Robust Models

Spring 2020
Instructor: Ankit Shah, Ph.D.

Factors Affecting Model Performance

Class Imbalance

- Suppose you have a training data set with multiple (discrete) classes as your response variable values
- Say, the relative frequencies of these classes are unequal
- You apply one of the models to fit the training data set
- Will this impact your model's training and making predictions for the data samples in your testing data set?

- If there was a significantly low proportion of one (or more) of the classes in your training data set, then it is a <u>class imbalance</u> issue that may affect the performance of your model

Class Imbalance

- How did it affect the performance of the model?
  - The class imbalance will bias predictions to the majority class

Confusion Matrix

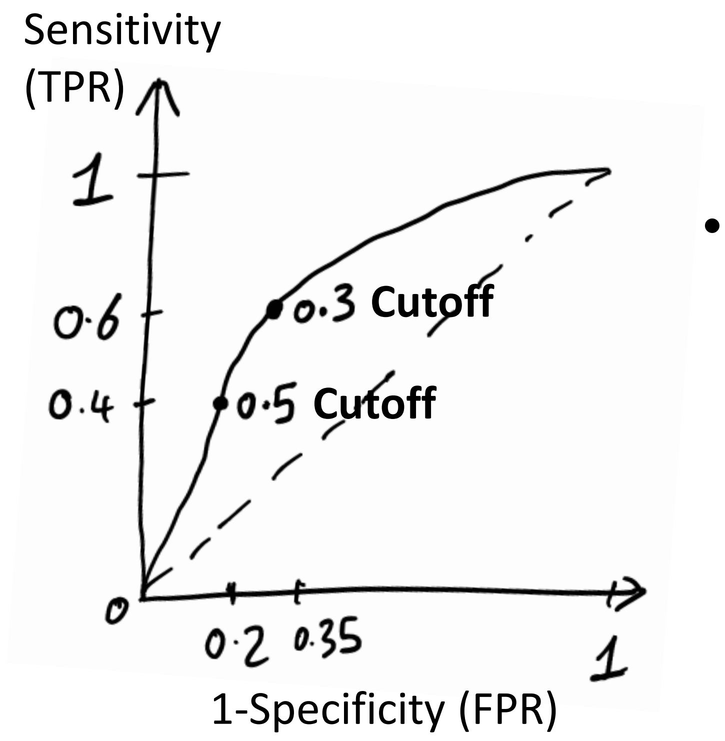
Confusion matrix is a cross-tabulation of the observed and predicted classes for the data

	Observed	
Predicted	Event	Nonevent
Event	**True Positives (TP)**	**False Positives (FP)**
Nonevent	**False Negatives (FN)**	**True Negatives (TN)**

- Top row corresponds to samples predicted to be events
- Bottom row corresponds to samples predicted to be nonevents
- Sensitivity = $\frac{TP}{TP+FN}$
- Specificity = $\frac{TN}{TN+FP}$

Receiver Operating Characteristic (ROC) Curve

- In Spam Filtering, whether to classify an email as Junk or Not, the focus is on Specificity
- Specificity = $\frac{TN}{TN+FP}$
- False Positives need to be minimized
- More important to minimize deletion of valid emails by incorrect classification as Junk
- Trade-off between reducing FP (Specificity) and FN (Sensitivity)
- Receiver Operating Characteristic (ROC) curve is one technique for evaluating this trade-off



- ROC curve is a graphical representation of how Sensitivity and Specificity vary as we change the probability threshold
- It can be thought of as a summarization of all the confusion matrices that can be obtained by varying the threshold from 0 to 1

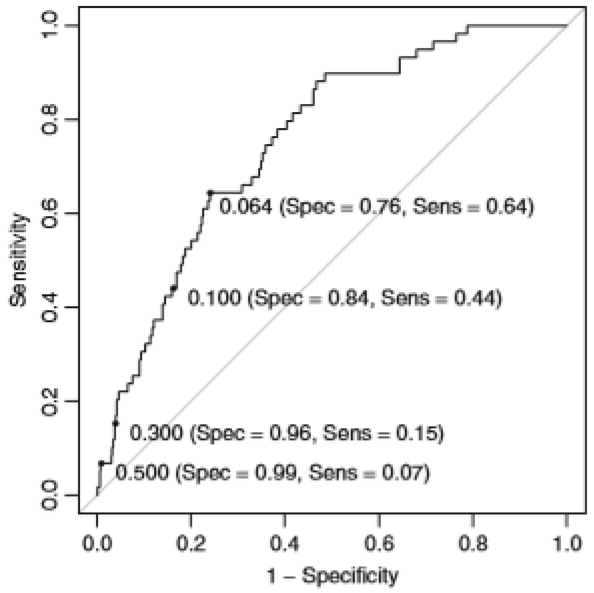
Class Imbalance: Alternate Cutoffs

Class Imbalance: Alternate Cutoffs

Improve the prediction accuracy of the minority class samples

- For a 2-class classification problem, one method for improving the performance of the model is to determine alternate cutoffs for the predicted probabilities that define a predicted event

- An approach is to use the ROC curve since it calculates the sensitivity and specificity across a continuum of cutoffs



- If we decrease the cutoff, it will increase the sensitivity at the expense of specificity
- Perhaps, 0.064 might be a good cutoff here – giving a good balance between the two

- Comparing models based on default sensitivity and specificity might be misleading in many classification problems
  - A better cutoff may be possible based on the ROC curve

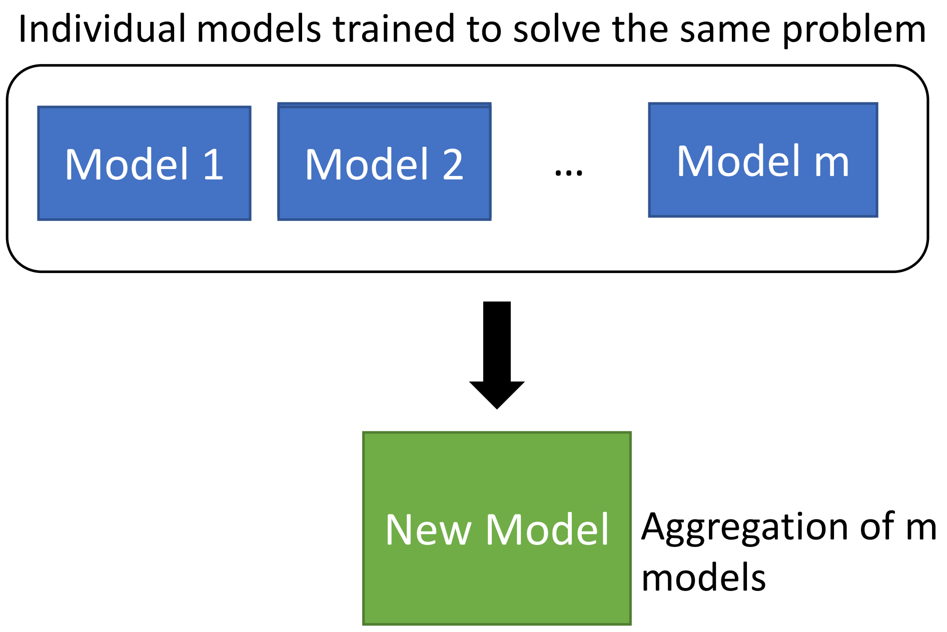
This is a post-processing method to redefine the class predictions!

Class Imbalance:
Assigning Different Weights to the Samples

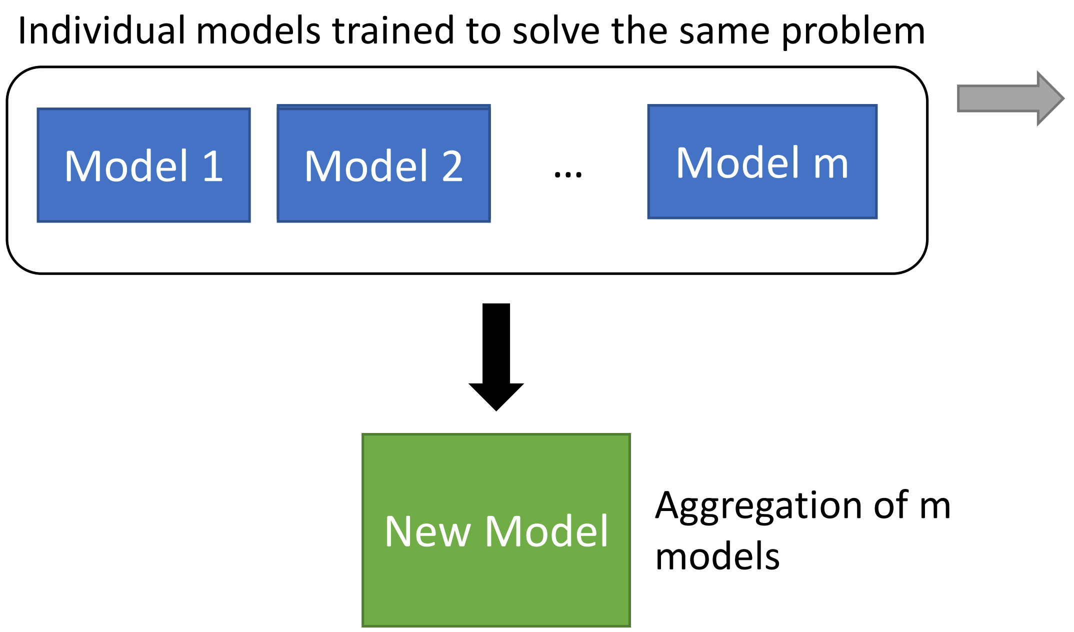
Class Imbalance: Assigning Different Weights to the Samples

- Another method to improve the model performance due to the issue of class imbalance is by assigning more emphasis on the individual data points from the minority class (which will be incorrectly classified) during the training phase
- An example of such a method is – Boosting

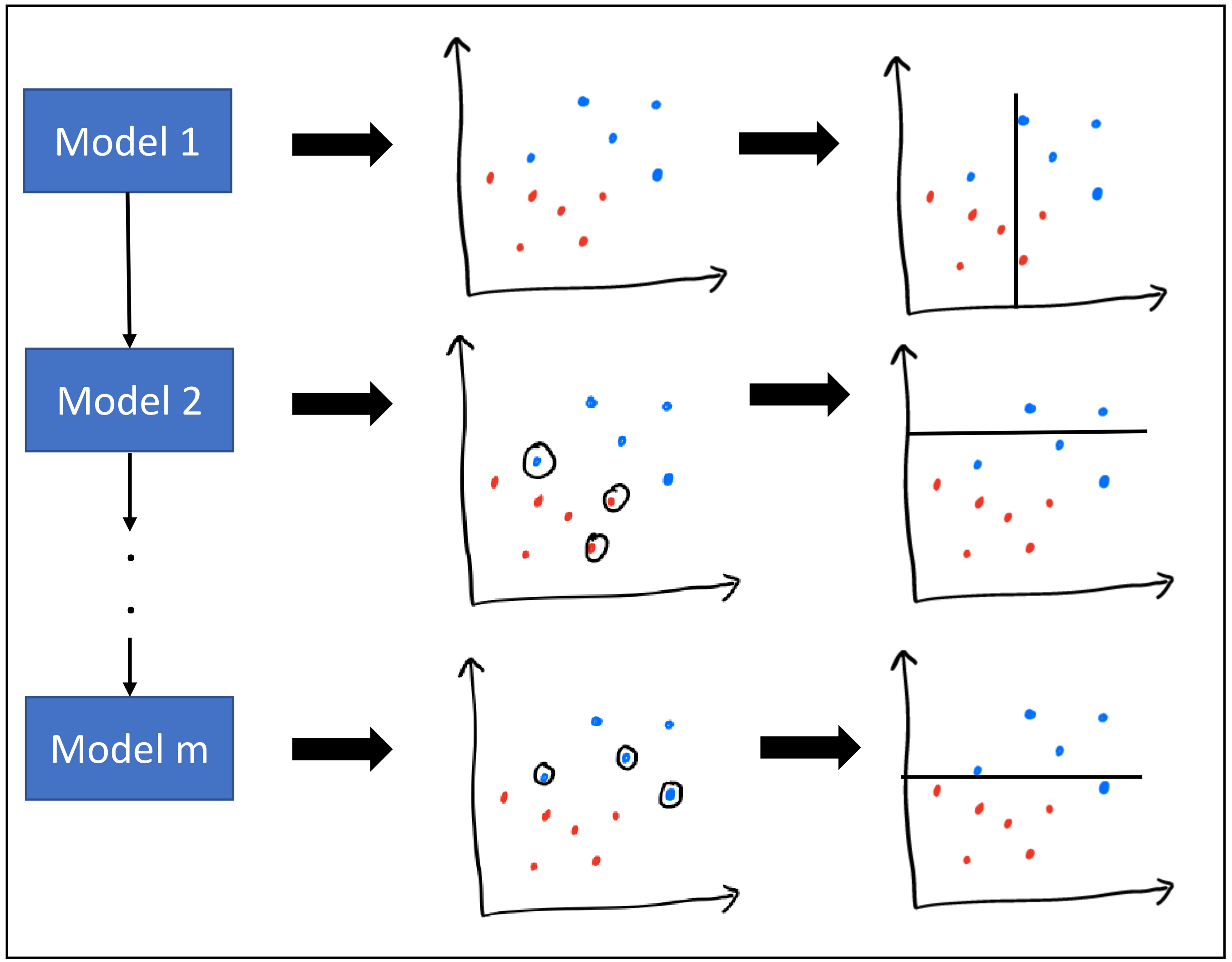
- Boosting is an ensemble technique
- In boosting, the "weak" learners are fit <u>sequentially</u>
- Data that was not handled correctly in the previous model becomes the focus of the current model
- This process continues until we obtain a "strong" learner, which has **a low bias**



Boosting



Individual models trained to solve the same problem
Model 1
Model 2
...
Model m
New Model
Aggregation of m models

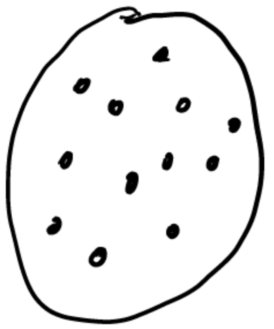


Model 1
Model 2
Model m

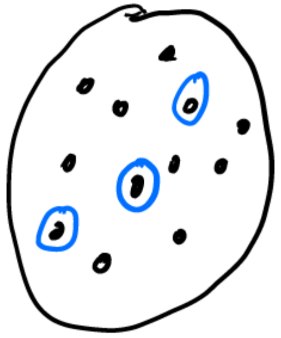
Adaptive Boosting - Classification

Model 1

Assign each sample an equal weight (1/n)



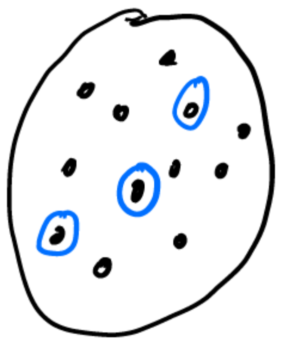
Data set

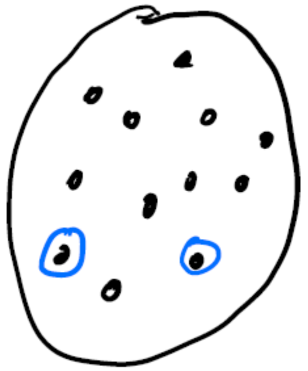


Record the misclassification rate (ε) and compute the model's value:
½ * ln((1- ε)/ε)

Model 2

Assign higher weights to the misclassified samples





Record the misclassification rate (ε) and compute the model's value:
½ * ln((1- ε)/ε)

Model m

Upon Training:

Model	ε	Value
1	0.27	0.424
2	0.65	-0.310
3	0.5	0

To classify a sample:

Predicted Class
1
-1
-1

$$\sum_{m} Predicted\ Class\_m \ * Value\_m$$

1*0.424 + (-1)*(-0.310) + (-1)*0

If the above sum is positive, then class is 1
Else, class -1

Class Imbalance: Sampling Methods

Class Imbalance: Sampling Methods

- One of the easiest way to sample is by selecting a training set where there are roughly equal event rates
  - In other words, instead of the model trying to balance the class frequencies, we can handpick the balanced frequencies
- This may not be possible in all cases, as there might be a significant difference between the frequencies of the classes and there may not be a good size training data set that could be obtained with this approach
- In this case, what are our options?

