Big Data Analysis using Machine Learning Techniques

Nirosha Rathnayake

January 30, 2018

Background

- Statistics for Big Data (Summer 2017, University of Washington, Seattle)
  1. Reproducible Research for Biomedical Data
  2. Statistical Maching Learning - Supervised learning
  3. Statistical Maching Learning - Unsupervised learning

Content

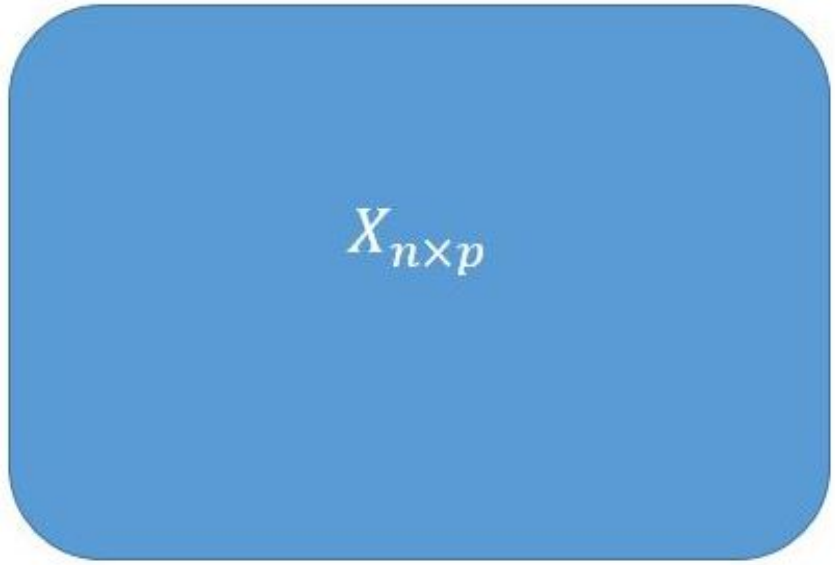
- Big Data and high dimensional data, examples
- Model Evaluation Methods (R^2, training error, test error)
- Estimation of Test Error
- Supervised learning - Introduction
  - Regression with Big Data
  - Classification with Big Data
    - Support Vector Machines (SVM)
- Deep learning

Big Data - High Dimensional Data

Data matrix: n number of observations and p number of variables(features).

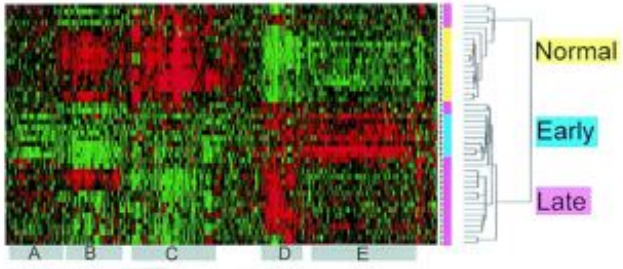
Big Data - High Dimensional Data

Data matrix: n number of observations and p number of variables(features).



When $n \approx p$ or $p >>> n$, the data is called High Dimensional Data.

Examples of Big Data - High Dimensional Data



	Movie rating				
	Alex	Mike	Nancy	Amy	.....
Titanic	5	2	3	5	.....
Harry potter	3	4	5	3	.....
Star wars	3	?	4	5	.....
Transformers	5	4	3	?	.....
Avengers	5	3	2	3	.....

Why High Dimensional Data matters?

- Most of the cases, we have larger number of variables than number of observations.
- Classical statistical techniques have a high risk of overfitting, false positives, and more.
- Our goal is to
  - Predict risk of diabetes based on DNA sequence data ($n = 1000$ patients, $p = 100000$)
  - Predict risk of developing cancer considering gene expression data (hundreds of thousands of genes(features))
  - Identify genes whose expression is associated with survival time ($n = 300$ cancer patients, $p = 20000$)
  - Predict future sale prices on homes (Ex:Zestimate zillow home prices, price \$1,200,000)
- Statistical machine learning techniques are very important in High Dimensional Data analysis.

Statistical Machine Learning

Two main areas in statistical machine learning

1. Supervised Learning: A data set (X) is used to predict or detect association with a response y.

- Regression (For a quantitative response)
- Classification (For a categorical response)

Statistical Machine Learning

Two main areas in statistical machine learning

1. Supervised Learning: A data set (X) is used to predict or detect association with a response y.

- Regression (For a quantitative response)
- Classification (For a categorical response)

2. Unsupervised Learning: No response variable to be considered. Discover patterns/signals in X or detect associations within X.

- Dimension reduction
- Clustering

Statistical Machine Learning

Two main areas in statistical machine learning

1. Supervised Learning: A data set (X) is used to predict or detect association with a response y.

   - Regression (For a quantitative response)
   - Classification (For a categorical response)

2. Unsupervised Learning: No response variable to be considered. Discover patterns/signals in X or detect associations within X.

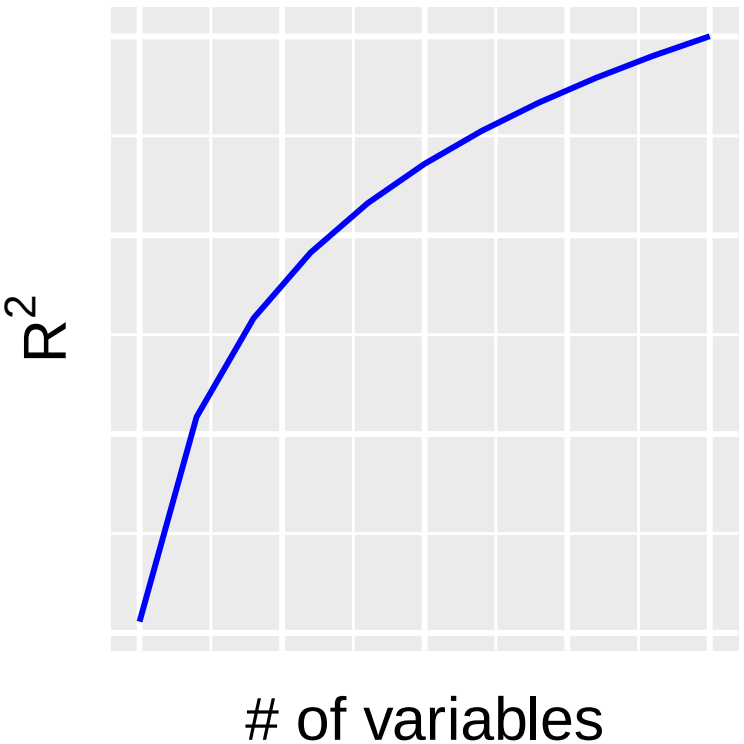
   - Dimension reduction
   - Clustering

