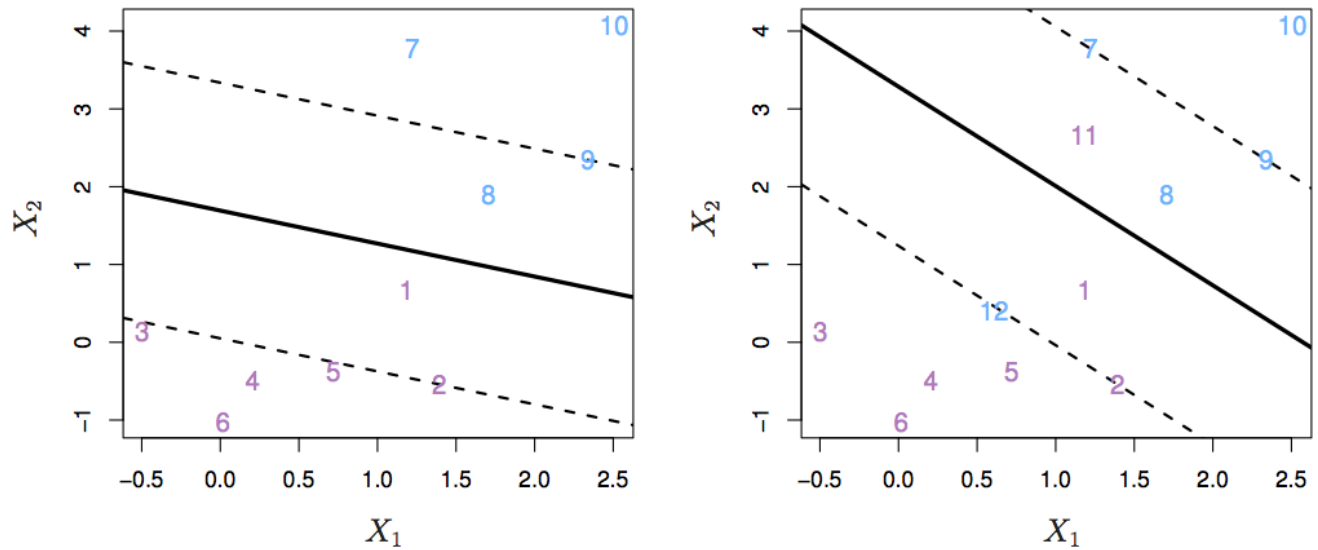
This is a *constrained* optimization problem

Minimize the norm of $\mathbf{w}$ under the constraint that it classifies every observation correctly.

Non-separable data

The method we have discussed so far runs into an important complication:

What if there is no separating hyper-plane?.



Non-separable data

The solution is to penalize observations on the **wrong side of the margin** by introducing *slack variables* to the optimization problem.

$$\min_{b,\mathbf{w},\xi} \sum_{i=1}^{N} \xi_i + \frac{\lambda}{2}\|\mathbf{w}\|^2$$
$$\text{s. t } y_i(b + \mathbf{w}' x_i) \geq 1 - \xi_i \, \forall i$$
$$\xi_i \geq 0 \, \forall i$$

Non-separable data

$$\min_{b,\mathbf{w},\xi} \sum_{i=1}^{N} \xi_i + \frac{\lambda}{2}\|\mathbf{w}\|^2$$
$$\text{s.t } y_i(b + \mathbf{w}' x_i) \geq 1 - \xi_i \, \forall i$$
$$\xi_i \geq 0 \, \forall i$$

λ is a parameter that tradeoffs the width of the margin vs. the penalty on observations on the *wrong* side of the margin.