- There are 2 very important post hoc approaches that can help reduce the effects of an imbalanced data set
  - Up-sampling
  - Down-sampling

Class Imbalance: Sampling Methods

Up-sampling

- A technique that simulates or imputes additional data points to improve balance across classes
- One approach is to keep sampling cases from the minority class with replacement until the classes are balanced
- Issue: some data points will show up in the training data set multiple times while the majority class will have data points that are unique

Down-sampling

- A technique that reduces the number of samples from the majority class to balance the classes
- One approach is to randomly sample the majority class such that both the classes are balanced

Class Imbalance:
Sampling Methods

Synthetic Minority Over-sampling TEchnique (SMOTE)

- A data sampling technique that uses up-sampling
- To up-sample for the minority class, this technique simulates new data points
    - It randomly selects a data point from the minority class
    - It finds the K-nearest neighbors of the data point
    - It creates a new synthetic data point

Python code to apply up-sampling technique:

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(k_neighbors=5)
x_training_set, y_training_set = sm.fit_sample(x_training_set,y_training_set)
```

What would happen if the up-sampling procedure is done prior to the data being split into training and test (holdout) sets?

Class Imbalance: Sampling Methods

Near Miss

- A data sampling technique that performs down-sampling
- To down-sample:
  - It calculates the distances between all instances of the majority class and the instances of the minority class
  - It finds n instances of the majority class that are the closest to each of the instances of the minority class – i.e., if there are k instances in minority class, it will find up to k*n instances of the majority class
  - It uses a triage rule to select the equal number of instances from the majority class to match the number of instances of the minority class, i.e., k number of instances

Python code to apply down-sampling technique:

```
from imblearn.under_sampling import NearMiss
nm = NearMiss(n_neighbors=3)
x_training_set, y_training_set = sm.fit_sample(x_training_set,y_training_set)
```

Robust Models - 2

Spring 2020

Instructor: Ankit Shah, Ph.D.

Factors Affecting Model Performance

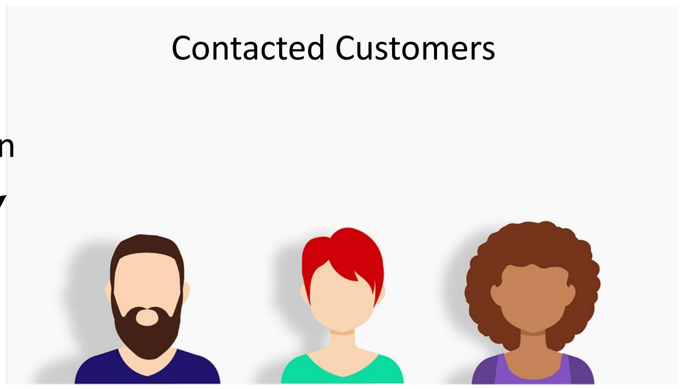
Model Performance

- Often companies get sub-optimal predictions due to reasons that are not related to the model
- One such aspect is the development of a model that answers a wrong question
- i.e., a question that the business really needs an answer for, is not what has been modeled

- Also, known as Type III errors

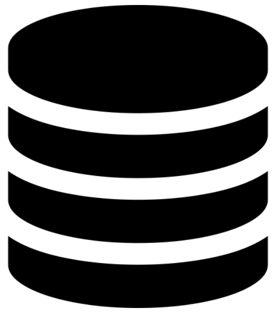
contacted via
email promotion

Contacted Customers



Customers that were not
reached (and new customers)

Stored Data



Information Collected
(Predictors)

- What time they were contacted?
- How many customers ordered?
- How much they ordered ($)?
- What they ordered?

Estimate (Response)

- Expected number of products that will sell

Decisions

- Which products to stock?
- How many quantities?

Noise: Measurement

- Data is collected through some process and there are some measurement errors that could occur. This is the noise/error associated with the measurement system.
  - $Y = f(X_1, X_2, \ldots, X_p) + \varepsilon$
  - The error term is independent of the predictor variables
    - e.g., measuring weight of an object on a weighing scale
    - If the scale has a systematic error – all the observations will be affected
  - Result: poor model performance
  - $\hat{Y} = \hat{f}(x_1, x_2, \ldots, x_p)$
  - $E(\{\hat{Y} - Y\}^2) = [\hat{f}(x_1, x_2, \ldots, x_p) - f(x_1, x_2, \ldots, x_p)]^2 + \mathrm{Var}(\varepsilon)$
  - Reducible error can be minimized by choosing a method that will generate better $\hat{f}$
  - Irreducible error is a constant and is not affected by the method used to produce $\hat{f}$

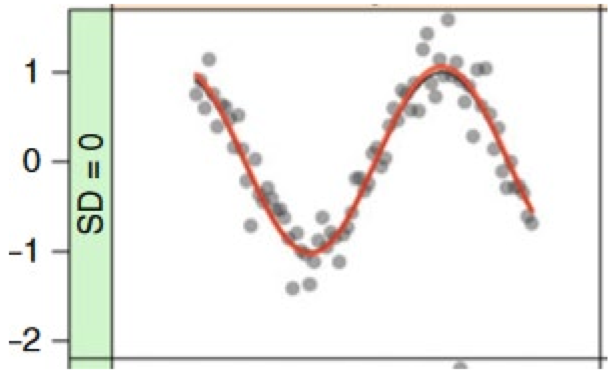
- The more the modeler understands the measurement system, the better is the understanding for the lower bound of the error

Given an extremely noisy system, will there be a significant difference in performance when one uses a highly flexible (complex) method vs a rigid (less complex) method?

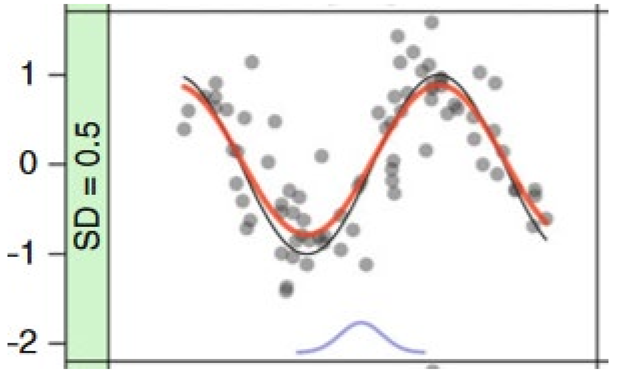
Noise: Measurement

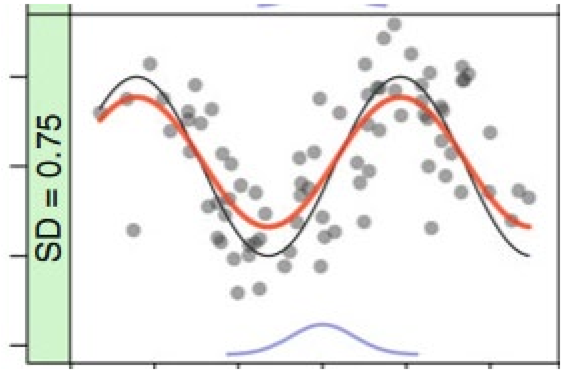
- Typically, it is assumed that there are no measurement errors with the predictors, but this is not always the case
  - An example is ratings conducting by humans
  - Adversarial techniques to manipulate predictor variables

- Data on the x-axis are evenly spaced values
- Response values are obtained by adding some normally distributed noise to the data



True fit (without noise)





As noise increases, the model performance decreases

Noise: Non-informative Predictors

- Another way noise can enter in your model is through attributes associated with the data points that have no relationship with the response variable
- There are certain methods that can eliminate or filter out such predictors, so they have no effect on the predictive performance of the model

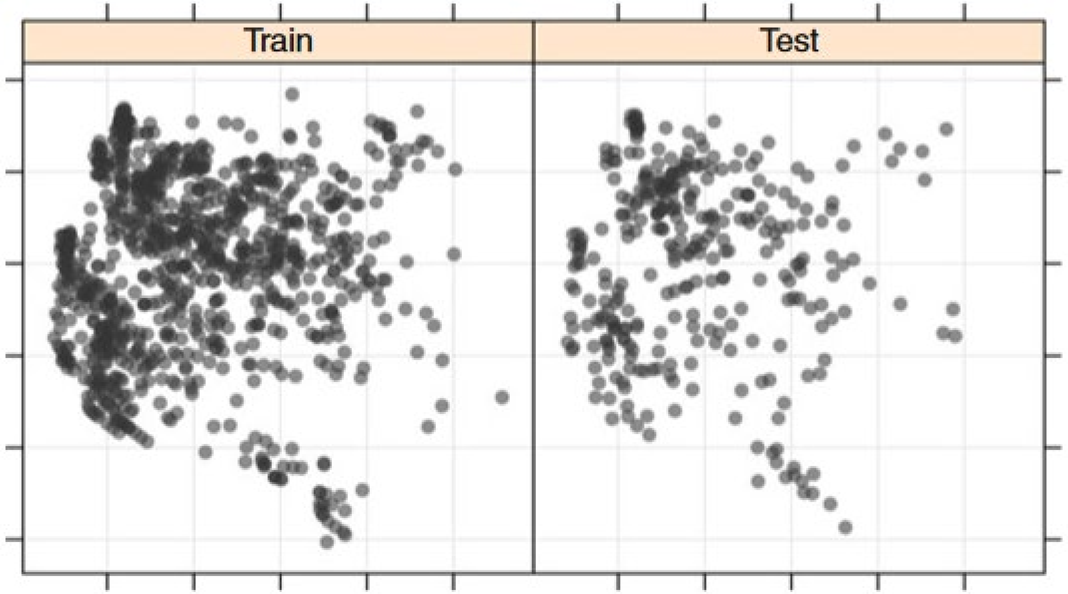
The Underlying Assumption

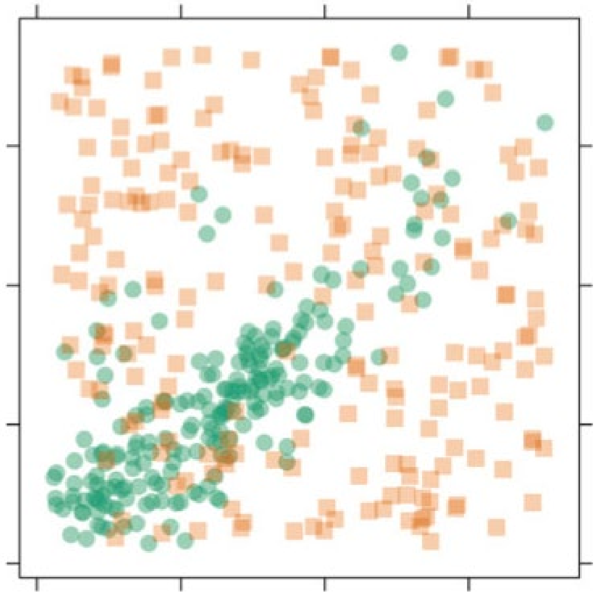
- An underlying assumption is that the mechanism that generated the training data set will continue to generate the new training samples
- Only in this event, we can be confident that the model we create will have good prediction accuracy for a new unseen data sample
- If this new sample is outside the range of the training data, what can be done?
    - Extrapolation
    - Such samples may not be trustworthy and will lead to poor predictions

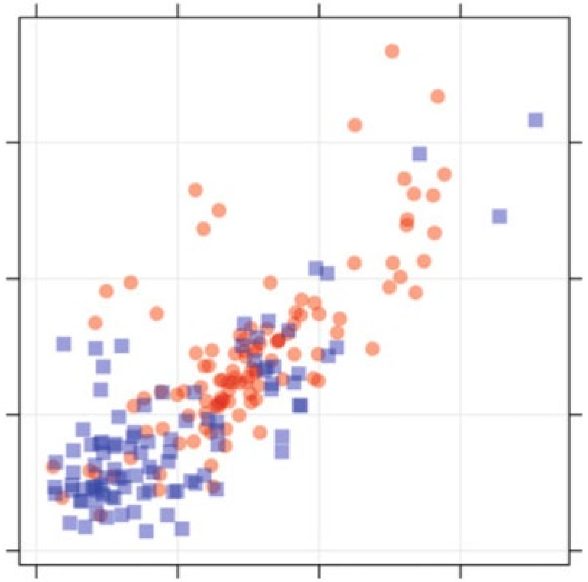
- Is it possible to know if the underlying mechanism is same for both the test and train data sets?
- If there are a few number of predictors we can examine the scatter plots
- However, if the dimensionality increases, this will be inefficient

- The applicability domain of the model is the region of the predictor space where the model is expected to make accurate predictions

The Underlying Assumption







- If the training data and test data are generated from the same mechanism, then the projection of these data will overlap in the scatter plot
- However, if training data and test data are found in different parts of the scatter plot, then they might be coming from different mechanisms

Number of Samples

- It is assumed that the size of the training data set or the number of samples is directly related to the model’s performance
  - If the data set is noisy, it minimizes any advantage that could be gained by a large number of samples
- One of the disadvantages is to add computational burden to train the model
- Imagine a single tree that exhaustively searches through all the samples and considers every predictor at a split to obtain the optimal splits at each level of the tree
- Now imagine the use of an ensemble technique, where we have many such trees
- There is a huge tradeoff between the model’s performance and the computational burden
- This effect is compounded when the samples are from the same population
  - i.e., there is no new signal to learn/train the model

- Large data set is beneficial:
  - Samples contain information through the predictor space
  - Noise is minimal – the predictors and the response values
  - Samples are not similar
  - Computational burden is affordable

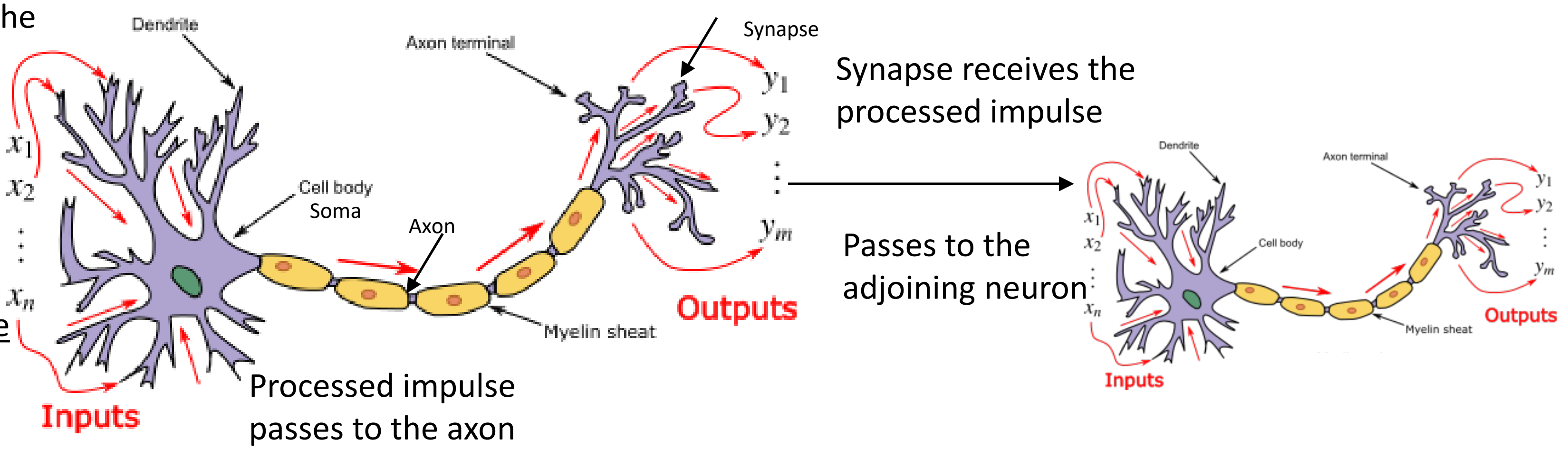
Nonlinear Models (Neural Networks)

Spring 2020
Instructor: Ankit Shah, Ph.D.

Biological Neural Networks

Receives the impulse from the synapse of another neuron

The impulse goes to the nucleus in the cell body (soma)
<u>Impulse gets processed here</u>



Synapse receives the processed impulse

Passes to the adjoining neuron

Processed impulse passes to the axon

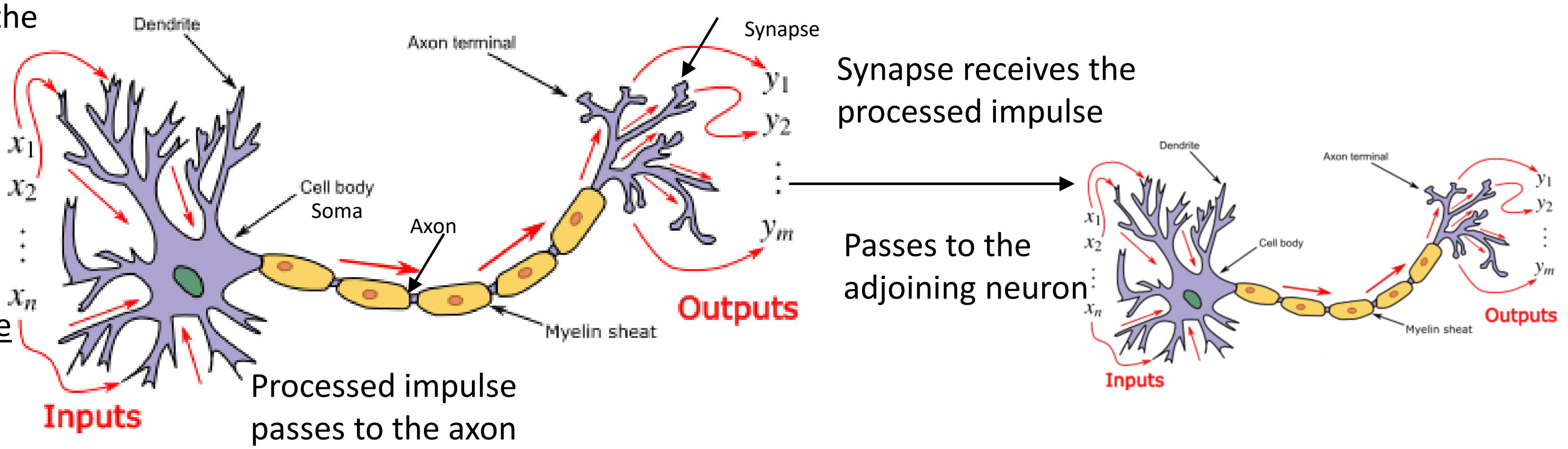
Neuron

- Human brain has a net of neurons (neural networks) – between 14 and 16 billion neurons in cerebral cortex
- Neurons are responsible for transmitting and processing information that we receive from our senses
- Dendrites: receive the information
- Synapses: transmit the processed information
- Soma (cell body) processes the impulses from dendrites and sends the processed impulse to the axon
- Axon: conducting structure through which the processed information is passed

Biological Neural Networks

Receives the impulse from the synapse of another neuron

The impulse goes to the nucleus in the cell body (soma)
<u>Impulse gets processed here</u>



Synapse receives the processed impulse

Passes to the adjoining neuron

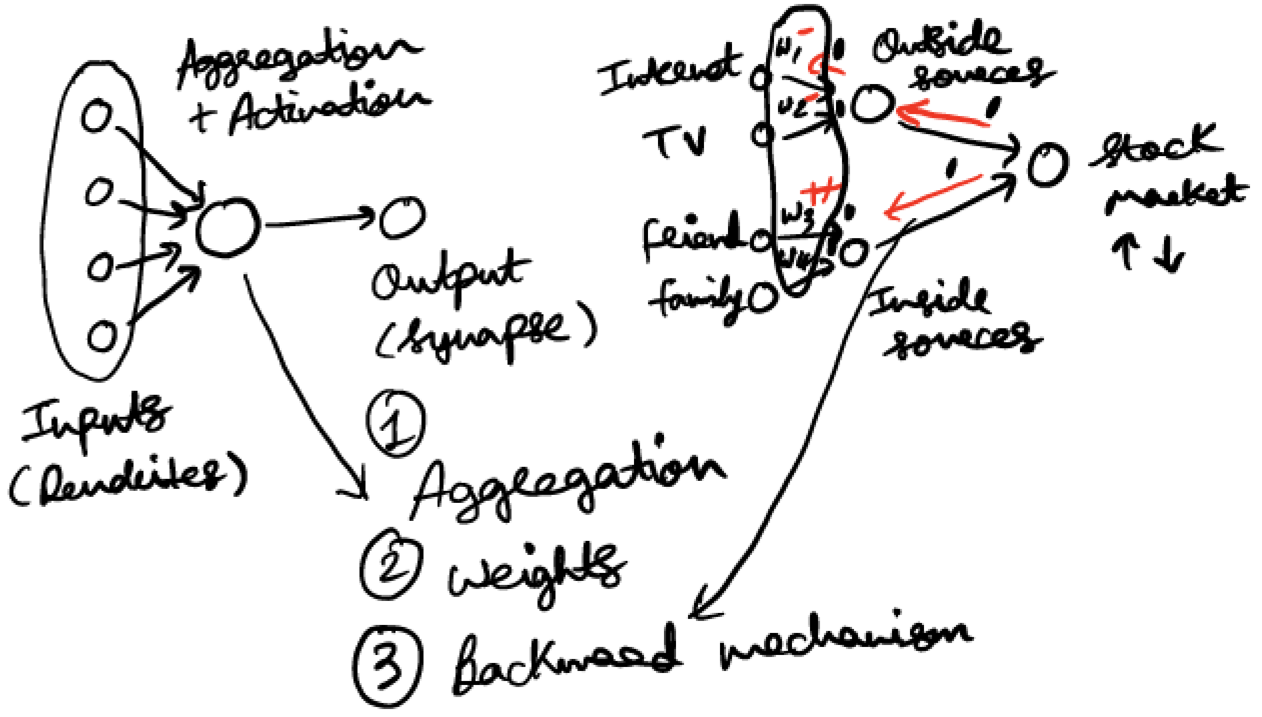
Processed impulse passes to the axon

Neuron

- Some impulses are more important than others and can trigger a neuron to fire easier
- In reality, there is no physical connection between neurons – chemicals are used to communicate the signals (impulses) from synapses to dendrites among neurons
- One neuron is connected to an adjoining neuron at either the entry or exit point(s)
- The neural networks learn patterns (which neurons to fire and the magnitude of signals) and create memories in our brain

Biological Neural Networks: An Example

I am providing the snapshot from the lecture as a placeholder.
Please go through the recorded lecture to understand the details.



Linear Regression: Predict House Prices

Response Variable

House Price

Predictor Variables

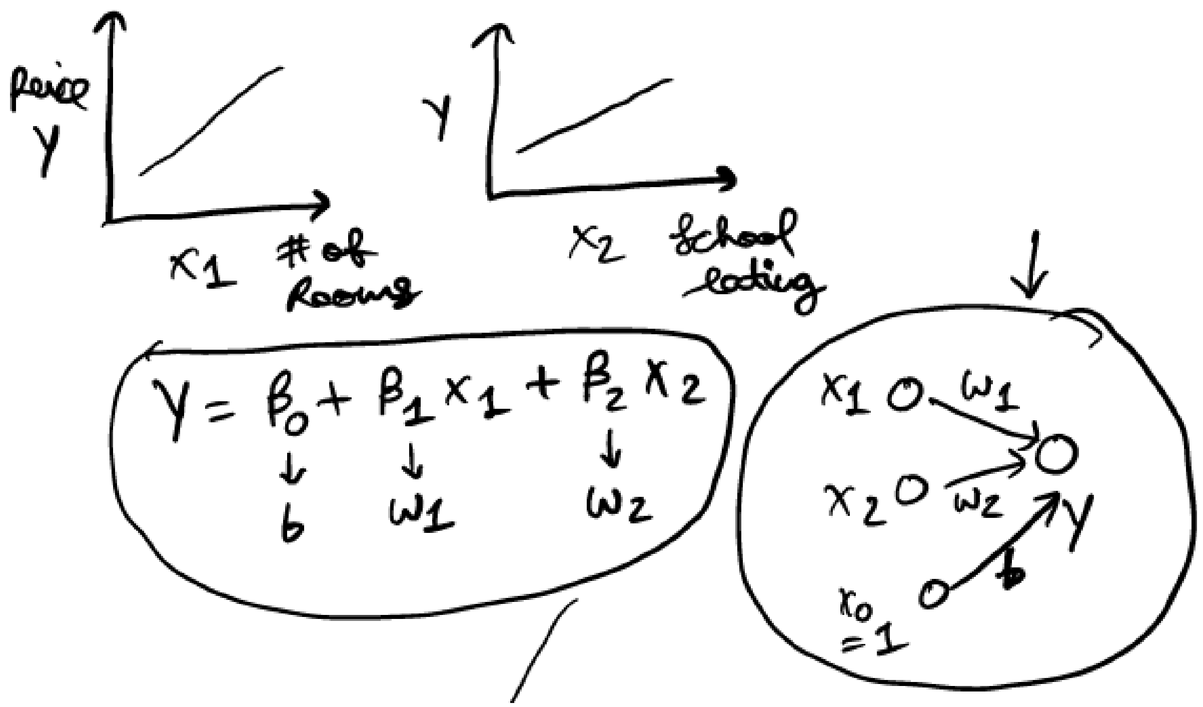
School Ratings

integer values (1-10)

Number of Rooms

integer values (1-10)

New Representation



I am providing the snapshot from the lecture as a placeholder.
Please go through the recorded lecture to understand the details.

Add New Predictors: Mimic a Neural Network

Response Variable

House Price

Predictor Variables

Crime Rate

value between 0 and 1

School Ratings

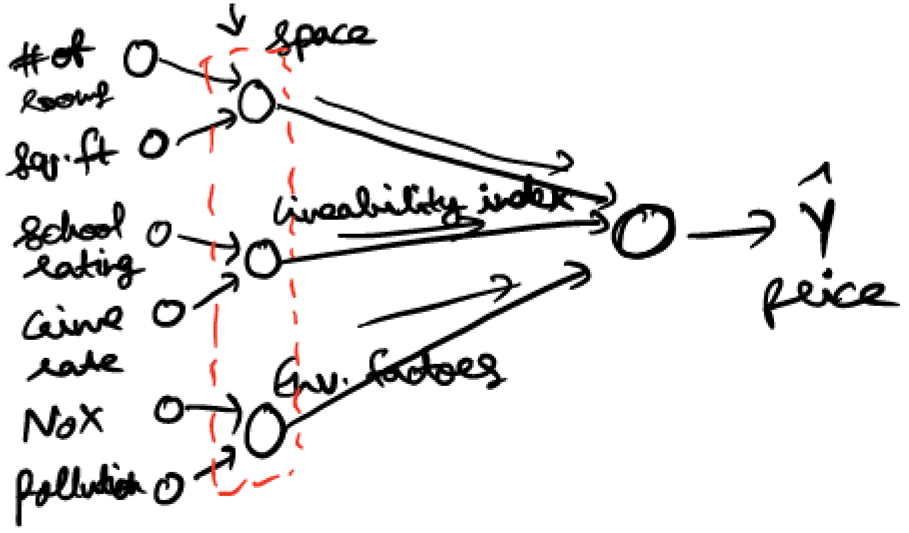
integer values (1-10)

Number of Rooms

integer values (1-10)

Nitric Oxides Concentration

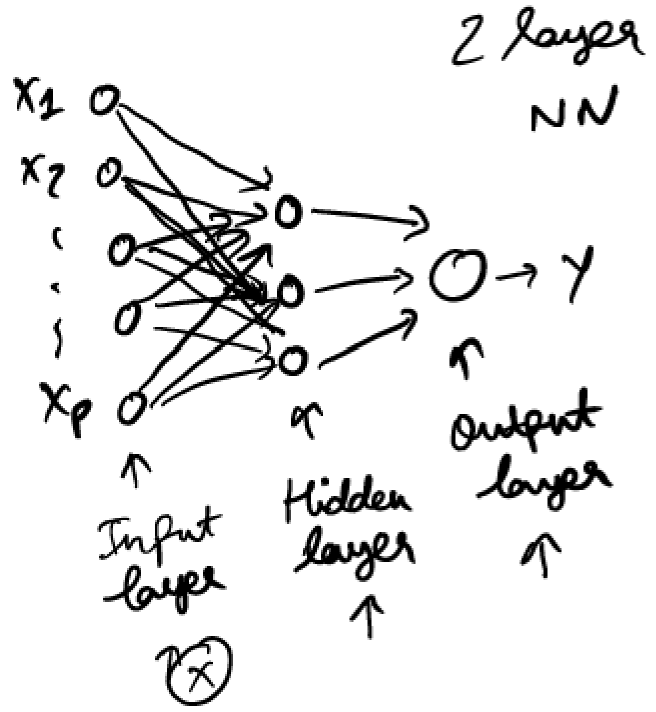
value between 0 and 1



I am providing the snapshot from the lecture as a placeholder. Please go through the recorded lecture to understand the details.

Generalized Representation of Neural Networks

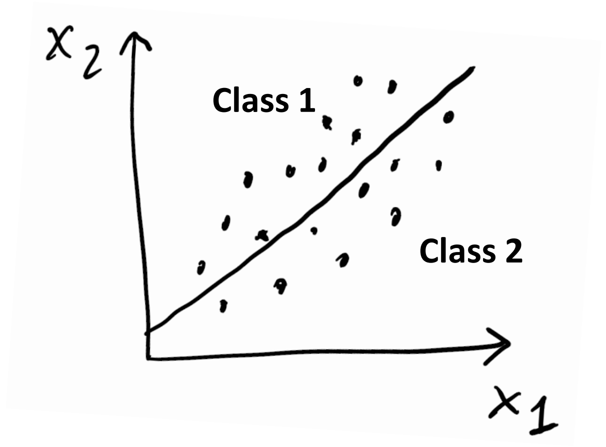
I am providing the snapshot from the lecture as a placeholder.
Please go through the recorded lecture to understand the details.



Also, called a Feedforward Neural Network

Classification

- The objective in classification is to predict a qualitative (categorical or nominal) response outcome, given a set of predictor variable values



- To construct a classifier, we partition the sample space of possible values of X into **non-overlapping** regions
- We give each region a predicted class

- Can we use Linear Regression to create a two-class classifier?
  - If there are only 2 possible outcomes for the response variable, we can assign them as 0-1 and use OLS regression to fit a linear model and obtain a classification boundary (for e.g., class 1 if $\hat{y} > 0.5$)
  - The fitted OLS model: $E(Y|x) = P(Y = 1|x)$ = $\beta_0 + \beta_1 x$
  - The problem here is that unless β_1= 0, the estimates of $P(Y = 1|x)$ will be more than 1 for some values of x

Logistic Regression

- To counter the issue of some predicted probabilities going outside the range of [0,1] when using a linear function, we use the logistic function:

$$p(x) = \frac{\mathrm{e}^{(\beta_0+\beta_1 x\ )}}{1+\mathrm{e}^{(\beta_0+\beta_1 x\ )}}$$

- For any values of the coefficients, positive, negative or 0, $p(x)$ will belong to (0,1)

Please go through the recorded lecture to understand the details.

Estimating the Parameters

- How to estimate w and b?
- Loss function: measures how well we predict $\hat{y}$ with respect to the ground truth label y for each data point in the training set
- Cost function: measures how well the parameters w and b are doing on the entire training set (with respect to the model fit)

Please go through the recorded lecture to understand the details.

Gradient Descent

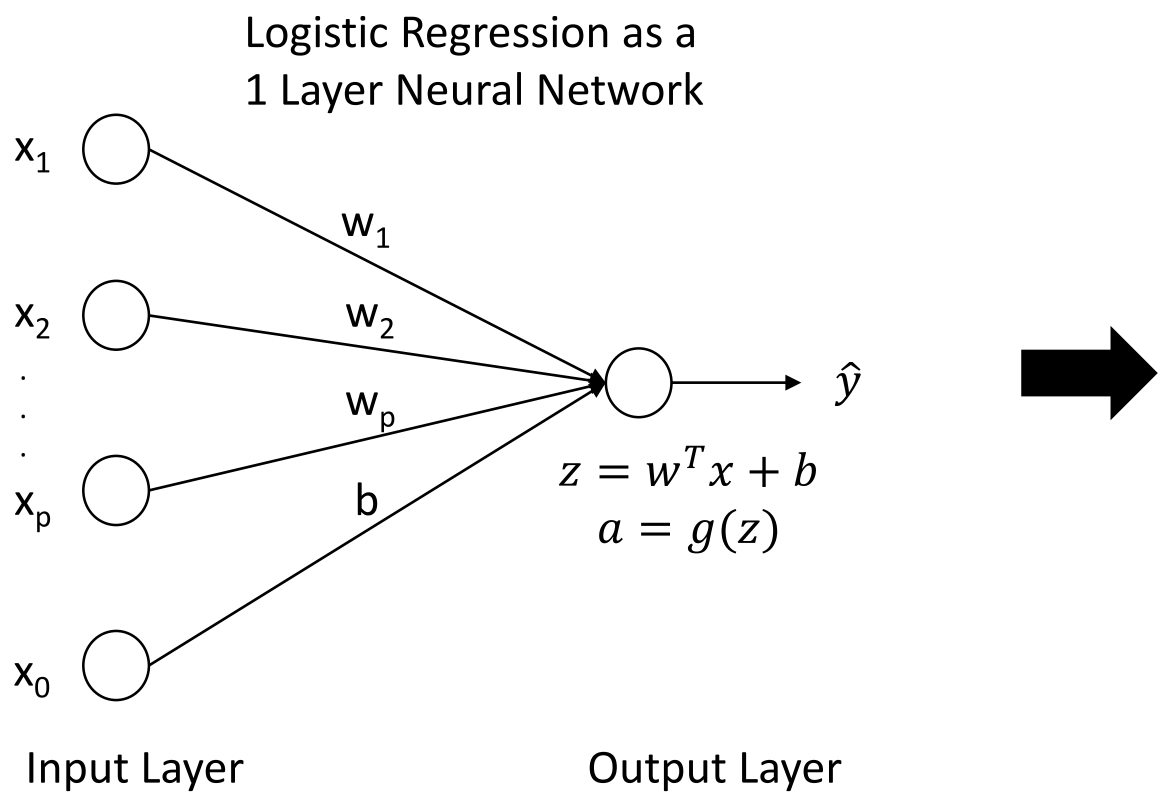
- How to train the model?

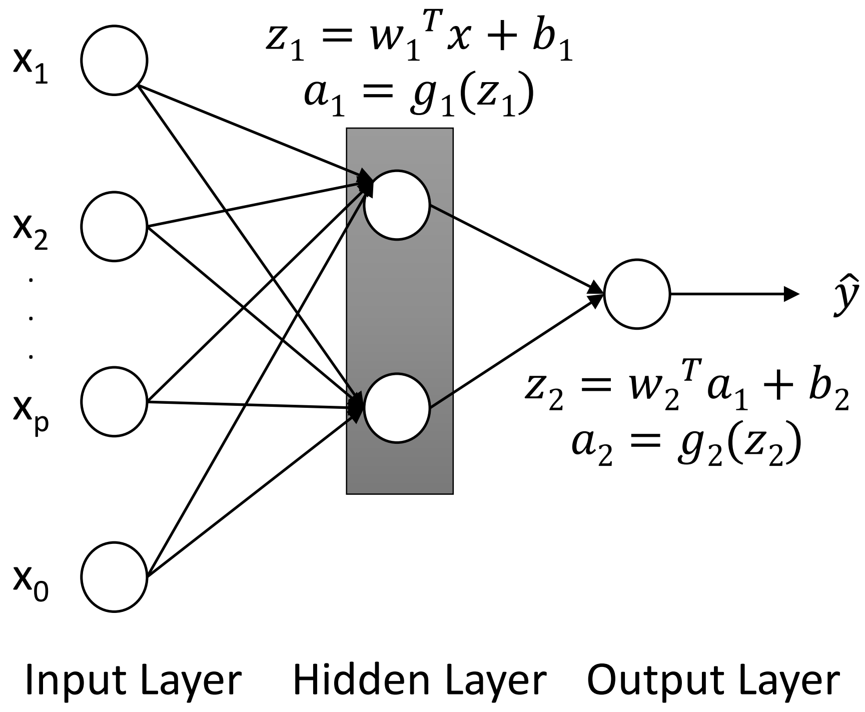
Please go through the recorded lecture to understand the details.

Forward Pass and Backward Pass

Please go through the recorded lecture to understand the details.

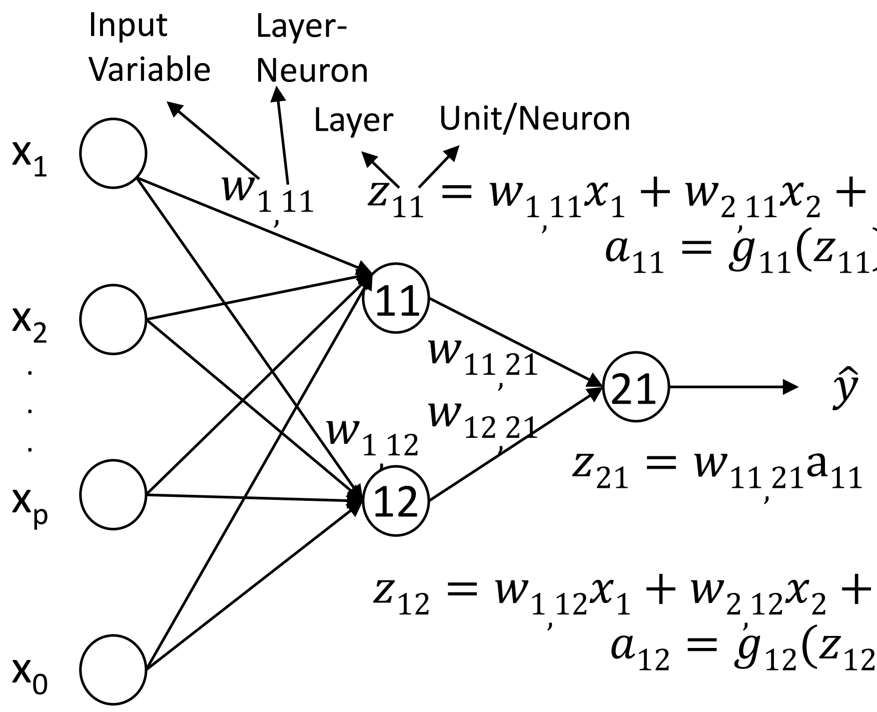
Representation of Neural Networks





Please go through the recorded lecture to understand the details.

Linear Activation Function



$$z_{11} = w_{1,11}x_1 + w_{2,11}x_2 + w_{p,11}x_p + b_1$$
$$a_{11} = g_{11}(z_{11})$$

$$z_{21} = w_{11,21}a_{11} + w_{12,21}a_{12} + b_2$$

$$z_{12} = w_{1,12}x_1 + w_{2,12}x_2 + w_{p,12}x_p + b_1$$
$$a_{12} = g_{12}(z_{12})$$

Please go through the recorded lecture to understand the details.

Regression Problem: Using a Neural Network

Response Variable

House Price

Predictor Variables

Crime Rate

value between 0 and 1

School Ratings

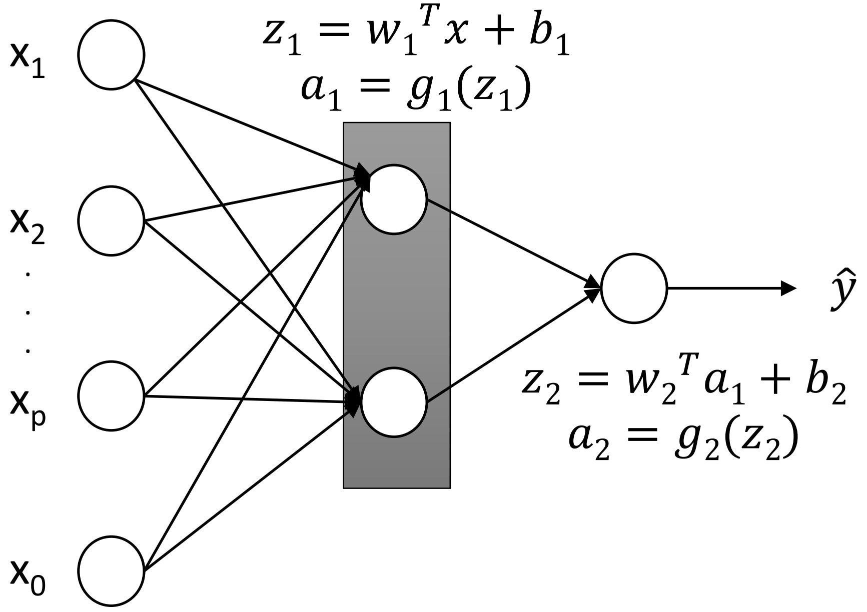
integer values (1-10)

Number of Rooms

integer values (1-10)

Nitric Oxides Concentration

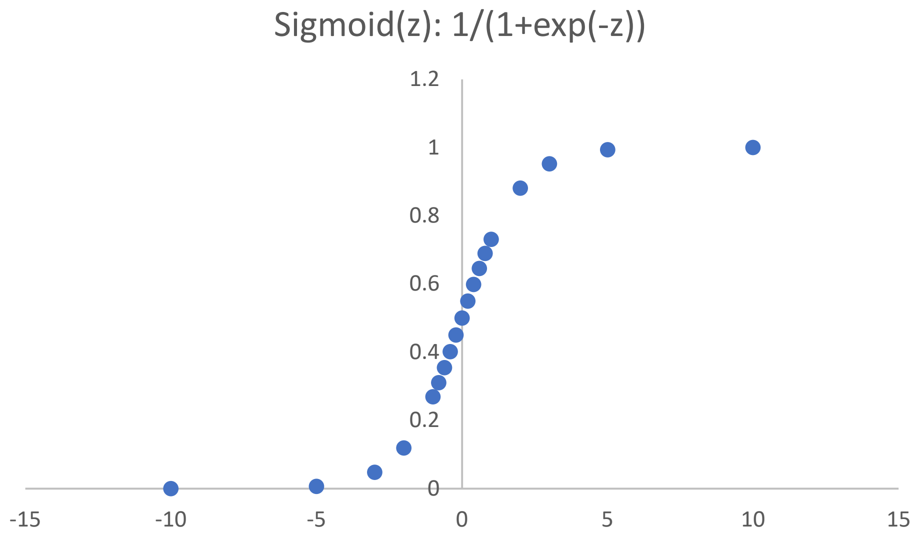
value between 0 and 1



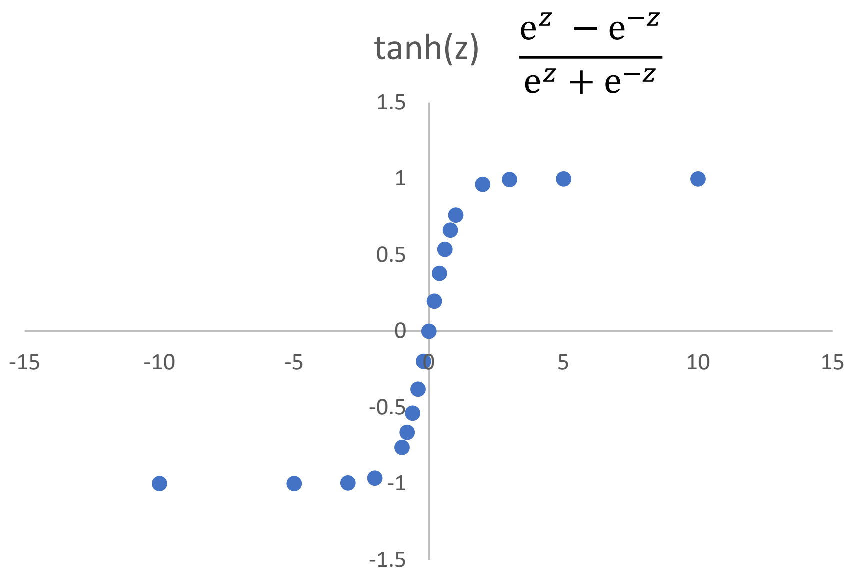
Input Layer Hidden Layer Output Layer

Other Activation Functions

z	1/(1+exp(-z))
-0.2	0.450166003
-0.4	0.40131234
-0.6	0.354343694
-0.8	0.310025519
-1	0.268941421
-2	0.119202922
-3	0.047425873
-5	0.006692851
-10	4.53979E-05
0	0.5
0.2	0.549833997
0.4	0.59868766
0.6	0.645656306
0.8	0.689974481
1	0.731058579
2	0.880797078
3	0.952574127
5	0.993307149
10	0.999954602



z	tanh(z)
-0.2	-0.19738
-0.4	-0.37995
-0.6	-0.53705
-0.8	-0.66404
-1	-0.76159
-2	-0.96403
-3	-0.99505
-5	-0.99991
-10	-1
0	0
0.2	0.197375
0.4	0.379949
0.6	0.53705
0.8	0.664037
1	0.761594
2	0.964028
3	0.995055
5	0.999909
10	1

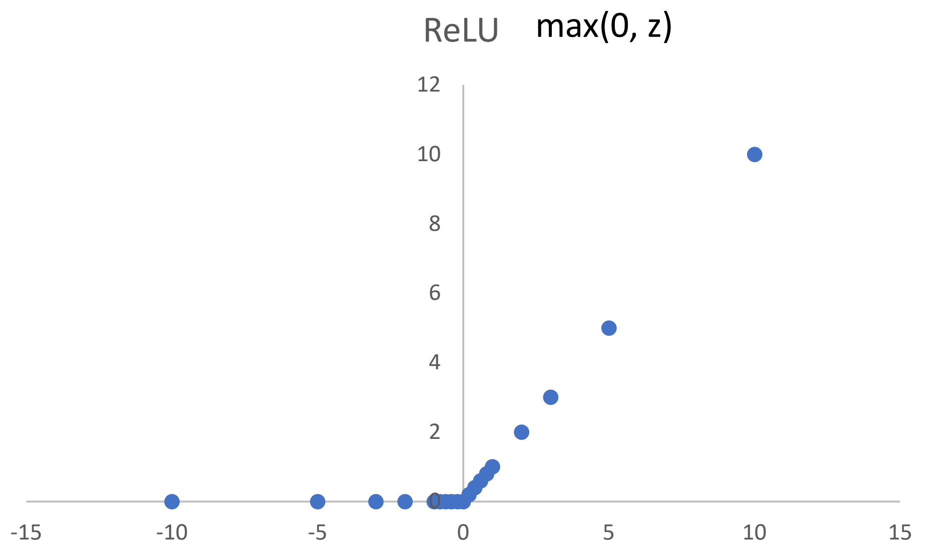


$$\frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Please go through the recorded lecture to understand the details.

Other Activation Functions

z	ReLU
-0.2	0
-0.4	0
-0.6	0
-0.8	0
-1	0
-2	0
-3	0
-5	0
-10	0
0	0
0.2	0.2
0.4	0.4
0.6	0.6
0.8	0.8
1	1
2	2
3	3
5	5
10	10



Please go through the recorded lecture to understand the details.

Parameters and Hyperparameters

- Hyperparameter is a parameter whose value is set before the learning process begins
- Whereas the values of the other parameters are derived upon learning

Please go through the recorded lecture to understand the details.
(List of parameters and hyperparameters in Neural Networks)

Neural Networks (in Python)

Import the module