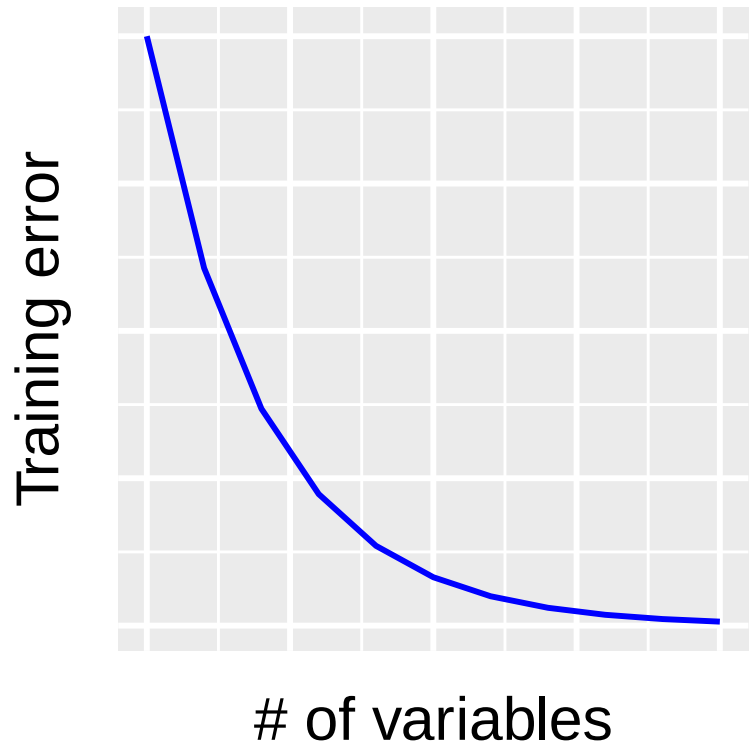
Classification and Regression Trees (CART) : Method of non-linear modeling

Text book: "An Introduction to Statistical Learning"
`http://www-bcf.usc.edu/~gareth/ISL/`

Training Error and Test Error

- When we fit a model, we use a training set of observations and evaluate the model using training error (MSE).
- Trainig error and R^2 are not good choices to evaluate the model performance, because when we have more variables, R^2 increases and training error decreases, the model might work perfectly only on training data.





Predictions, Training Error and Test Error

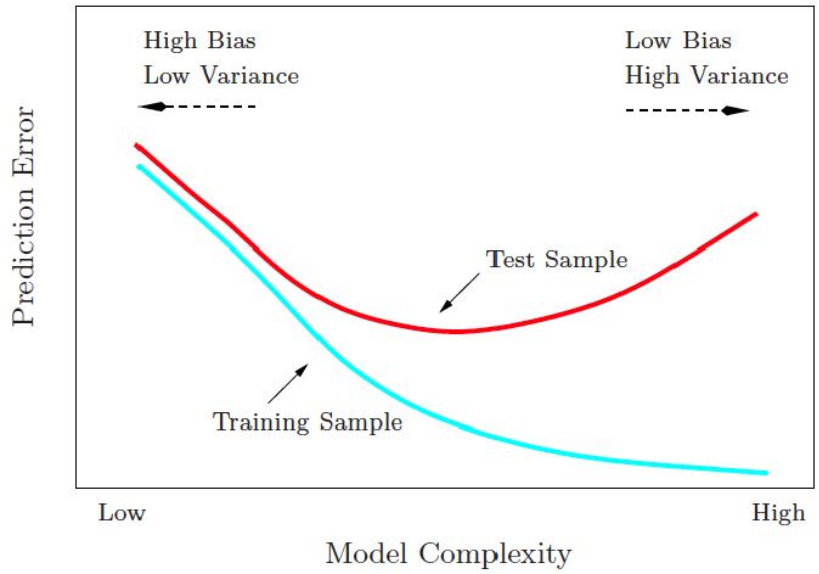
What we should consider to evaluate the model performance? Test error

- Because, we are going to use the model to predict on some data(test data) that did not use to fit the model.
- So, we should consider test error to evaluate the model performance.

$$\text{Mean Sum Squared Test Error} = \frac{1}{N}\sum_{i=1}^{N}(y_i^{test} - \hat{y}_i^{test})^2$$

where $i = 1, ..., N$ number of test observations.

Training Error and Test Error



- Our goal is to obtain a not-too complex model that performs well on predictions.

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}$$

Estimating test error

- The test error is estimated by three methods.
  1. Validation set approach
  2. Leave-one-out cross-validation
  3. K-fold cross-validation

1. Validation set approach

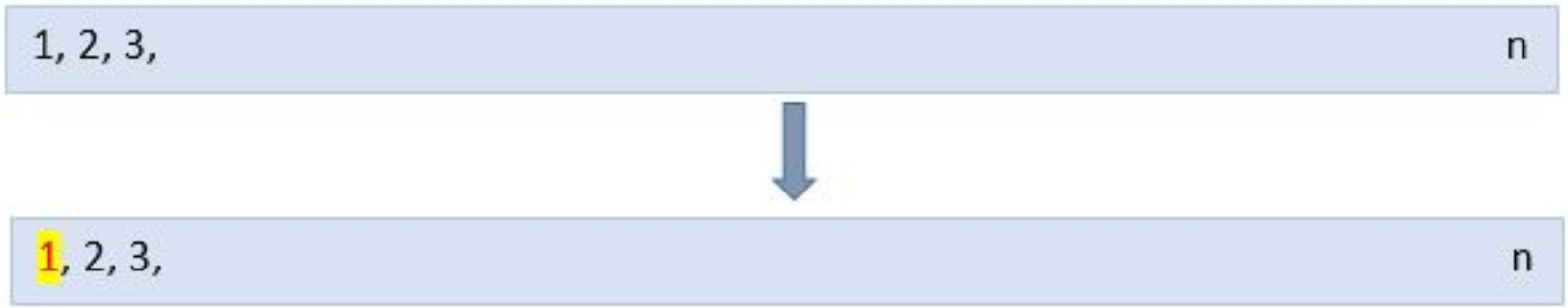
1. Randomly split n observations into two sets of approximately equal size
2. Train on one set (training set) and evaluate perfomance on the other set(test set or validation set).
3. compute validation set error for i^{th} observation
   $e_i = (y_i - X_i^T \hat{\beta}_{(train)})^2$
4. Calculate the total test set error by summing $e_i, i = 1, ..., n_t$.
5. Repeat step $1 - 4$.
6. Select the model that has the lowest total test error.

2. Leave-one-out cross-validation

- Fit n models, each model is on n-1 number of observations.
- Evaluate each model on the left-out observation.

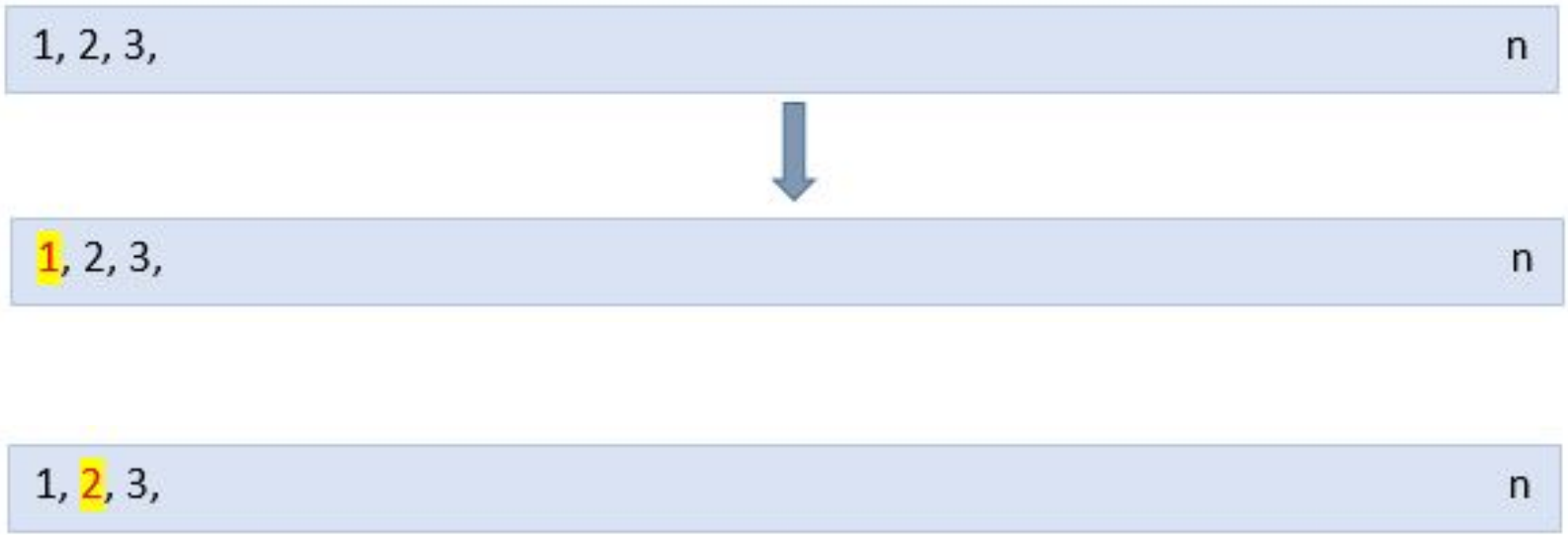
2. Leave-one-out cross-validation

- Fit n models, each model is on n-1 number of observations.
- Evaluate each model on the left-out observation.