Non-separable data

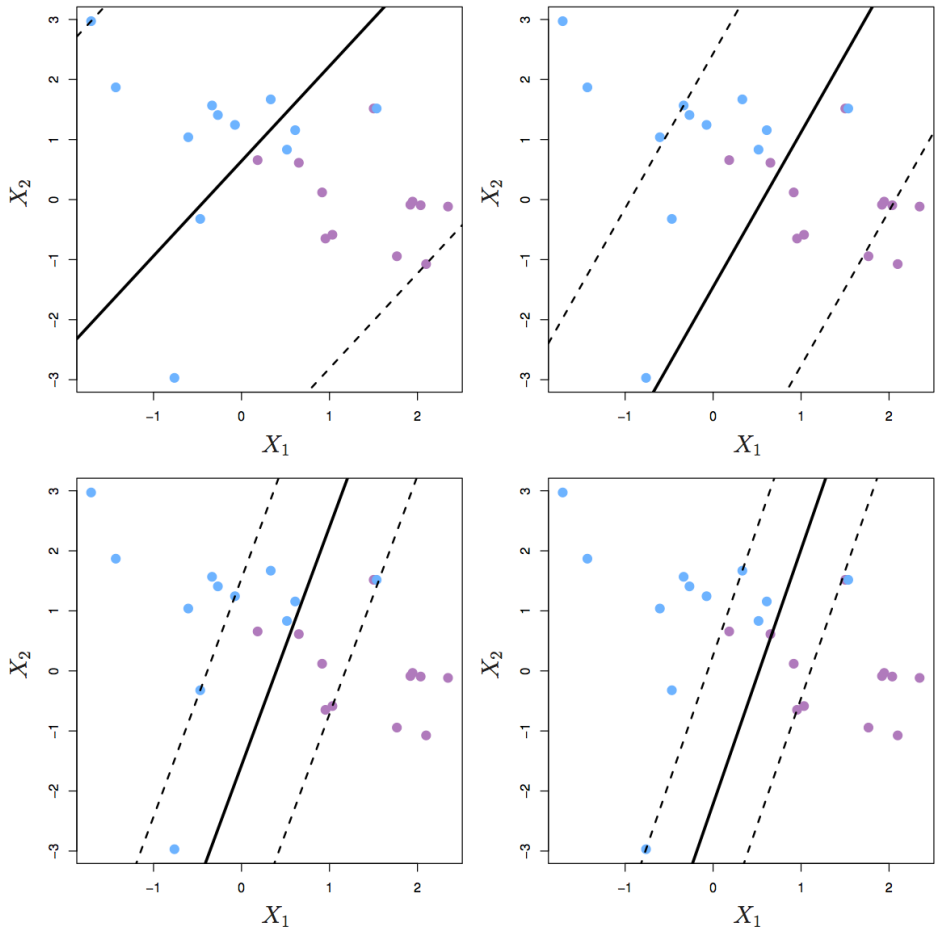
$$\min_{b,\mathbf{w},\xi} \sum_{i=1}^{N} \xi_i + \frac{\lambda}{2}\|\mathbf{w}\|^2$$
$$\text{s.t } y_i(b + \mathbf{w}' x_i) \geq 1 - \xi_i \, \forall i$$
$$\xi_i \geq 0 \, \forall i$$

λ is a parameter that tradeoffs the width of the margin vs. the penalty on observations on the *wrong* side of the margin.

This is a "data fit + model complexity" loss function.

Non-separable data

λ is a hyper-parameter to be selected by the user or via cross-validation model selection methods.



The SVM as a regularized estimation method

$$\min_{b,\mathbf{w}} \sum_{i=1}^{N}(1 - y_i f_i)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

If observation x_i is on the proper side of the margin,

then $y_i f_i > 1$ and thus $(1 - y_i fi)_+ = 0$.

The SVM as a regularized estimation method

$$\min_{b,\mathbf{w}} \sum_{i=1}^{N}(1 - y_i f_i)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

If observation x_i is on the proper side of the margin,

then $y_i f_i > 1$ and thus $(1 - y_i fi)_+ = 0$.

Otherwise, $(1 - y_i f_i)_+$ equals the signed distance to the margin for observation x_i.

The SVM as a regularized estimation method

$$\min_{b,\mathbf{w}} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

The SVM as a regularized estimation method

$$\min_{b,\mathbf{w}} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

The first term corresponds to a "loss function"

The SVM as a regularized estimation method

$$\min_{b,\mathbf{w}} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

The first term corresponds to a "loss function"

The second term a regularization term that controls model complexity.

Regularized Loss

This is a general framework we will see in many algorithms.