```
from sklearn.neural_network import MLPClassifier
```

For Regression - MLPRegressor

Create the model

```
model =  MLPClassifier() #a list of parameters that can be passed
```

Optimizer: Adam
(Adaptive Moment Estimation)

Fit the model (on training data)

```
model.fit(x_train, y_train)
```

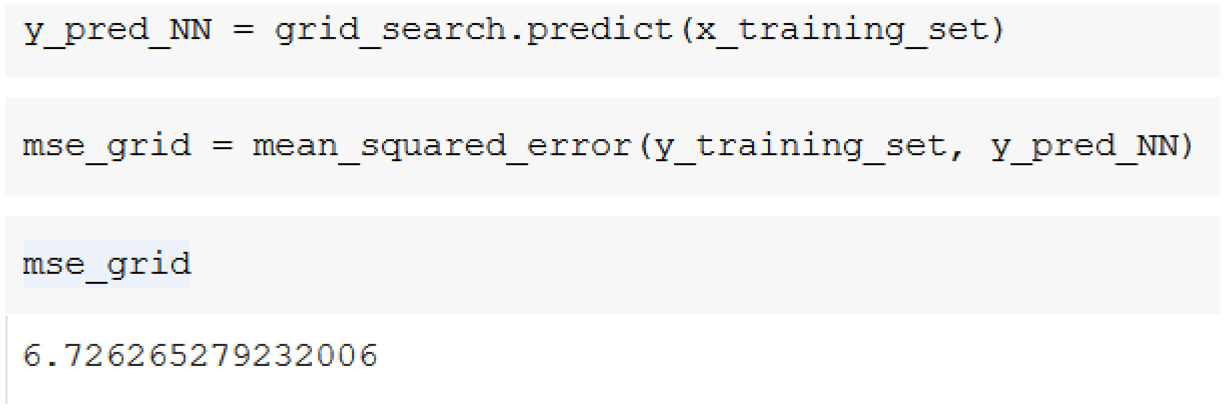
Predict y_hat values for the test data

```
y_hat = model.predict(x_test)
```

Calculate the accuracy: First import the module:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_test, y_hat)
accuracy_score(y_test, y_hat)
```

Result from the final code execution from recorded lecture (#19): NN_Class_Example_2 (regression)



```
y_pred_NN = grid_search.predict(x_training_set)
```

```
mse_grid = mean_squared_error(y_training_set, y_pred_NN)
```

```
mse_grid
```

```
6.726265279232006
```

Nonlinear Models

Spring 2020

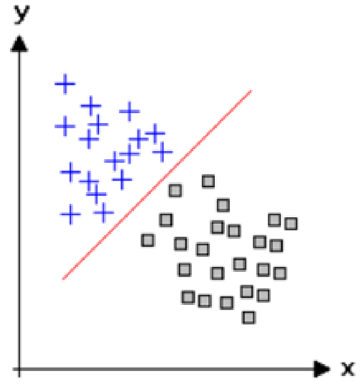
Instructor: Ankit Shah, Ph.D.

Types of Supervised Learning Methods

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} {x_1}^2 + \beta_{22} {x_2}^2$$

$$y = e^{\beta x_1}$$



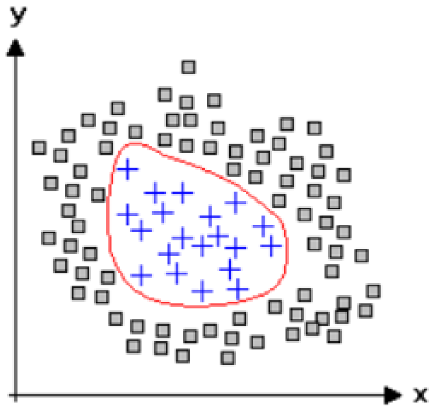
$$\begin{tikzcd}
\text{Linear} & \text{Supervised Learning} \arrow[l] \arrow[r] \arrow[d] & \text{Nonlinear} \\
& \text{Tree-based} &
\end{tikzcd}$$

Linear

Linear Regression
Logistic Regression
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)

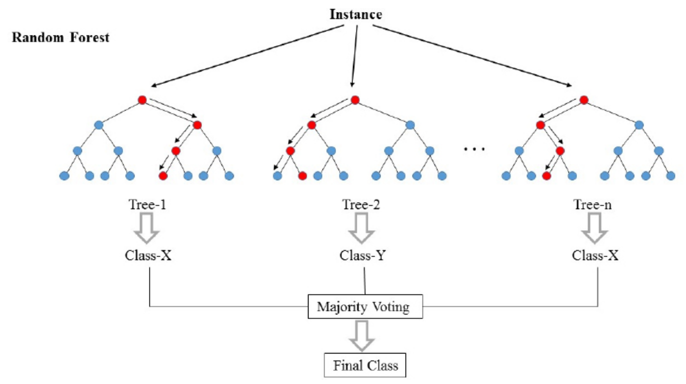
Nonlinear

Neural Networks
Support Vector Machine/Regression



Tree-based

Regression and Classification Trees
Random Forests



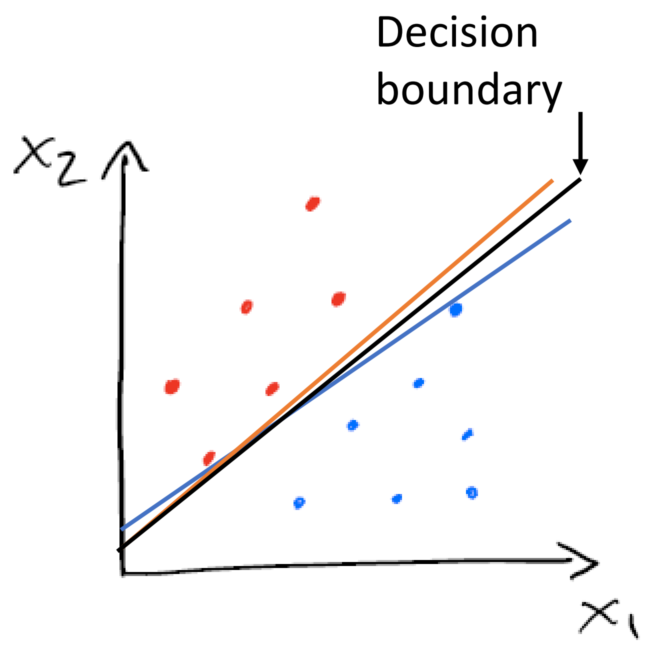
Linear models are linear in the parameters that need to be estimated, but not necessarily in the independent variables

Nonlinear Models

We are first going to motivate the need for a nonlinear model by discussing 2 classifiers that produce linear decision boundaries: 1) Maximal Margin Classifier and 2) Support Vector Classifier

Maximal Margin Classifier

- It is a simple method – more intuitive
- Used when classes can be separated with a linear boundary
- i.e., the classes can be separated with a (p-1)-dimensional hyperplane
- The decision boundary for the maximal margin classifier is the optimal separating hyperplane, which is the farthest away from any of the sample points in the training data



What is a hyperplane?

A hyperplane in a p-dimensional space can be defined by an equation of the form:

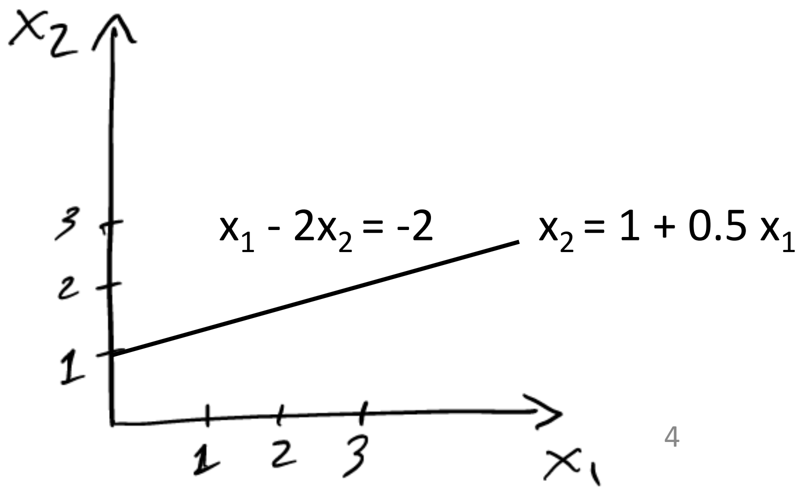
$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0$

Why is the above (p-1)-dimensional?