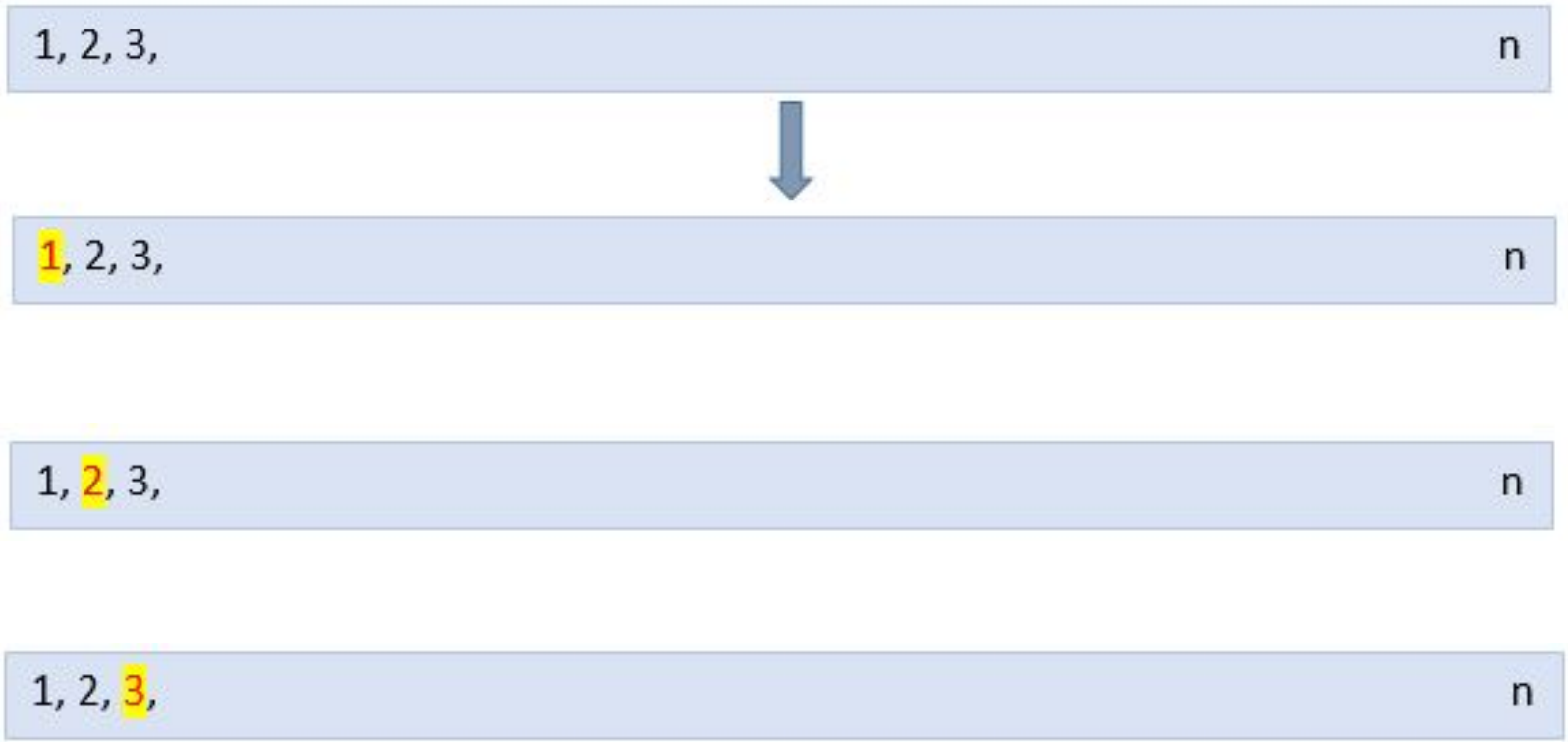
2. Leave-one-out cross-validation

- Fit n models, each model is on n-1 number of observations.
- Evaluate each model on the left-out observation.