We write the models we are learning as solutions to *optimization* problems with *regularized* objective functions

$$f = \arg\max_f L(y, f) + \lambda R(f)$$

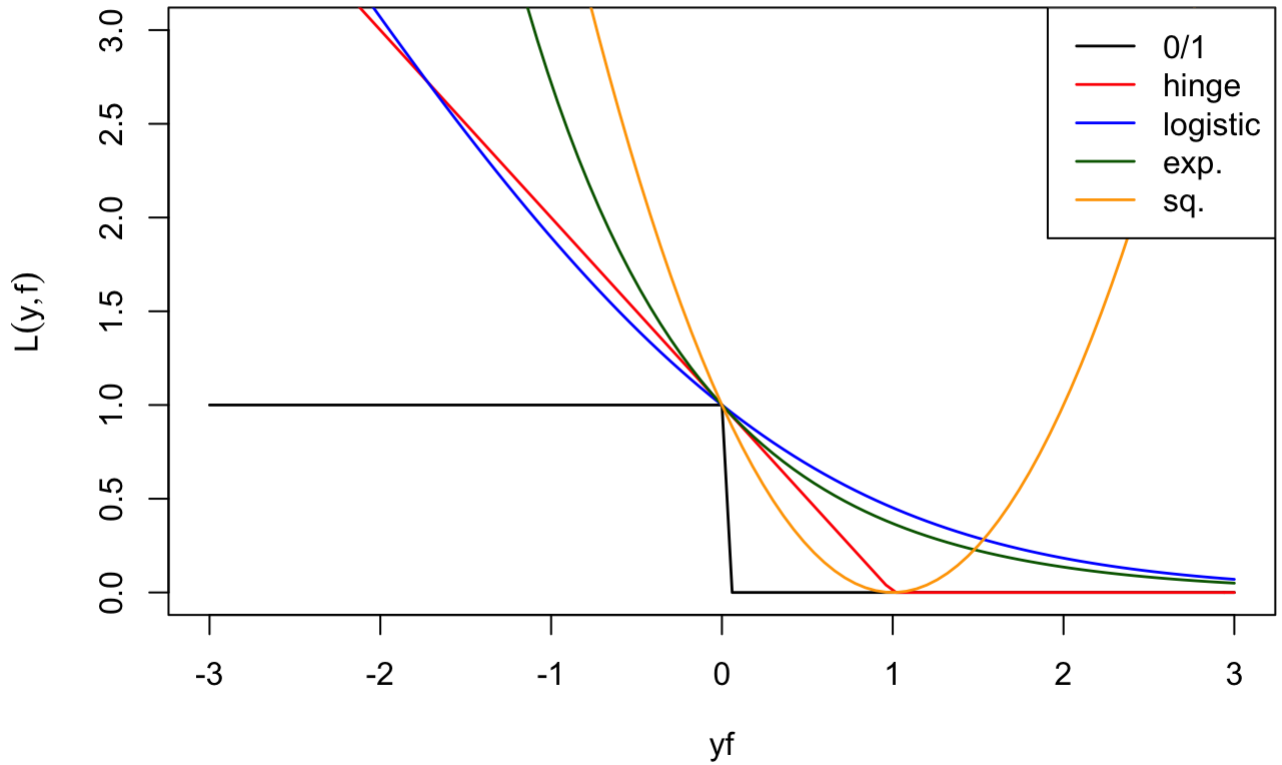
- $L(y, f)$ is a loss function, e.g., $\sum_{i=1}^{N}(1 - y_i f_i)_+$
- $R(f)$ is a *regularizer*, e.g., $\|\mathbf{w}\|^2$

Regularized Loss

Other loss functions:

- Zero/one loss: $L(y, f) = \mathbf{1}[yf \leq 0]$
- Hinge: $L(y, f) = (1 - yf)_+$
- Logistic: $L(y, f) = \frac{1}{\log 2} \log(1 + \exp -yf)$
- Exponential: $L(y, f) = \exp -yf$
- Squared: $L(y, f) = (y - f)^2$

Regularized Loss



Regularized Loss

The reason we like this is that now we have tons of flexibility in learning models

We can apply gradient descent to our loss and regularizer of choice as appropriate to specific application

Regularized Loss

Quiz Derive gradient descent for the SVM!

Support Vector Machines

Different algorithms depending on data size

- Massive number of examples with few predictors, train with stochastic gradient descent
- Moderate number of examples, use quadratic optimization (later in semester)
- For quadratic version, can subset observations that could be support vectors

Support Vector Machines

State-of-the-art for many applications

We will see later that makign this a non-linear model is very powerful and straightforward

Very elegant formulation serves as springboard to understand many models