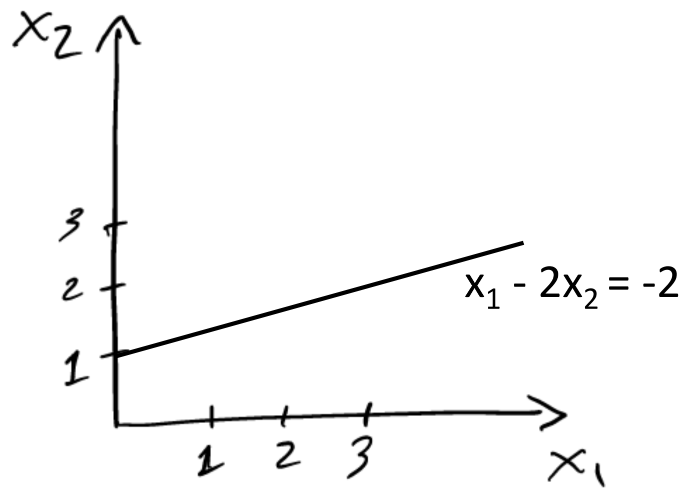
For 2 predictor variables:

$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$

$$x_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} x_1$$



Maximal Margin Classifier



- Say, we have the following data points (x_1, x_2) in the training data:
  (1,3), (3,1), (3,3), (1,1)
  Classify them in 2 different classes: -1 and +1
- *Hyperplane:* $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$
  $2 + x_1 - 2x_2 = 0$
- (1,3)
- (3,1)
- (3,3)
- (1,1)

- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p > 0$, then it belongs to one side of the p-dimensional space: y = 1
- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p < 0$, then it belongs to another side of the p-dimensional space: y = -1
- Generalizing the above: a separating hyperplane has the following property

  $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$ for (i = 1, …, n)

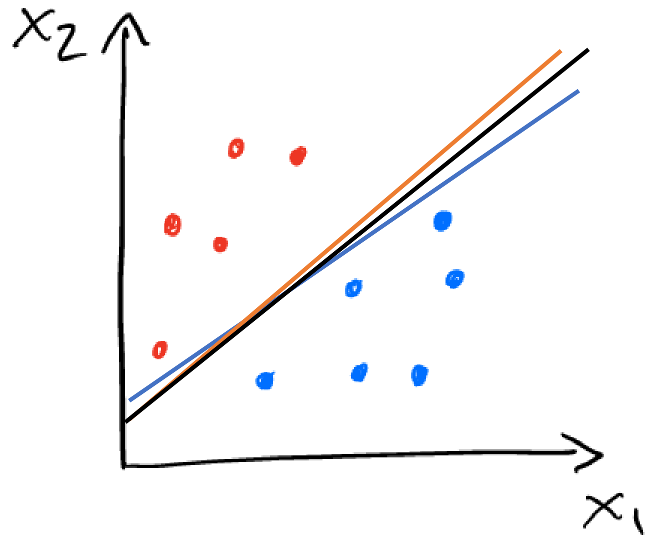
Classification using a Separating Hyperplane

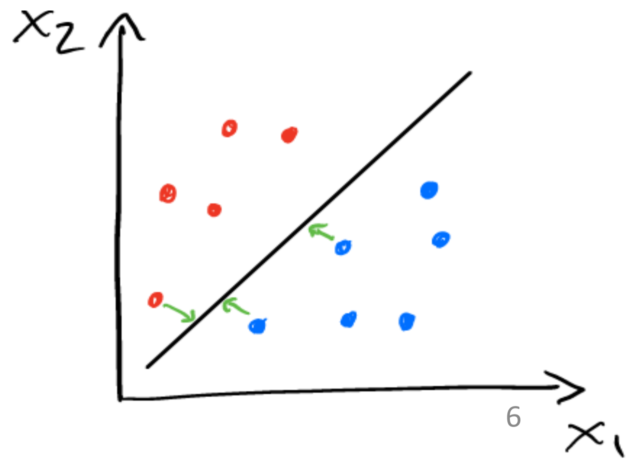
- If a separating hyperplane exists, then there will be an infinite collection of them
- We can adjust the hyperplane by a little and still separate the classes
- We multiply the β_j by a non-zero constant such that the equation still holds true: $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$
- Our objective is to construct a classifier that optimally separates the classes such that the separating hyperplane is the farthest distance from the training observations
    - We compute the perpendicular distance from each training data point to the given separating hyperplane
    - The smallest such distance is the minimal distance from the observations to the hyperplane – <u>Margin</u>
    - Our objective is to maximize the margin
- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

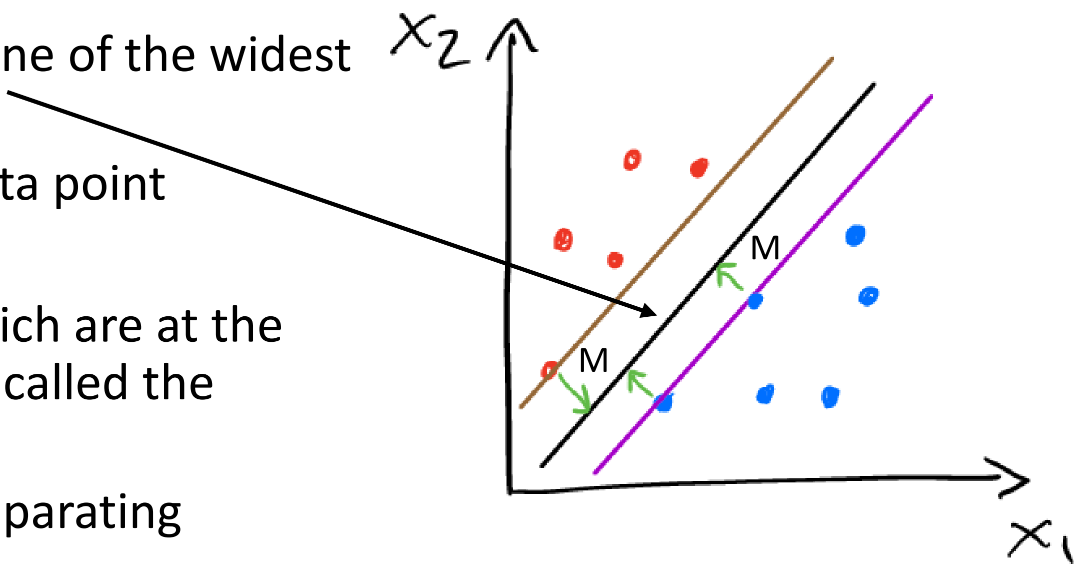
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M$$





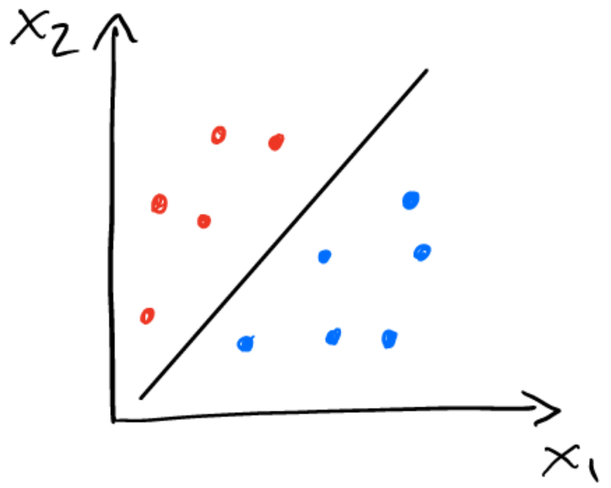
Classification using a Separating Hyperplane

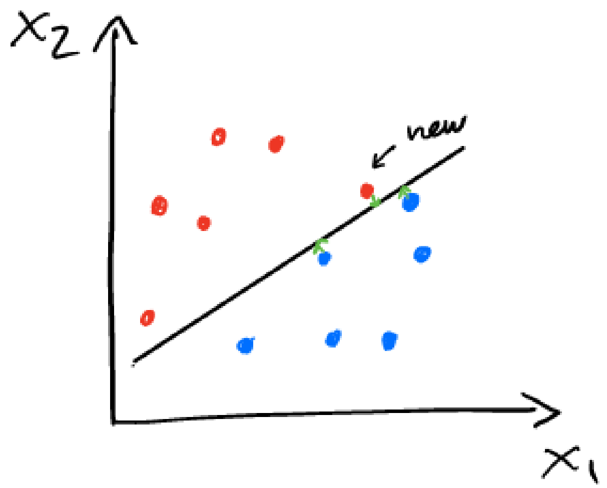
- Think of the maximal margin hyperplane as the mid-line of the widest "slab" that we can insert between the 2 classes
- The value M (smallest distance between a training data point and the hyperplane) is called the margin
- Observations (data points in the training data set) which are at the exact distance M from the separating hyperplane are called the support vectors
- Each data point is at least a distance of M from the separating hyperplane



- How many support vectors do we have in the example above?
- What is the dimensionality of the support vector?

Maximal Margin Classifier





- A small change in the data set can result in a fairly large change in the decision boundary of the maximal margin classifier
- Such sensitivity suggests that the maximal margin classifier overfits the training data

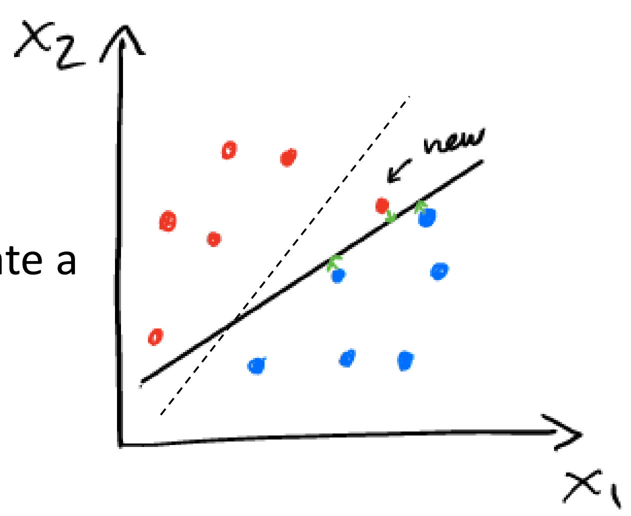
  When the model is tested on unseen data points – would the classification accuracy be lower/higher than that obtained on the training data points? Why?

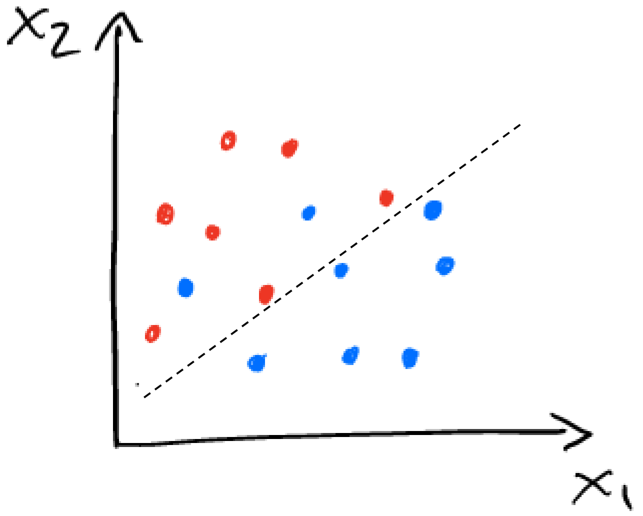
- This motivates a need for a classifier that can be used to produce a more appealing decision boundary

Support Vector Classifier

- Support vector classifier is also called the “soft” margin classifier
- Used when the classes of the training data cannot be perfectly separated with a hyperplane and also when a more appealing decision boundary is needed in which separation is possible

We may have to misclassify a few data points in order to create a more robust model (unseen data points)

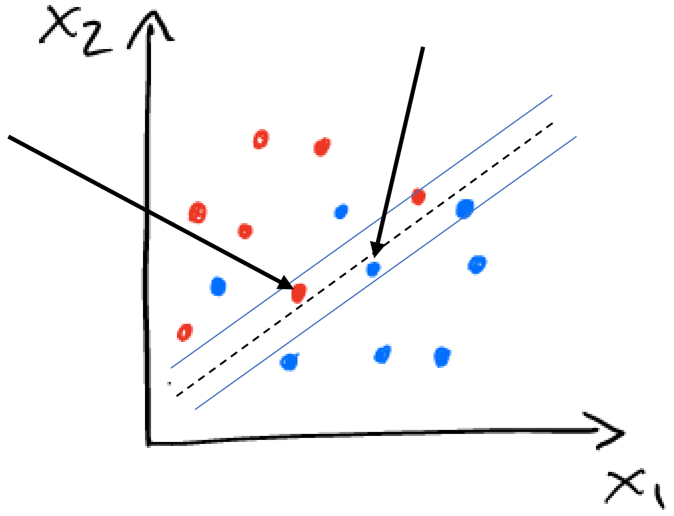




- In the above cases, we may consider a hyperplane that does not separate the 2 classes perfectly
- Our objective is to provide 1) robustness to the model and 2) better classification for majority of the data points

Support Vector Classifier

- As shown in the previous example, we allowed some data points to be on the incorrect side of the hyperplane and also be closer to the hyperplane on the correct side but with a value less than the margin
- Hence, it is a soft margin
  - i.e., it can be violated by some data points to obtain more robust models



- The support vector classifier has a (p-1)-dimensional hyperplane for the decision boundary and has a soft margin that could be violated by a few data points

What is the distance of the closest point to the decision boundary in support vector classifier?

Support Vector Classifier

- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \underset{\text{Slack variable}}{\epsilon_i}),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq \underset{\text{Tuning parameter}}{C},$$

Where will the data point lie, given the following conditions?

$$\varepsilon_i = 0$$

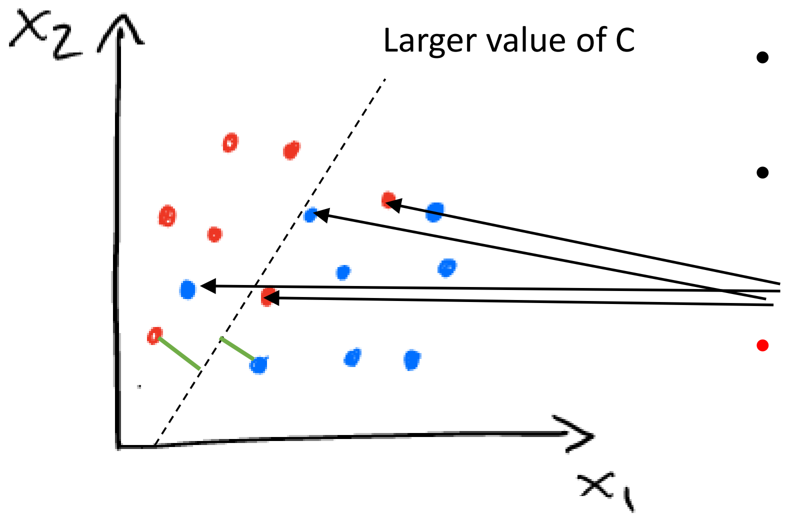
$$\left.\begin{array}{c} 0 < \varepsilon_i < 1 \\ \varepsilon_i = 1 \\ \varepsilon_i > 1 \end{array}\right\} \text{Support vectors: All data points with } \varepsilon_i > 0$$

- ε_i allows the data point *i* to be on the wrong side of the margin or the hyperplane
- C determines the number and severity of the violations to the margin and the hyperplane

What happens when C = 0?

When C > 0, how many data points at most can be on the wrong side of the decision boundary?

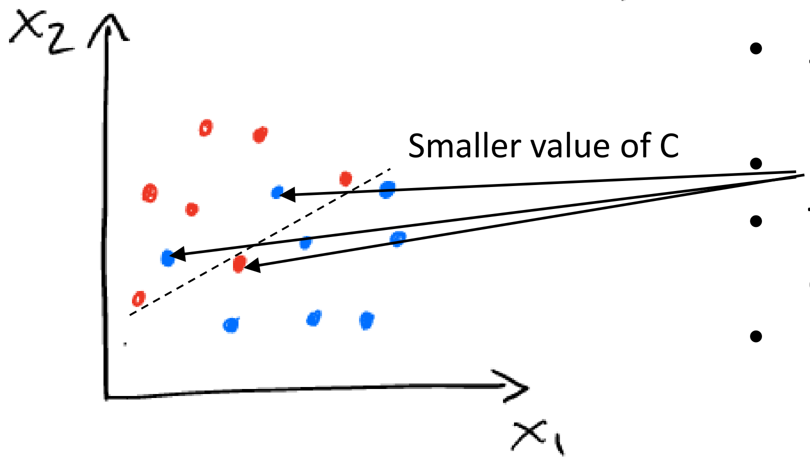
Relationship of C with M



- Larger value of C -> larger value of M
- More slack variables come into play and hence more violators (support vectors)
- This is a condition with?

  Low variance – High bias

  High variance – Low bias



- Smaller value of C -> smaller value of M
- Less number of support vectors
- This is a condition where we are chasing the data points
- Overfit -> Low bias

How do we choose the value of C?