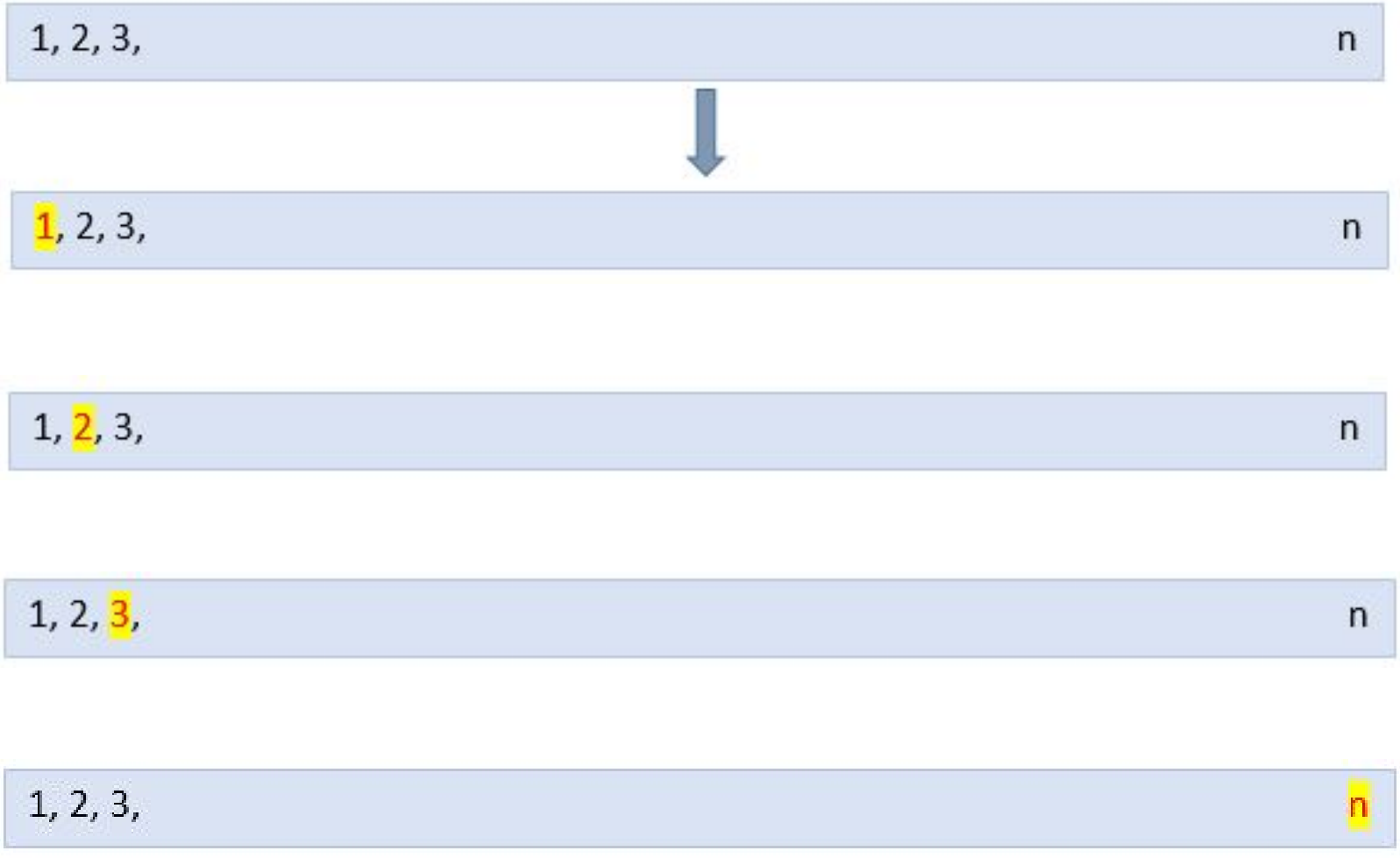
2. Leave-one-out cross-validation

- Fit n models, each model is on n-1 number of observations.
- Evaluate each model on the left-out observation.



2. Leave-one-out cross-validation

- Fit n models, each model is on n-1 number of observations.
- Evaluate each model on the left-out observation.



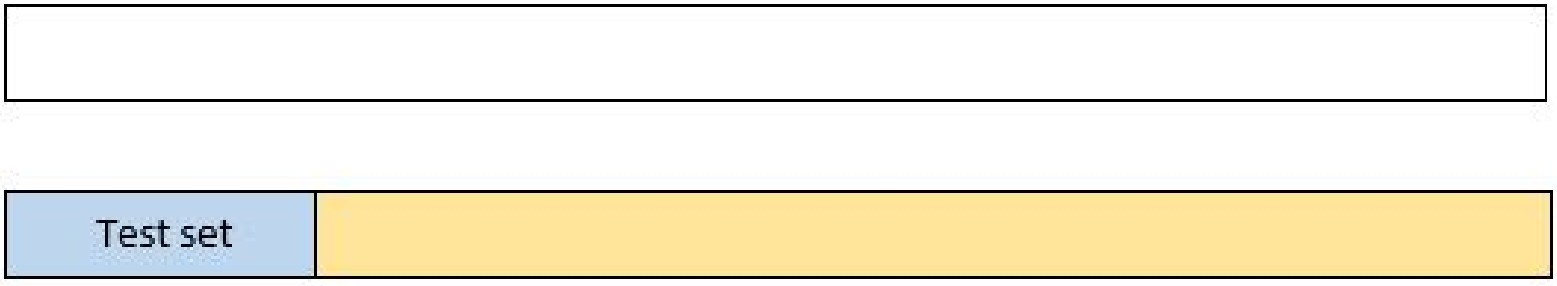
2. Leave-one-out cross-validation

- Steps
  - For $i = 1, ..., n$, fit the model using observations $1, ..., i-1, i+1, ..., n$.
  - compute total test error $e = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - x_i^T \hat{\beta}_{(i)})$
  - Select the model (out of n models) that has the lowest test error.

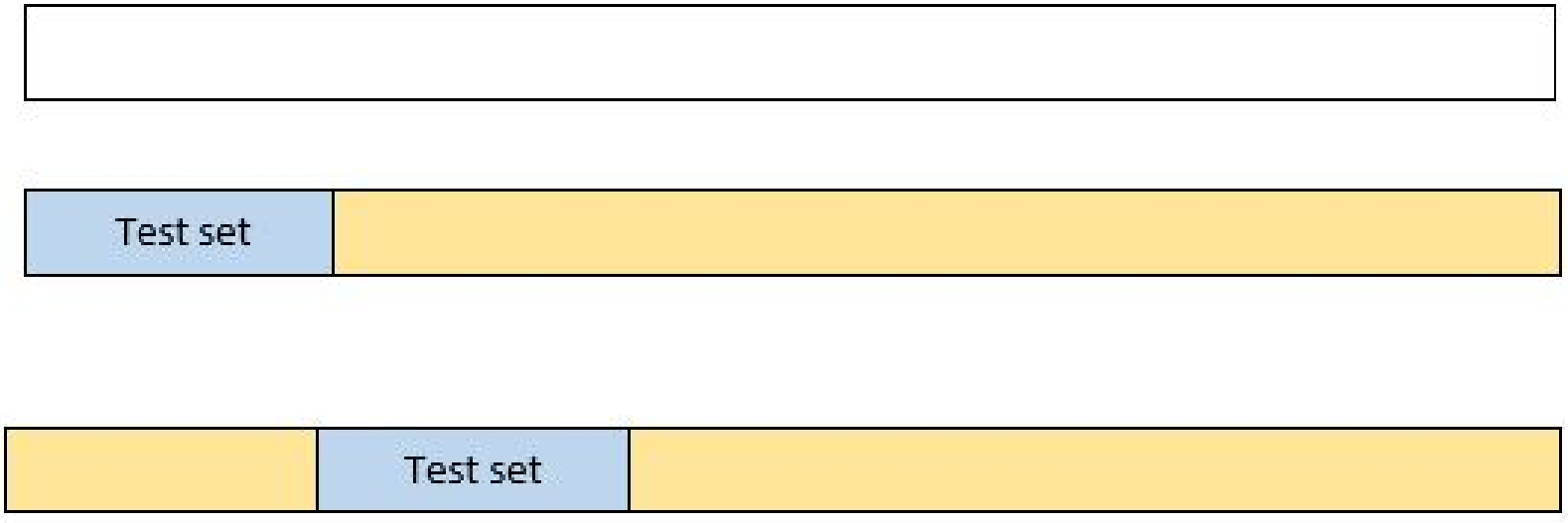
3. K-fold cross-validation

- Randomly split observations into K folds.
- Fit the model for each fold using the observations not in that fold
- We will have k validation set errors(e_k) for each fold.
- Get the total CV error by summing over $e_k, k = 1, ..., K$
- K-fold cross-validation provides better results than Leave-one-out cross-validation.

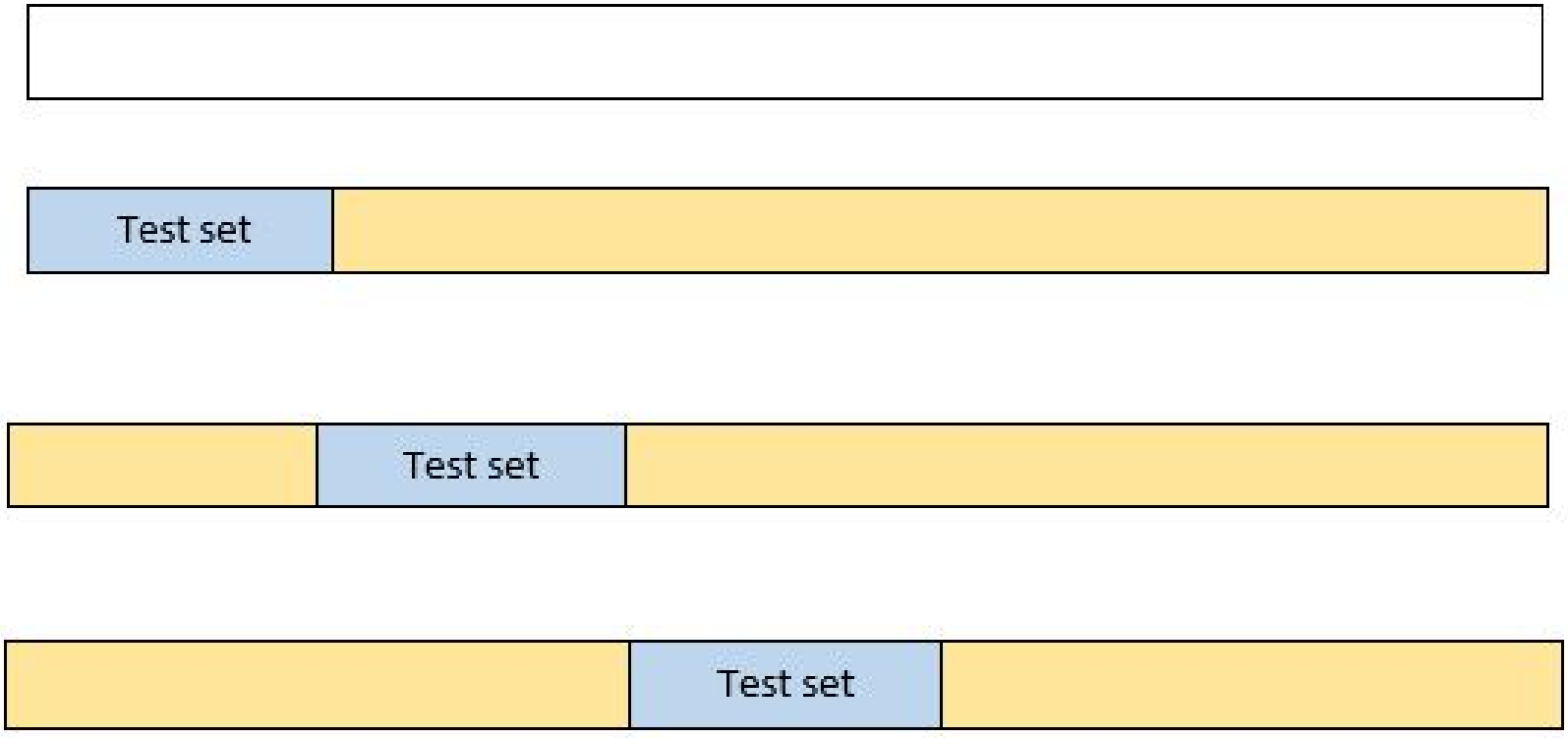
3. 5-fold cross-validation



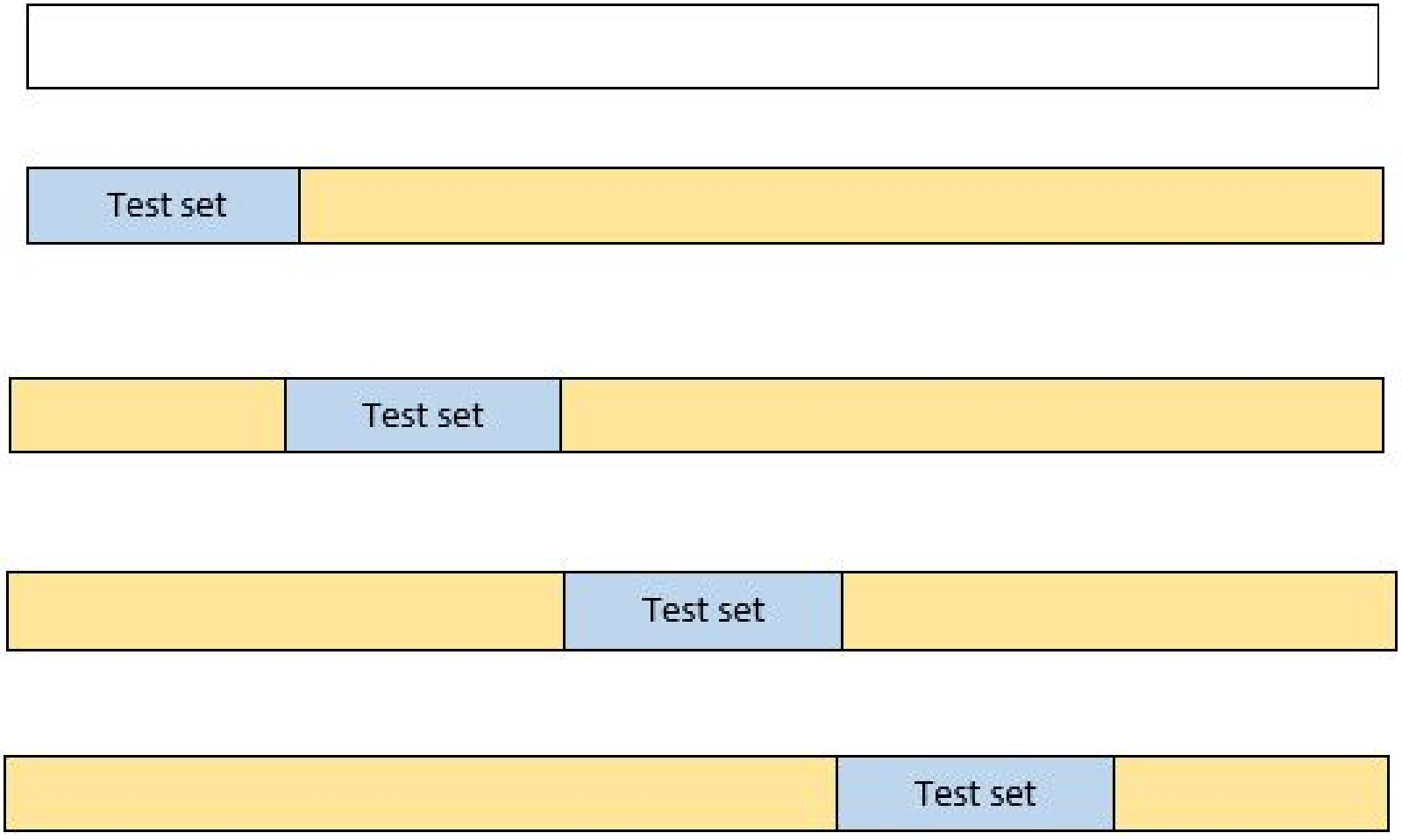
3. 5-fold cross-validation



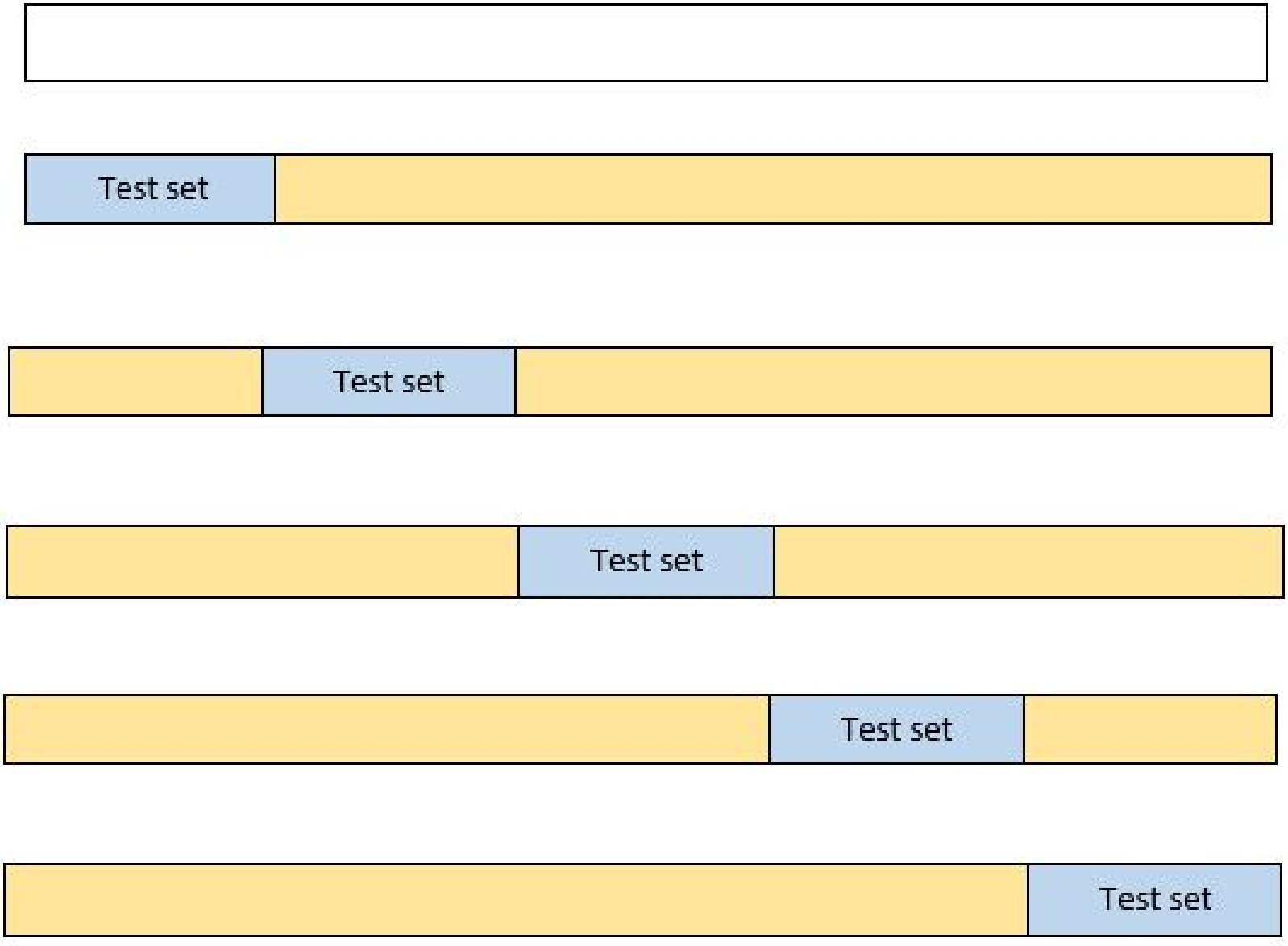
3. 5-fold cross-validation



3. 5-fold cross-validation



3. 5-fold cross-validation



Supervised Learning

Our goal is to obtain a model to predict the response (y) or to find any association between X and y.

1. Regression with big data : When the response is continuous.

Supervised Learning

2. Classification with Big Data - when the response variables are categoricle or qualitative.

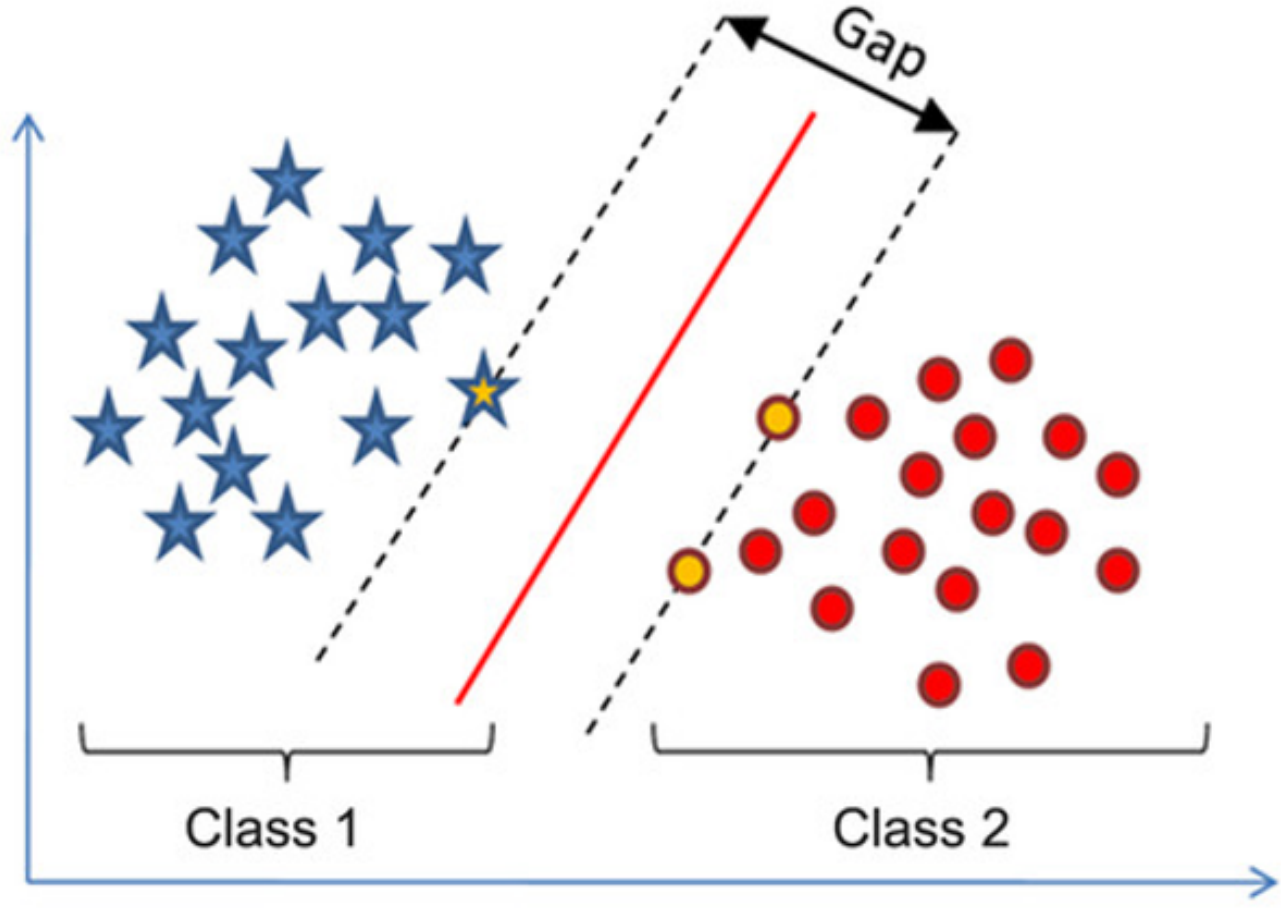
Supervised Learning

2. Classification with Big Data - when the response variables are categoricle or qualitative.

- Classification techniques
  1. K-nearest neighbors
  2. Logistic regression in High Dimensions
     - 2.1 Variable pre selection
     - 2.2 Forward stepwise logistic regression
     - 2.3 Ridge logistic regression
     - 2.4 Lasso logistic regression
     - 2.5 PC logistic regression
  3. Discriminant methods (Classifier that relies on Bayes rule)
  4. Support vector machines (SVM)

Support Vector Machines (SVM)

For a binary response variable (Blue stars and red circles), we try to classify these two groups using a separating hyperplane.