Cross-validation

Support Vector Machines

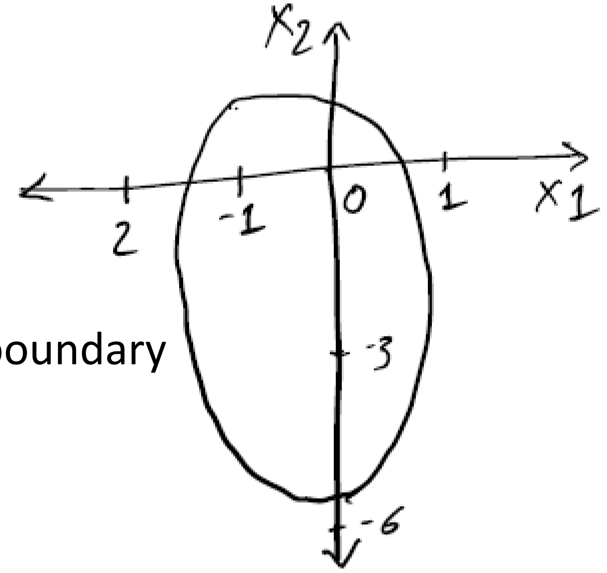
Support Vector Machines

- Suppose we have 2 predictor variables $x_1$and x_2 and we add 2 more predictor variables $x_3 \; = x_1^2$ and $x_4 \; = x_2^2$
- Next, we want to apply the support vector classifier and we obtain the decision boundary that satisfies the condition:
- $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \; \ldots + \beta_{\mathrm{p}} x_p = 0$

$\Big\downarrow$ Obtained after fitting on a training data set

- $-0.13 + 0.7\, x_1 + 0.5\, x_2 + 0.5\, x_1^2 + 0.1\, x_2^2 = 0$ Note that $\sum_j \beta_j^2 = 1$

- From the above we can arrive at:
- $0.5(x_1^2 + 1.4\, x_1) + 0.1(x_2^2 + 5\, x_2) = (1-0.87)$
- $0.5\,(x_1^2 + 1.4\, x_1 + 0.49) + 0.1\,(x_2^2 + 5\, x_2 + 6.25) = 1$
- $0.5(x_1 + 0.7)^2 + 0.1(x_2 + 2.5)^2 = 1$ $\Leftarrow$ Equation of an ellipse

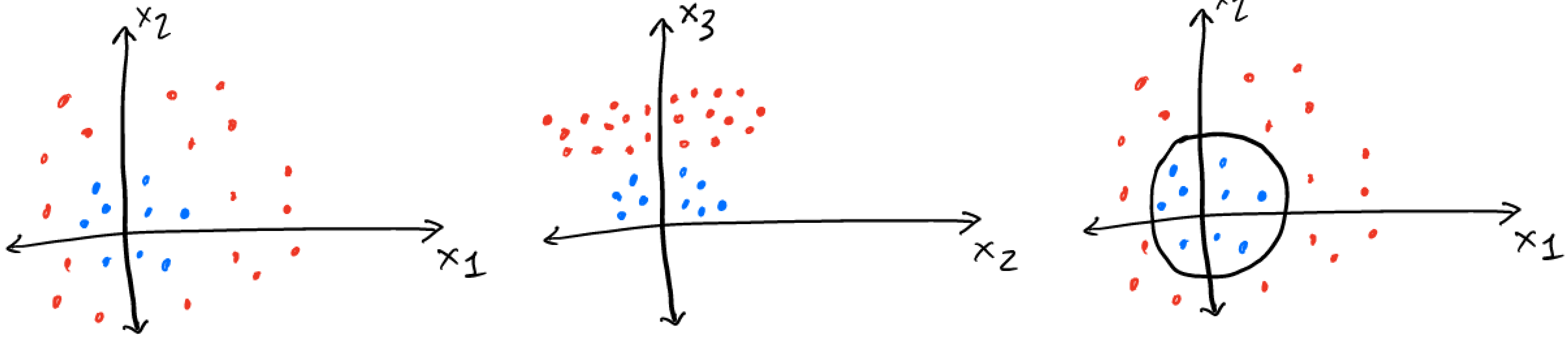


Elliptical decision boundary

The Support Vector Machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way (using *kernels*)

- Kernel method is an efficient computational approach for achieving the above

Support Vector Machines



- There is no separating hyperplane that can segregate the 2 classes
- Apply a transformation (kernel function) that maps the original data points into a higher-dimensional space in which they become separable
- Upon transforming it back to the original plane, we obtain the nonlinear circular boundary as shown above

Nonlinear Models
(Support Vector Regression, K-Nearest Neighbors)

Spring 2020

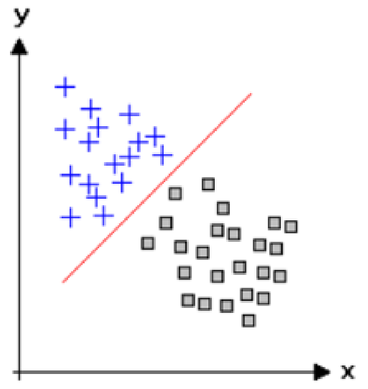
Instructor: Ankit Shah, Ph.D.

Types of Predictive Methods

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} {x_1}^2 + \beta_{22} {x_2}^2$$

$$y = e^{\beta x_1}$$



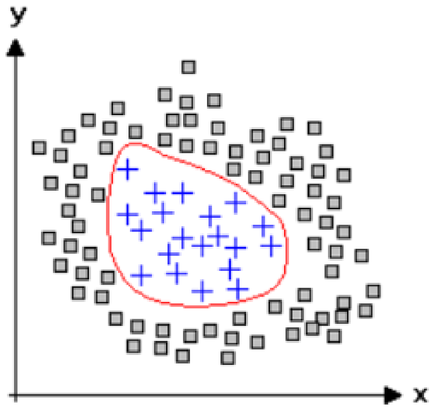
Predictive Methods

Linear

Linear Regression
Logistic Regression
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)

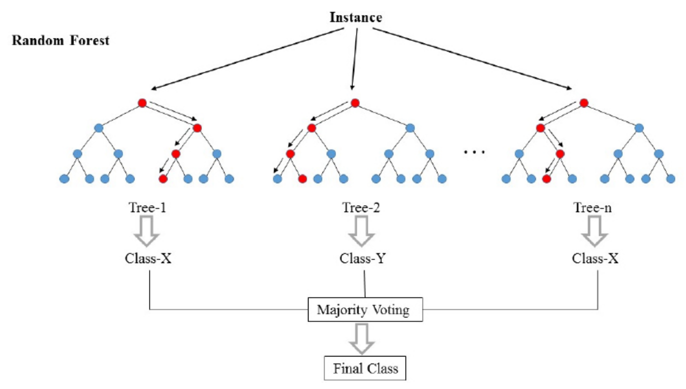
Nonlinear

Support Vector Machine/Regression
Neural Networks



Tree-based

Classification and Regression Trees
Random Forests



Linear models are linear in the parameters that need to be estimated, but not necessarily in the independent variables

Regression

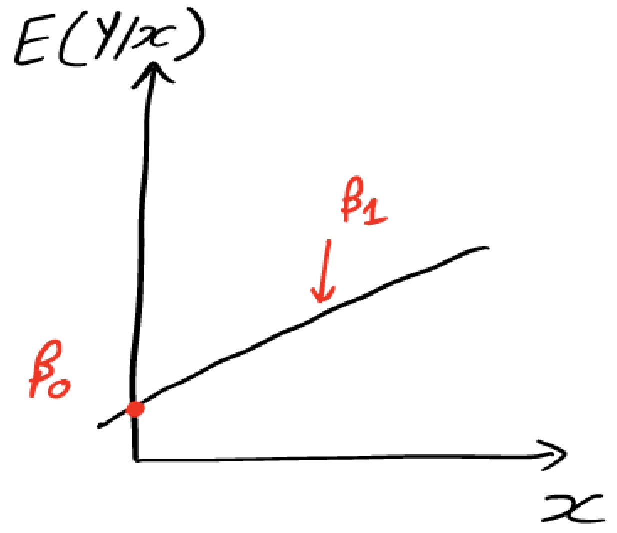
Simple Linear Regression

- Simple Linear Regression:
  - One predictor variable
  - First-order linear model

x	y

$$E(Y_i|x_i) = \beta_0 + \beta_1 x_i$$

where x_i represents the value of the predictor in the ith sample, β_0 is the intercept and β_1 is the slope



- Intercept: expected response value when x takes the value of 0
- Slope of the line: change in the expected response value for one unit change in x

- If we know the values of the parameters β_0 and β_1, then we can predict y_i with good accuracy
- However, if we do not know these values, then we need to estimate $\widehat{\beta_0}$, $\widehat{\beta_1}$
- Once we obtain these estimates, then we can find $\hat{y}_i = \widehat{\beta_0} + \widehat{\beta_1} x_i$

Multiple Linear Regression

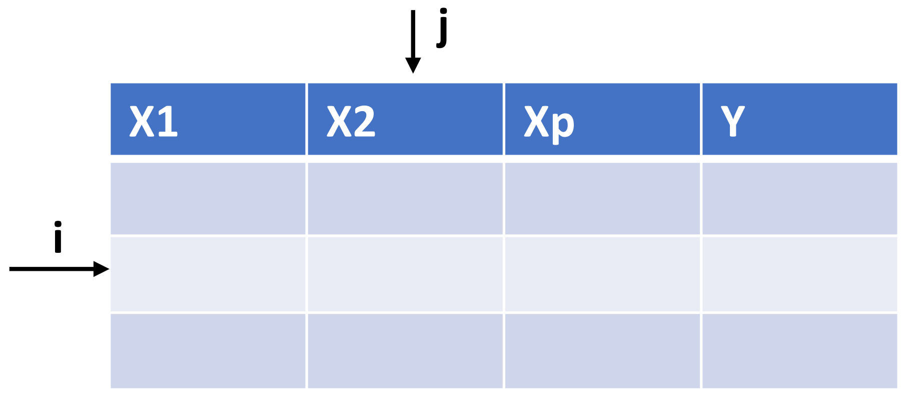
- The simple linear regression model can be extended to allow for more than one predictors:

$$E(Y_i|x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ \dots + \beta_p x_{ip}$$

x_i represents the vector of values for p different predictor variables for the ith sample

x_{ij} is the value of the jth predictor variable for the ith case

β_j is the change in the mean of the response variable due to a one unit increase in x_{ij}, when the other predictor variable values are held fixed

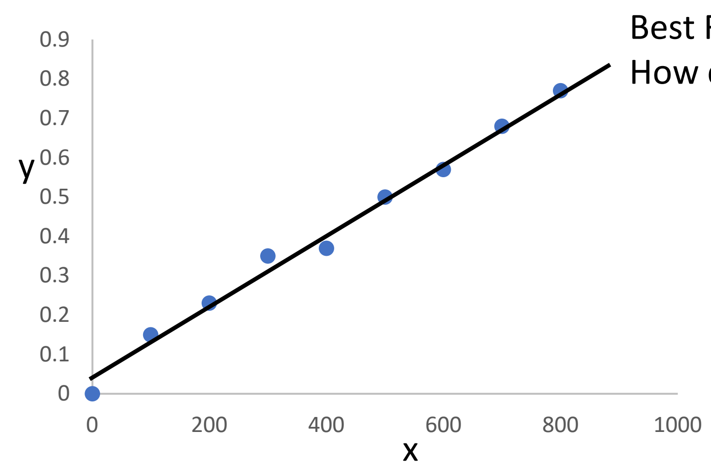


- Ordinary Least Squares method for fitting a simple linear regression model can be extended and the least squares solution can be derived using matrices

Linear Regression

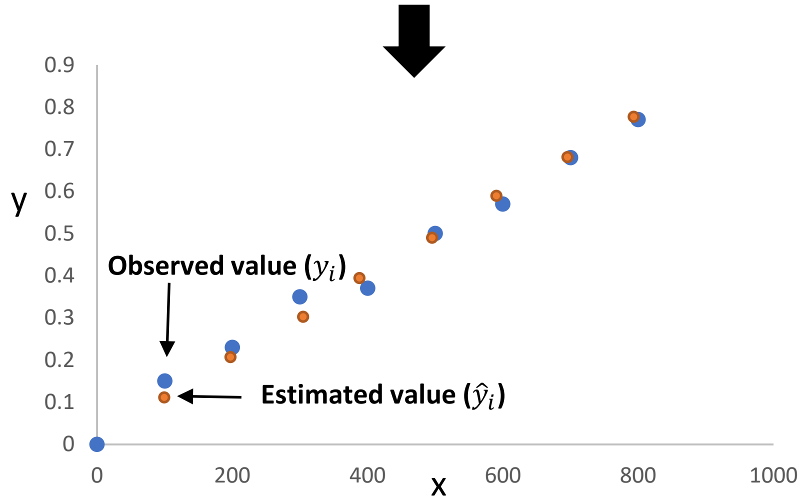
- Find the relationship between the force applied to a beam and the deflection it causes.

x (force in lbs.)	y (deflection in in.)
0	0
100	0.15
200	0.23
300	0.35
400	0.37
500	0.5
600	0.57
700	0.68
800	0.77



Best Fit Line
How do we find such a line?

Least Squares Fit



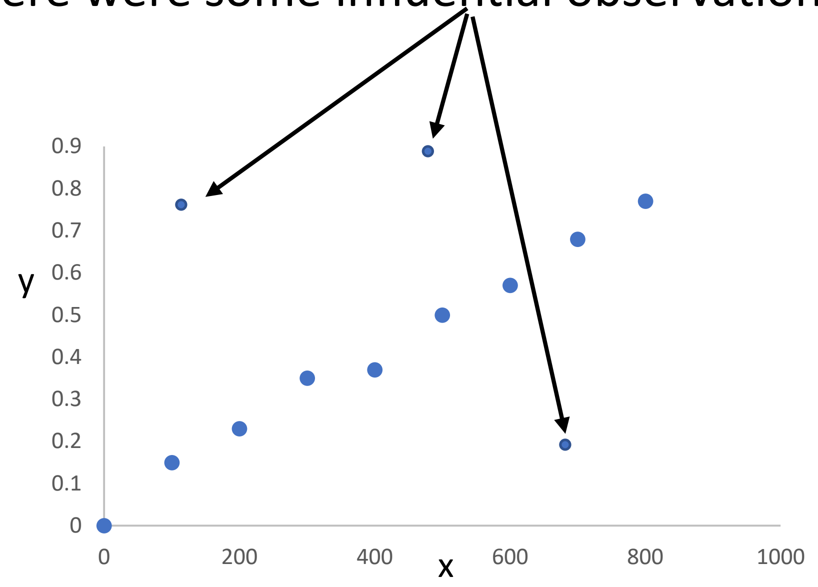
$\hat{y}_i - y_i$ is called ith error or ith residual

Objective:
Minimize $\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

Sum of Squared Residuals (SSR)
or Sum of Squared Errors (SSE)
or **Residual Sum of Squares (RSS)**

Linear Regression

- What will happen if there were some influential observations in your training data set?



Observations that cause significant changes in the parameter estimates are called influential observations

- Linear regression seeks to find estimates that minimize RSS
  - Hence, it will chase the observations that are far away from the overall trend of the majority of the data points

Huber Function

Huber Function

- Linear regression is prone to chasing observations that are away from the overall trend of the majority of the data
  - There are no tuning parameters for multiple regression methods
- One approach to deal with the influential observations is to simply consider taking the absolute residuals

$$\text{Minimize } \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Another approach is to use a robust loss function

$$L(y, \hat{y}) = \begin{cases} (y - \hat{y})^2 & \rightarrow |y - \hat{y}| \leq \alpha \\ |y - \hat{y}| & \rightarrow |y - \hat{y}| > \alpha \end{cases}$$

Huber Function

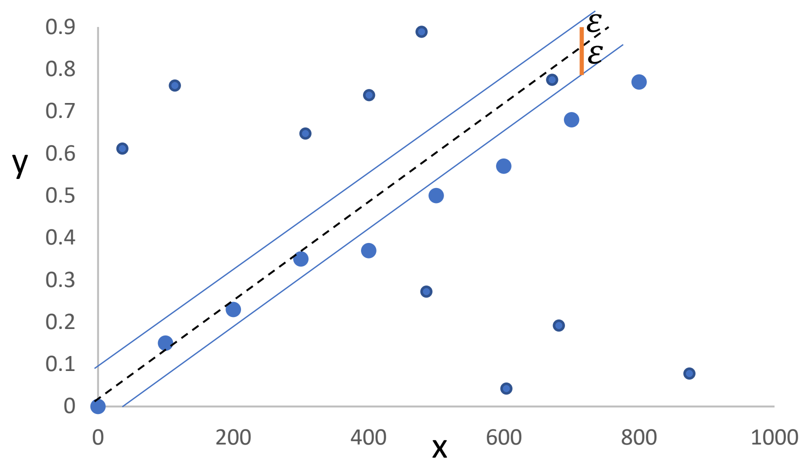
- In other words, <u>Huber Function</u> uses the squared residuals (RSS) when the residuals are small and absolute residuals when the residuals are large
  - Tuning parameter is α

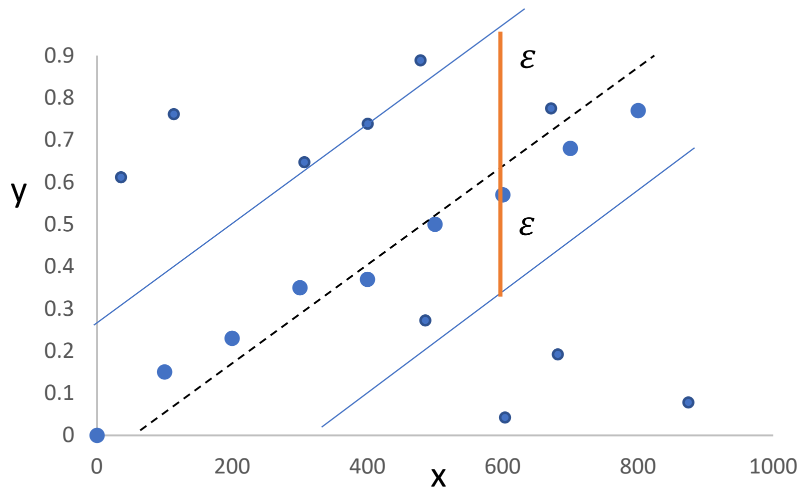
What happens if we keep the value of α very high?