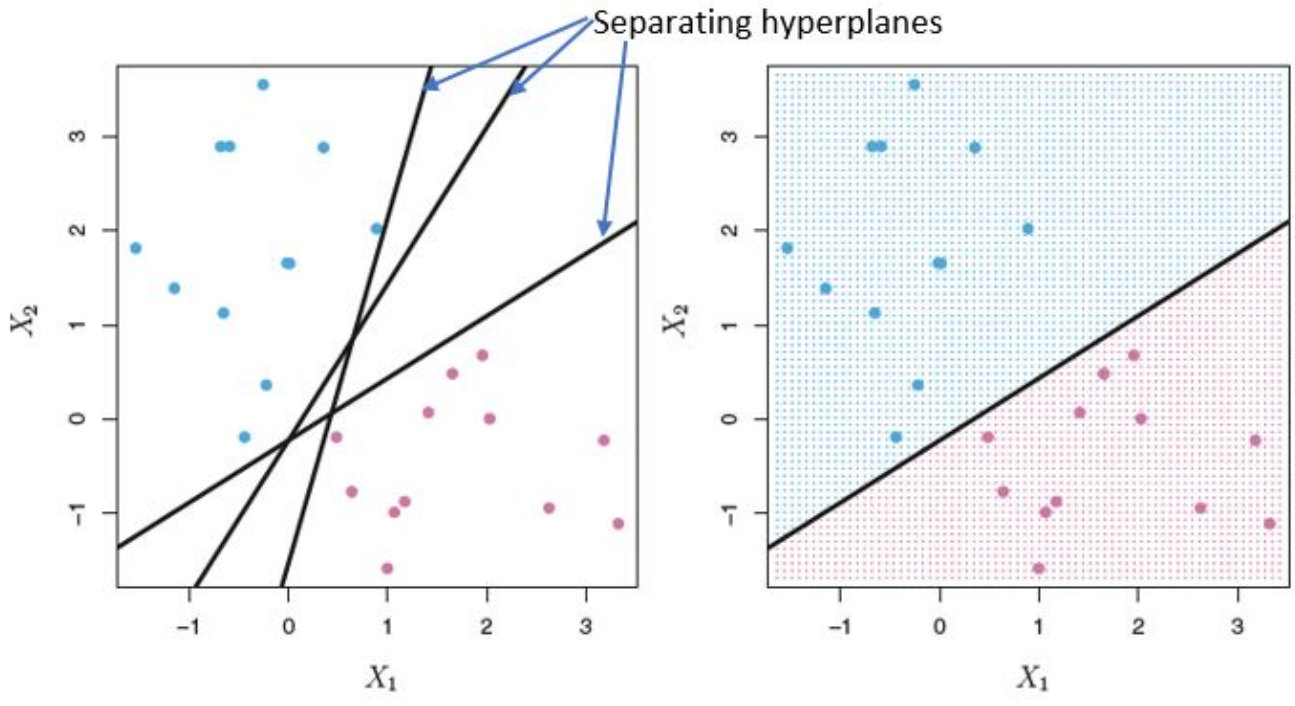


Support Vector Machines (SVM)

- The SVM algorithm was invented by **Vladimir Vapnik** and **Alexey Ya. Chervonenkis** in 1963.
- **Vladimir Vapnik** was one of the main developers in theory of statistical learning.
- SVM was developed in the computer sceince society in the 1990s by suggesting non-linear classifiers.
- Similar to logistic regression, but SVM often performs well compared to logistic regression.

Seperating Hyperplanes

- For a binary response variable



- We want to find the best hyperplane to seperate blue dots and red dots, is called a Seperating Hyperplane or Classifier.

Seperating Hyperplanes

- In 2D, a hyperplane is a line, defined by

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 ==> \beta^* X + \beta_0 = 0$$

where $\beta^* = [\beta_1, \beta_2]^T$ and $X = [X_1, X_2]$

Seperating Hyperplanes

- In 2D, a hyperplane is a line, defined by

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 ==> \beta^* X + \beta_0 = 0$$

where $\beta^* = [\beta_1, \beta_2]^T$ and $X = [X_1, X_2]$

- In $p-$dimensional space, a hyperplane is a $p-$**dimensional hyperplane**.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0$$

Seperating Hyperplanes

- In 2D, a hyperplane is a line, defined by

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 ==> \beta^* X + \beta_0 = 0$$

where $\beta^* = [\beta_1, \beta_2]^T$ and $X = [X_1, X_2]$

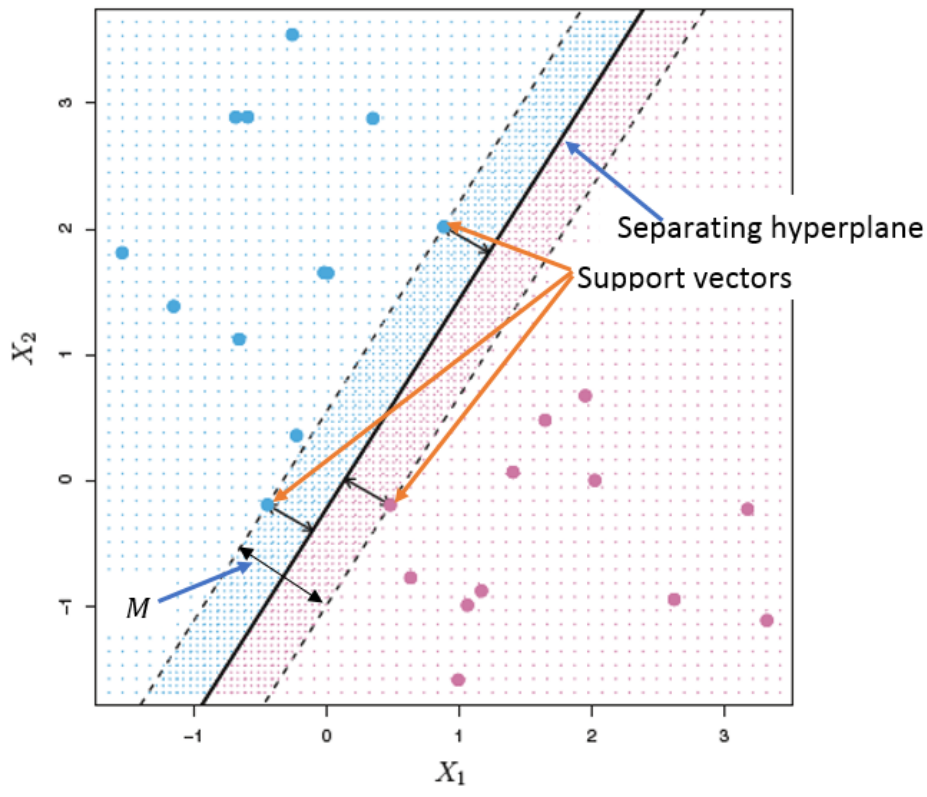
- In $p-$dimensional space, a hyperplane is a $p-$**dimensional hyperplane**.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0$$

- How we are going to find this best separating hyperplane ?