Support Vector Regression

Support Vector Regression (SVR)

- Support Vector Regression is another method to minimize the effect of the influential observations





Support Vector Regression (SVR)

- Support Vector Regression is another method to minimize the effect of the influential observations
- Method:
  - Choose a small value, say, ε
  - Data points with absolute residuals greater than ε contribute to the regression fit
  - Data points whose residuals are small (less than or equal to ε) have no effect on the regression equation
  - Note: this method looks at the absolute residuals and not the squared residuals
    - Softens the impact of the influential observations on the model fit

- The loss function used to estimate the parameters is given as follows:

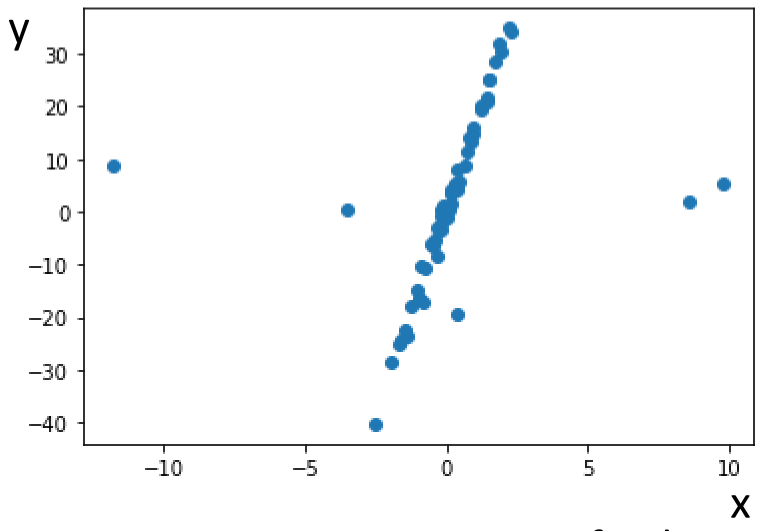
$$\underbrace{C}_{\text{cost penalty}} \sum_{i=1}^{n} \underbrace{L_{\epsilon}}_{\varepsilon\text{-sensitive function}} (y_i - \hat{y}_i) + \sum_{j=1}^{p} \beta_j^2$$

If $(y_i - \hat{y}_i) \le \varepsilon$, then 0

Important:
The penalty is attached to the residuals

Support Vector Regression (SVR)

- Suppose we obtained the following scatter plot between x and y for a given training data set



SVR Loss Function:

$$C \sum_{i=1}^{n} L_\epsilon \ (y_i - \hat{y}_i) + \sum_{j=1}^{p} \beta_j^2$$

What happens if we assign a large cost penalty?

$$\begin{tikzcd} & & \text{increase} \\ & \text{flexibility} \arrow[ur] \arrow[dr] & \\ \text{What happens if we assign a large cost penalty?} \arrow[ur] \arrow[ddr] & & \text{decrease} \\ & & \text{over} \\ & \text{model fit} \arrow[ur] \arrow[dr] & \\ & & \text{under} \end{tikzcd}$$

- In practice, we may want to fix the value of ε and then tune the cost penalty parameter C

inversely proportional

- Import SVR from sklearn.svm module

```
from sklearn.svm import SVR
```

- Call the SVR function

```
model_SVR = SVR(C=1.0, epsilon=0.1, kernel = ‘linear’)
```

Support Vector Regression (SVR)

- Given a new sample, u, we can find the predicted value according to the equation:

$$\beta_0 + \beta_1 u_1 + \beta_2 u_2 + \ldots + \beta_{\mathrm{p}} u_p = 0$$

This can be further expressed as:

- $\beta_0 + \sum_{j=1}^{\mathrm{p}} \beta_{\mathrm{j}}\ u_j$

obtained from the training data points

- $\beta_0 + \sum_{j=1}^{\mathrm{p}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_i x_{ij}\ u_j$

dot product

- $\beta_0 + \sum_{i=1}^{\mathrm{n}} \alpha_i \sum_{\mathrm{j}=1}^{\mathrm{p}} x_{ij}\ u_j$

Kernel function

- $\beta_0 + \sum_{i=1}^{\mathrm{n}} \alpha_i\ K(x_i, u)$

- It is to be noted that we need the training data points to calculate the value for the test sample (predictions)
- We now have a set of unknow parameters α_i
  - How many?
  - What happens to the α values for the data points whose residuals are less than or equal to ε?
- Support Vectors are data points whose value for $\alpha \neq 0$
  - In other words, data points whose residuals are greater than ε
- Sum of cross products will greatly affect the model if the predictor scales are different
  - Hence, it is critical to center and scale the predictors

Support Vector Regression (SVR)

- Given a new sample, u, we can find the predicted value according to the equation:

  $\beta_0 + \beta_1 u_1 + \beta_2 u_2 + \ldots + \beta_\mathrm{p} u_p = 0$

This can be further expressed as:

- $\beta_0 + \sum_{j=1}^{\mathrm{p}} \beta_\mathrm{j}\ u_j$

  obtained from the training data points

- $\beta_0 + \sum_{j=1}^{\mathrm{p}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_i x_{ij}\ u_j$

  dot product

- $\beta_0 + \sum_{i=1}^{\mathrm{n}} \alpha_i \sum_{\mathrm{j}=1}^{\mathrm{p}} x_{ij}\ u_j$

  Kernel function

- $\beta_0 + \sum_{i=1}^{\mathrm{n}} \alpha_i\ K(x_i, u)$

A polynomial kernel of degree d:

$K(x_\mathrm{i}, u) = (\sum_{\mathrm{j}=1}^{\mathrm{p}} x_{ij}\ u_j)^{\mathrm{d}}$

A radial basis kernel:

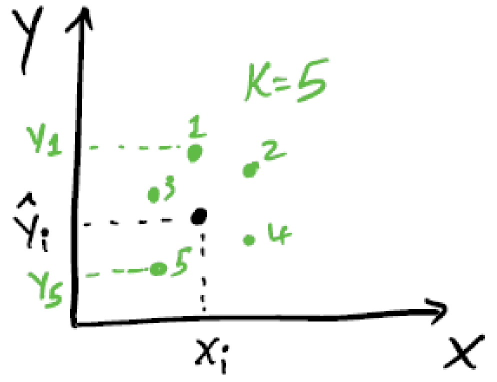
$K(x_\mathrm{i}, u) = \exp(-\gamma \sum_{j=1:p} (x_\mathrm{ij} - u_j)^2)$

K-Nearest Neighbors Regression

K-Nearest Neighbors (KNN) Regression

- Objective is to find the K nearest data points from the new data point and then predict the value of the response variable based on the K response values
- For regression, this value can be the mean of the K response values

$$\hat{y}_i = \frac{1}{k}\sum_{\mathrm{m}=1}^{k} y_m$$



- How do you find the nearest neighbors?
  - One of the popular distance metrics is the Euclidean distance (straight-line distance between 2 data points)

$$\left(\sum_{\mathrm{j}=1}^{\mathrm{p}} (x_{aj} - x_{bj})^2\right)^{1/2}$$

An Example

Training data

y = Health Score	x1 = Height (cm)	x2 =Weight (grams)
1.6	-1.186	-0.707
2.3	-0.074	1.414
3.1	1.260	-0.707

New (Test) Sample

y	x1	x2
?	0.996	1.414

If the predictor data are in different measurements, we must first center and scale the data

$$\left(\sum_{j=1}^{p}(x_{aj}-x_{bj})^{2}\right)^{1/2}$$

	(New Sample - Data Point 1)^2	(New Sample - Data Point 2)^2	(New Sample - Data Point 3)^2
x1	4.76	1.14	0.07
x2	4.50	0.00	4.50
Sum	9.26	1.14	4.57
Sqrt	3.04	1.07	2.14

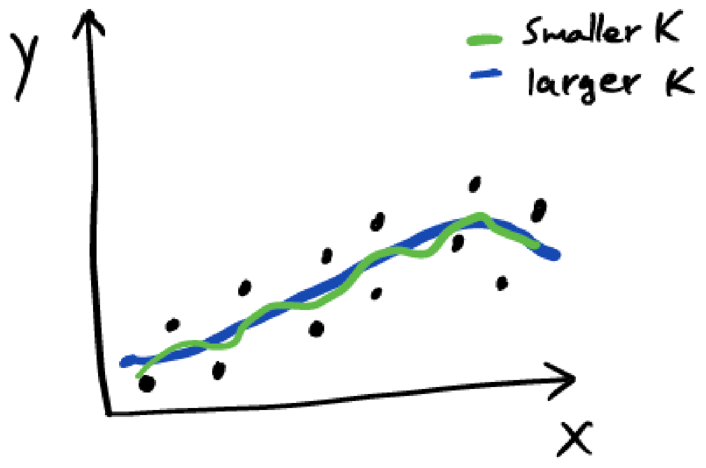
What if the KNN method was weighted?
i.e., 70% weight associated with the nearest neighbor

$$\hat{y}_i = \frac{1}{k}\sum_{m=1}^{k} y_m$$

for K = 2

y	x1	x2
(2.3+3.1)/2 = 2.7	0.996	1.414

KNN Regression



- Smaller value of K will provide more flexibility
  - How does this impact Bias or Variance?
- Larger value of K will reduce flexibility
  - How does this impact Bias or Variance?

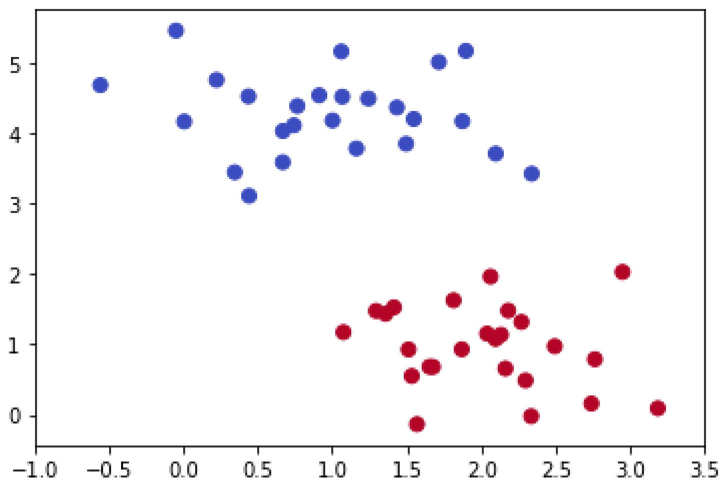
- If one or more predictor values for a sample are missing, the distance cannot be calculated
  - Hence, either the sample may have to be removed or values need to be imputed
- Noisy predictors also contribute to poor response value estimates
- Value of K is determined by resampling methods using the training data set
- Method works well when p is not too large, else performs poorly when p is large (curse of dimensionality)
- Efficient data structures such as a k-dimensional tree representation can help with computational challenges
- KNN Regression does not help with interpretability

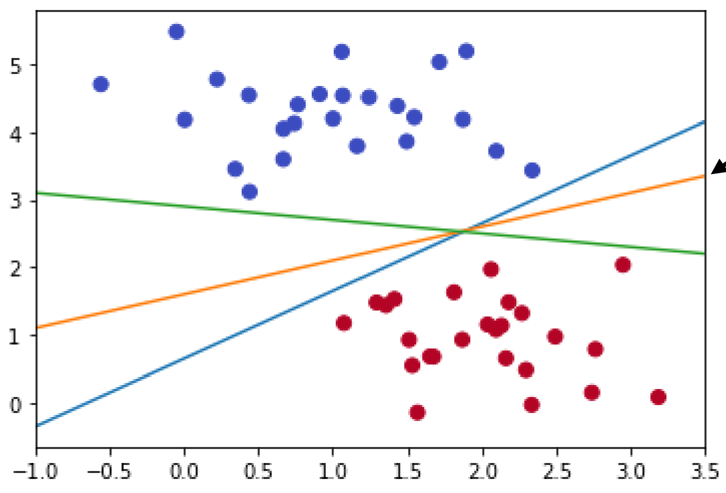
Nonlinear Models

Spring 2020
Instructor: Ankit Shah, Ph.D.

Classifiers with Linear Decision Boundaries

Maximal Margin Classifier





Optimal separating hyperplane:

$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0$

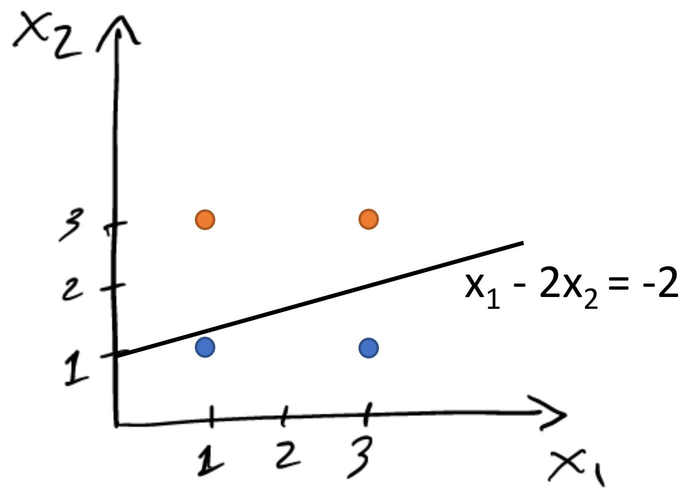
One that is the farthest away from any of the samples in the training data set

Used when classes can be separated well by a linear boundary
i.e., the classes can be separated by a (p-1)-dimensional hyperplane

Code to obtain the above figure in Python:

```
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=50, centers=2,random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='coolwarm’)
```

Maximal Margin Classifier



- Say, we have the following data points (x_1, x_2) in the training data: (1,3), (3,1), (3,3), (1,1)
  Classify them in 2 different classes: -1 and +1
- *Hyperplane:* $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$
  $2 + x_1 - 2x_2 = 0$
- (1,3) -> -ve value -> class -1
- (3,1) -> +ve value -> class +1
- (3,3) -> -ve value -> class -1
- (1,1) -> +ve value -> class +1

- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p > 0$, then it belongs to one side of the p-dimensional space: y = 1
- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p < 0$, then it belongs to another side of the p-dimensional space: y = -1
- Generalizing the above: a separating hyperplane has the following property

  $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$ for (i = 1, …, n)

Classification using a Separating Hyperplane

- If a separating hyperplane exists, then there will be an infinite collection of them
- We can adjust the hyperplane by a little and still separate the classes
- We multiply the β_j by a non-zero constant such that the equation still holds true: $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$

Objective:

- We want to construct a classifier that optimally separates the classes such that <u>the separating hyperplane is the farthest distance from the training observations</u>

How do we do that?:

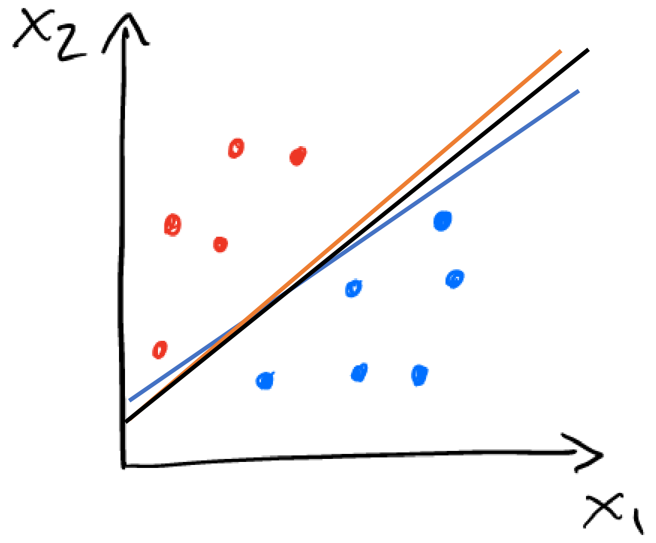
  - We compute the perpendicular distance from each training data point to the given separating hyperplane
  - The smallest such distance is the minimal distance from the observations to the hyperplane – <u>Margin</u>
  - Our objective is to maximize the margin   Max. min. distance

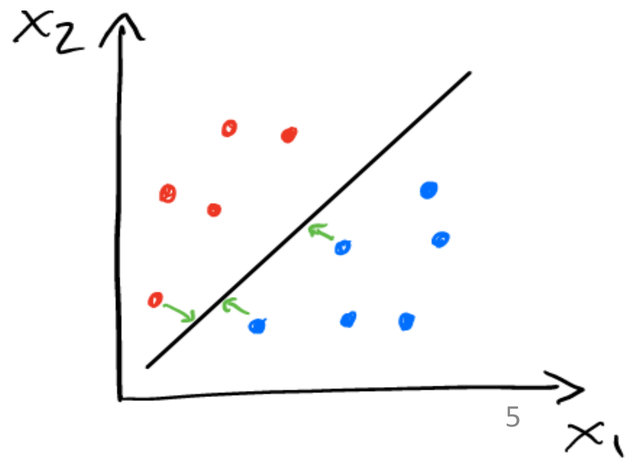
- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

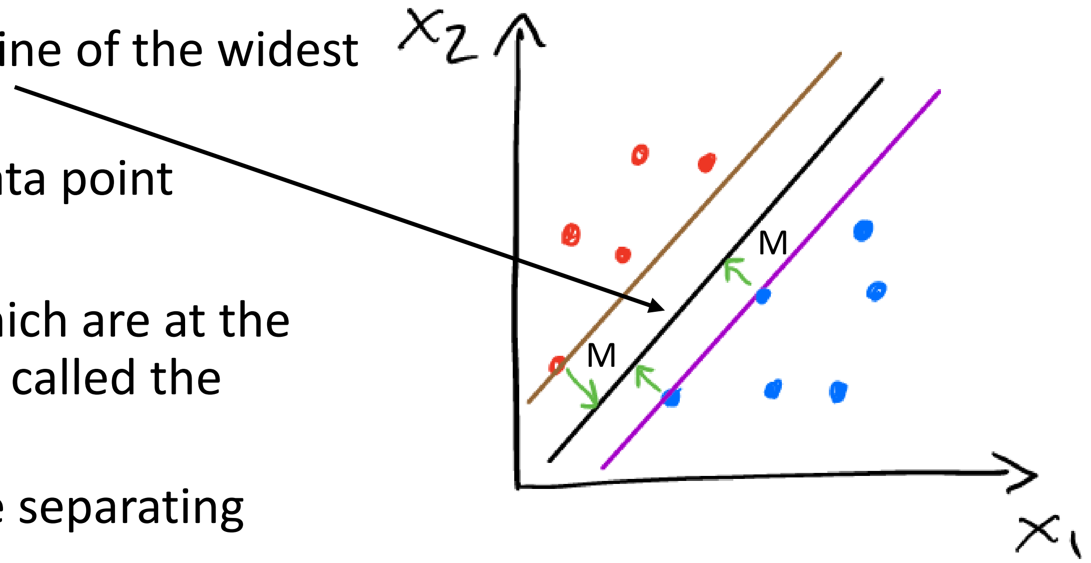
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M$$



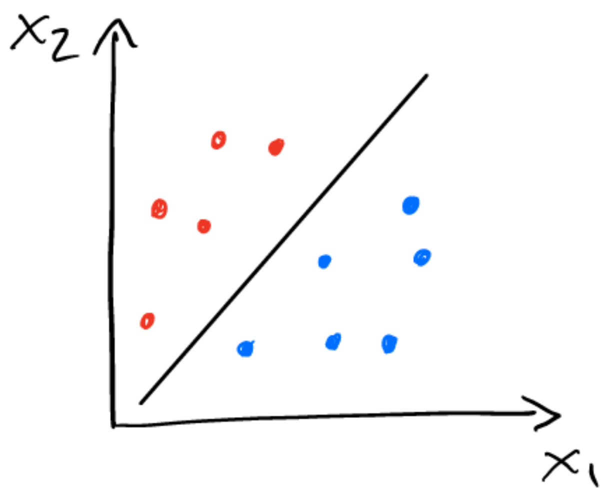


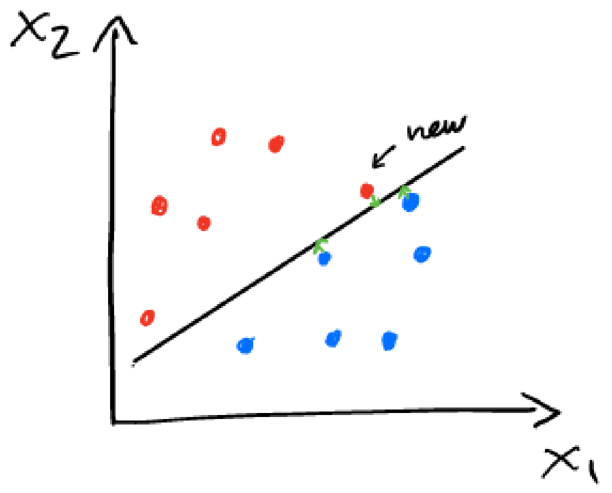
Classification using a Separating Hyperplane

- Think of the maximal margin hyperplane as the mid-line of the widest "slab" that we can insert between two classes
- The value M (smallest distance between a training data point and the hyperplane) is called the margin
- Observations (data points in the training data set) which are at the exact distance M from the separating hyperplane are called the support vectors
- Each data point is at least at a distance of M from the separating hyperplane



Maximal Margin Classifier

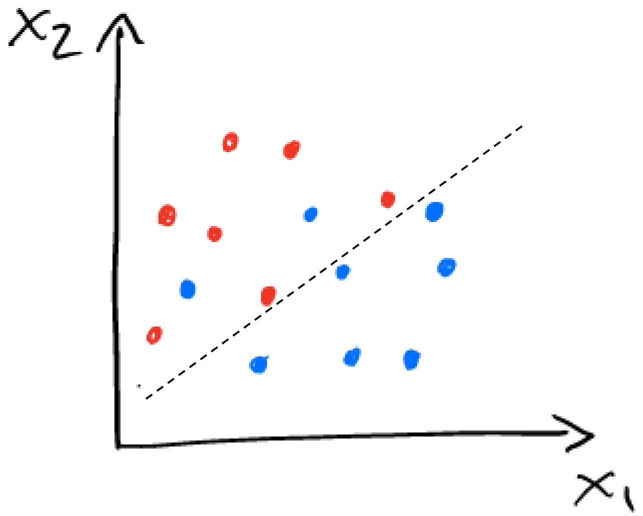


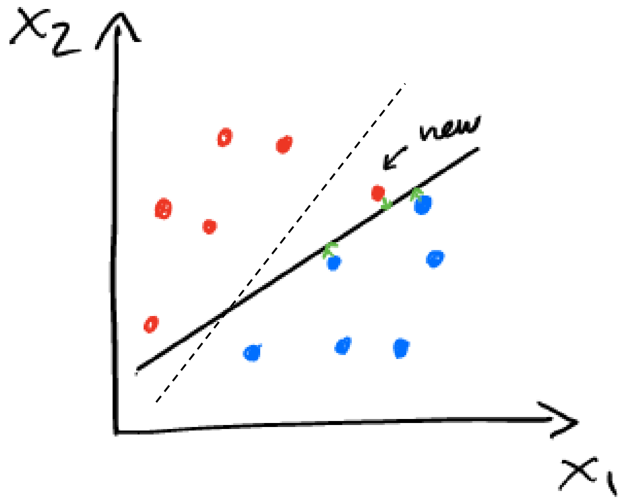


- A small change in the data set can result in a fairly large change in the decision boundary of the maximal margin classifier
- Such sensitivity suggests that the maximal margin classifier overfits the training data
    - i.e., the model is prone to a higher misclassification rate on the testing (unseen) data set
- This motivates a need for a classifier that can be used to produce a more appealing decision boundary

Support Vector Classifier

- Support vector classifier is also called the “soft” margin classifier
- Used when the classes of the training data cannot be perfectly separated by a hyperplane and also when a more appealing decision boundary is needed in which separation is possible

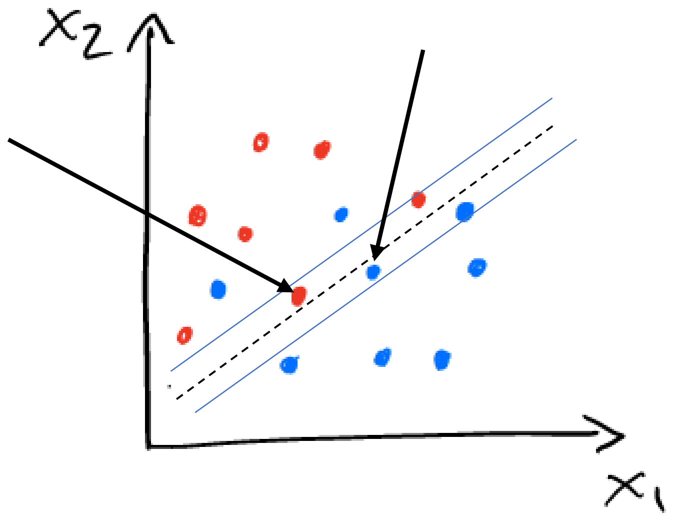




- In the above cases, we may consider a hyperplane that does not separate the 2 classes perfectly
- Our objective is to provide 1) robustness to the model and 2) better classification for majority of the data points

Support Vector Classifier

- As shown in the previous example, we allowed some data points to be on the incorrect side of the hyperplane and also closer to the hyperplane on the correct side but with a value less than the margin
- Hence, we call it a soft margin
    - i.e., it can be violated by some data points to obtain a more robust model



- The support vector classifier has a (p-1)-dimensional hyperplane for the decision boundary and has a soft margin that could be violated

Support Vector Classifier

- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

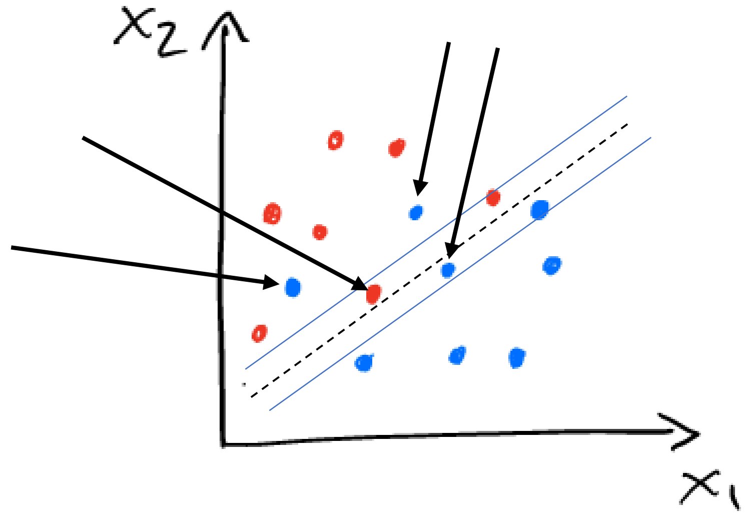
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

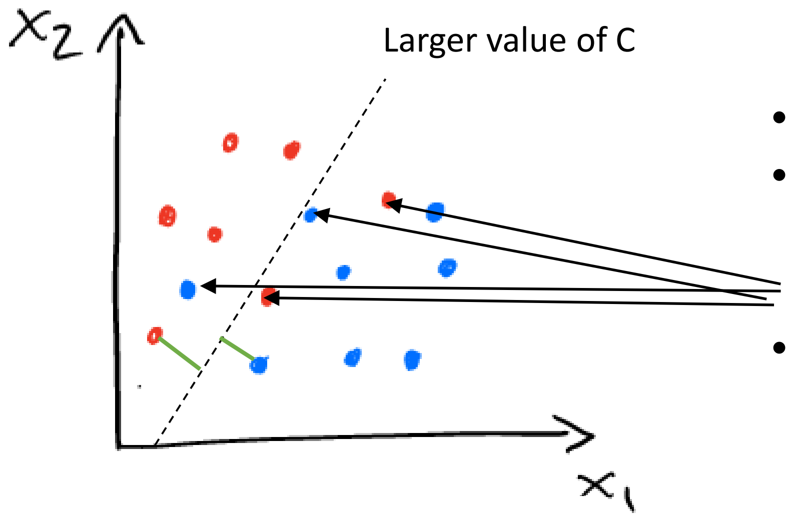
Slack variable

Tuning parameter

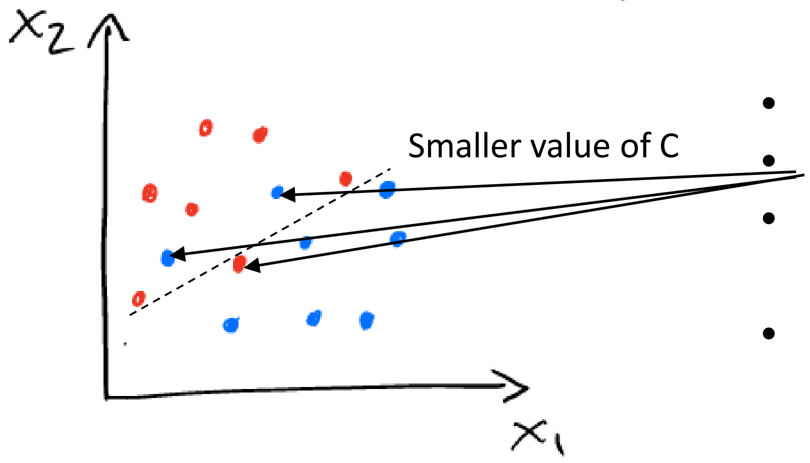


- ε_i allows the data point *i* to be on the wrong side of the margin or the hyperplane
- C determines the number and severity of the violations to the margin and the hyperplane
- Support vectors: All data points with $\varepsilon_i > 0$

Relationship of C with M



- Larger value of C -> larger value of M
- More slack variables come into play and hence more violators (support vectors)
- This is a condition with
  Low variance – High bias



- Smaller value of C -> smaller value of M
- Less number of support vectors
- This is a condition where we are chasing the data points
- Overfit -> Low bias

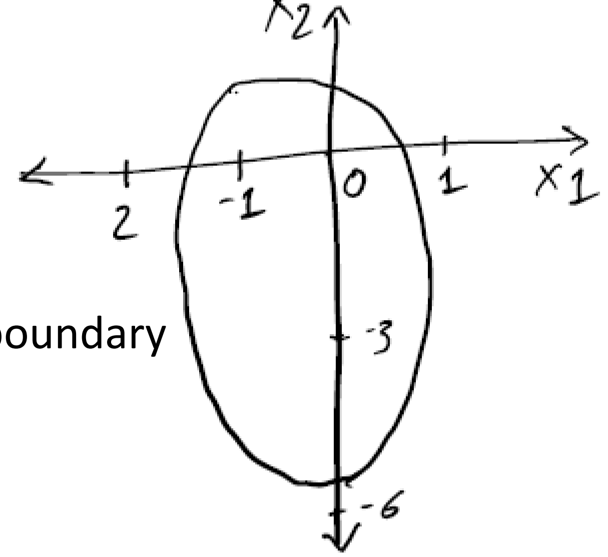
Support Vector Machines

Support Vector Machines

- Suppose we have 2 predictor variables $x_1$and x_2 and we add 2 more predictor variables $x_3 = x_1^2$ and $x_4 = x_2^2$
- Next, we want to apply the support vector classifier and we obtain the decision boundary that satisfies the condition:
- $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_\mathrm{p} x_p = 0$

$\Big\downarrow$ Obtained after fitting on a training data set

- -0.13 + 0.7 x_1 + 0.5 x_2 + 0.5 x_1^2 + 0.1 x_2^2 = 0 Note that $\sum_j \beta_j^2 = 1$



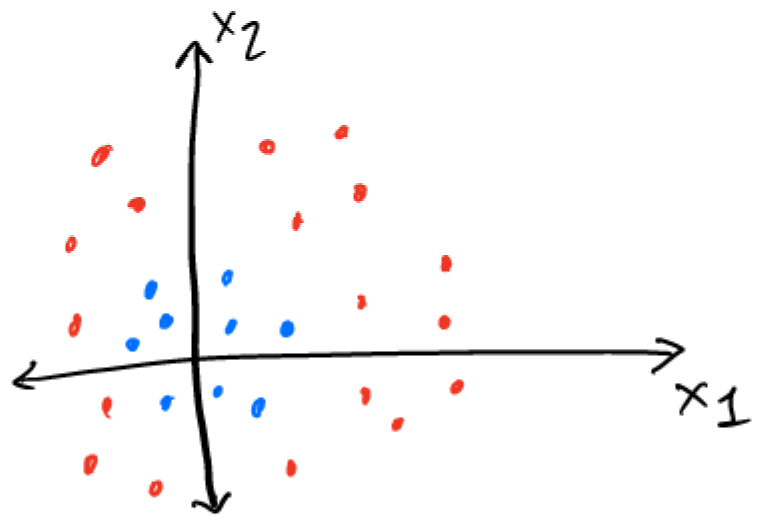
Elliptical decision boundary

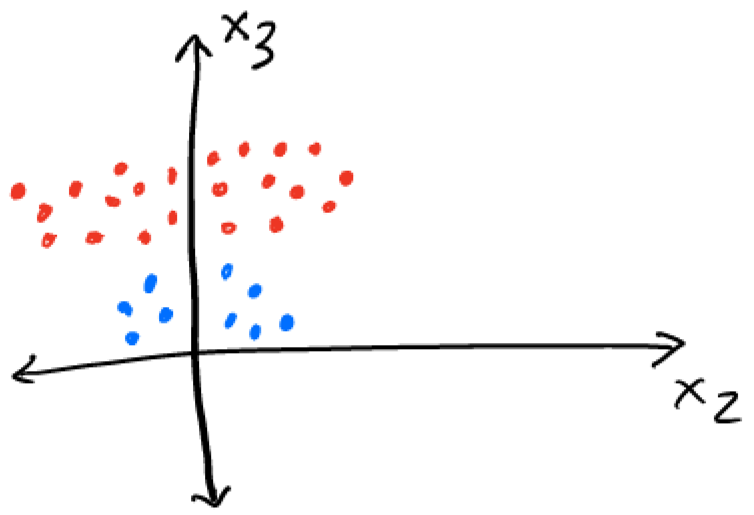
- From the above we can arrive at:
- 0.5(x_1^2+1.4 x_1) + 0.1(x_2^2+5 x_2) = (1-0.87)
- 0.5 (x_1^2+1.4 x_1+0.49) + 0.1 (x_2^2+5 x_2+6.25) = 1
- 0.5(x_1+0.7)^2 + 0.1(x_2+2.5)^2 = 1 $\Longleftarrow$ Equation of an ellipse

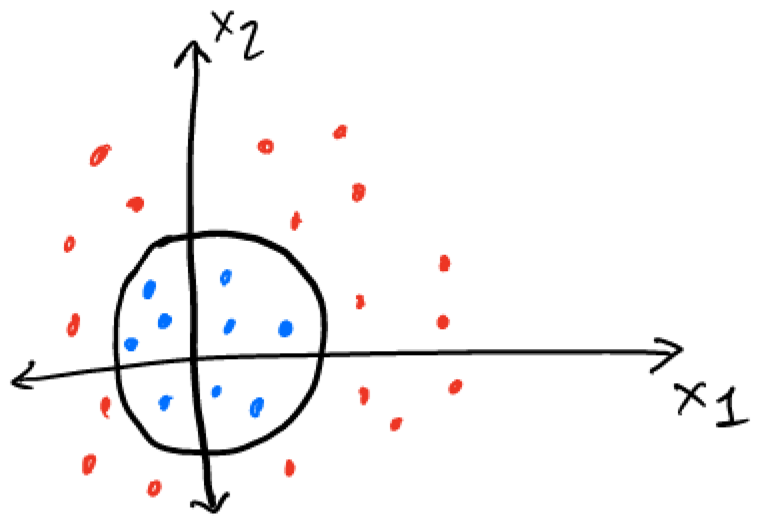
The Support Vector Machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way (using *kernels*)

- Kernel method is an efficient computational approach for achieving the above

Support Vector Machines







- There is no separating hyperplane that can segregate the 2 classes
- Apply a transformation (kernel function) that maps the original data points into a higher-dimensional space in which they become separable
- Upon transforming it back to the original plane, we obtain the nonlinear circular boundary as shown above

Support Vector Machines

- We can obtain the solution to the support vector classifier by simply taking inner products of the data points
- The inner product of vectors x_1 and x_2 can be represented as $\langle x_1, x_2 \rangle = \sum_{j=1:p} x_{1j} x_{2j}$
- The decision rule for the support vector classifier can then be expressed as:

$$f(x) = \beta_{0+} \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

α_i are the parameters that need to be estimated along with β_0

- α_i values are non-zero only for the support vectors

  What does n represent?

- In order to obtain the value of $f(x)$, we need to compute the inner product between the new point x and each of the training points x_i Measure of similarity
- We then classify the new point x as class 1 if $f(x) > 0$ and class -1 if $f(x) < 0$

Support Vector Machines

- The inner product of vectors x_1 and x_2: $\langle x_1, x_2 \rangle = \sum_{j=1:p} x_{1j} x_{2j}$
- We can generalize the above as $K(x_i, x_{i'})$, where K is called a *kernel*

For example, a linear kernel will be:

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

And a polynomial kernel of degree d will be:

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

When the Support Vector Classifier is used with such a nonlinear kernel, then the resulting classifier is called a Support Vector Machine (SVM)

One of the popular kernels is the radial kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1:p} (x_{ij} - x_{i'j})^2\right)$$

The tuning parameter γ controls how quickly a training data point's influence decreases with increasing distance to the test data point

Support Vector Machines

$p=3$

$x_1 = (x_{11}, x_{12}, x_{13})$

$x_2 = (x_{21}, x_{22}, x_{23})$

polynomial degree $d=2$

$f_1 =$
$(x_{11}x_{11}, x_{11}x_{12}, x_{11}x_{13},$
$x_{12}x_{11}, x_{12}x_{12}, x_{12}x_{13},$
$x_{13}x_{11}, x_{13}x_{12}, x_{13}x_{13})$

$f_2 =$
$(x_{21}x_{21}, x_{21}x_{22}, x_{21}x_{23},$
$x_{22}x_{21}, x_{22}x_{22}, x_{22}x_{23},$
$x_{23}x_{21}, x_{23}x_{22}, x_{23}x_{23})$

If we had used the kernel with $d=2$,

$$K(x_1, x_2) = \left((1,2,3)\begin{pmatrix}4,\\5,\\6\end{pmatrix}\right)^2$$
$$= (4+10+18)^2$$
$$= (32)^2$$
$$= 1024$$

$x_1 = (1,2,3)$

$x_2 = (4,5,6)$

$f_1 = (1,2,3,$
$2,4,6,$
$3,6,9)$

$f_2 = (16,20,24,$
$20,25,30,$
$24,30,36)$

Inner product

$$= 16+40+72+40+100 + \dots + 324$$
$$= 1024$$

Please go through the coding examples (Python) from the recording