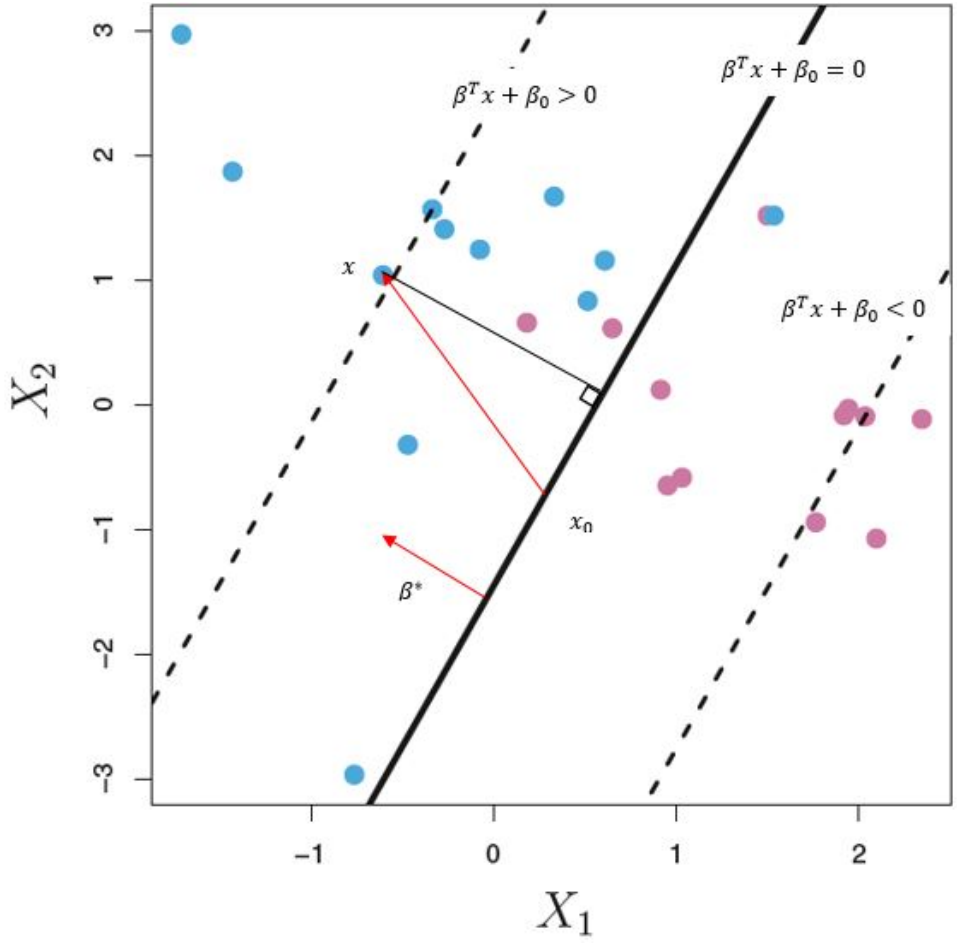
Maximal Seperating Hyperplane

- We are trying to separate these 2 groups as much as possible by increasing the distance between the 2 groups.
- The hyperplane with the largest margin is called the Maximal Seperating Hyperplane or Maximal Margin Classifier.



Computing Maximal Margin Classifier

- If we try to keep the margin as much as larger such a way that it provides a better seperation on the training data, hence it will provide a better classification on the test data.



Computing Maximal Margin Classifier

Decision rule: We try to obtain a maximal margin classifier on the training data, so that we can obtain a better classification on the test data.

If $x_i^T \beta + \beta_0 > 0$ then the new point will be in $y_i = 1$ group

if $x_i^T \beta + \beta_0 < 0$ then the new point will be in $y_i = -1$ group

Where $\beta = [\beta_1, \beta_2]$ and $x = [x_1, x_2]^T$

Computing Maximal Margin Classifier

- Now, we set $\beta^T x_i + \beta_0 = 1$, then

$$x_i^T \beta + \beta_0 > 1 \; if \; y_i = 1 ==> y_i(x_i^T \beta + \beta_0) > 1$$
$$x_i^T \beta + \beta_0 < -1 \; if \; y_i = -1 ==> y_i(x_i^T \beta + \beta_0) > 1$$

- We try to estimate the classifier, keeping all the points at least a signed M distance from the classifier. i.e. $y_i(x_i^T \beta + \beta_0) > M$

Computing Maximal Margin Classifier

we can write the optimization problem in order to maximize the margin (M)

$$\underset{\beta,\beta_0,||\beta||=1}{maximize}\ M$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1, ..., N$$

Computing Maximal Margin Classifier

we can write the optimization problem in order to maximize the margin (M)

$$\underset{\beta,\beta_0,||\beta||=1}{\textit{maximize}}\ M$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1, ..., N$$

Removing the condition $||\beta|| = 1$ and modifying the constraint,

$$\underset{\beta,\beta_0}{\textit{maximize}}\ M$$
$$\text{subject to } \frac{1}{||\beta||} y_i(x_i^T\beta + \beta_0) \geq M, i = 1, ..., N$$

Computing Maximal Margin Classifier

- The signed distance from any point x_i to the hyperplane is given as ${\beta^*}^T(x_i - x_0) = \frac{1}{||\beta||}(\beta^T x_i + \beta_0)$, where β^* is the normal vector to seperating hyperplane.
- Now, since $\beta^T x_i + \beta_0 = 1$ the margin for $y = 1$ is $\frac{1}{||\beta||}$. Maximizing M is equivalent to minimizing $||\beta||$, hence

$$\underset{\beta,\beta_0}{minimize} \frac{1}{2}||\beta||^2$$

$$\text{subject to} \frac{1}{||\beta||} y_i(x_i^T \beta + \beta_0) \geq 1, i = 1, ..., N$$

Computing Maximal Margin Classifier

- Using Lagrange function,

$$L = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1]$$

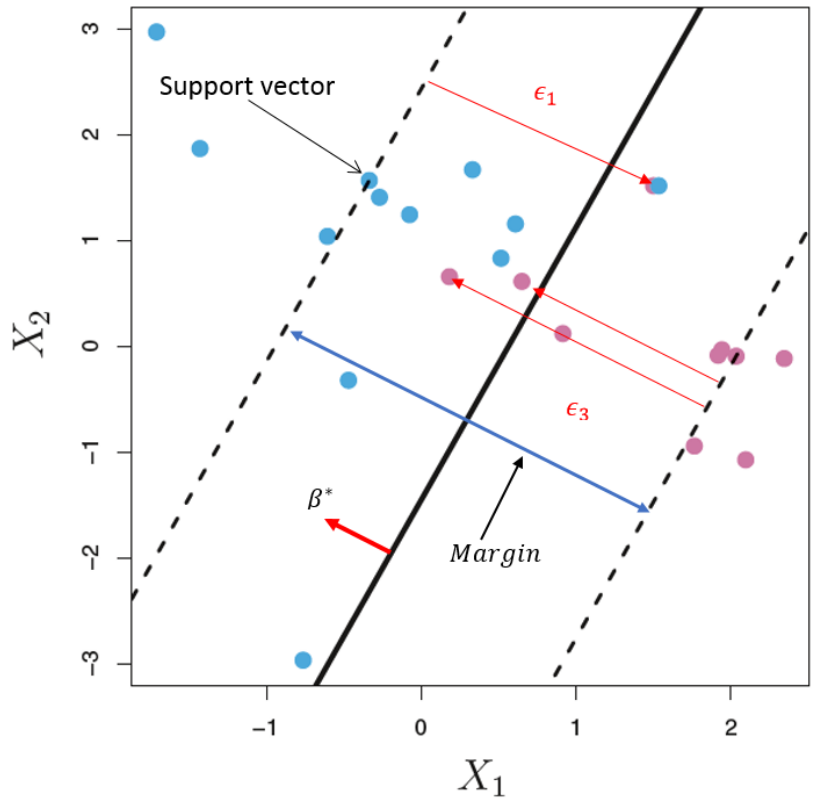
$$L = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i f(x_i) - 1], \ \alpha_i > 0, - - -(**)$$

- The solution for β can be obtained by solving the optimization problem $(**)$, $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$ Now, the solution function $f(x) = x^T \beta + \beta_0$ can be written as

$$\hat{f}(x) = x^T \hat{\beta} + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i x_i + \beta_0$$

Non-Seperable Case (Soft margine classifier)

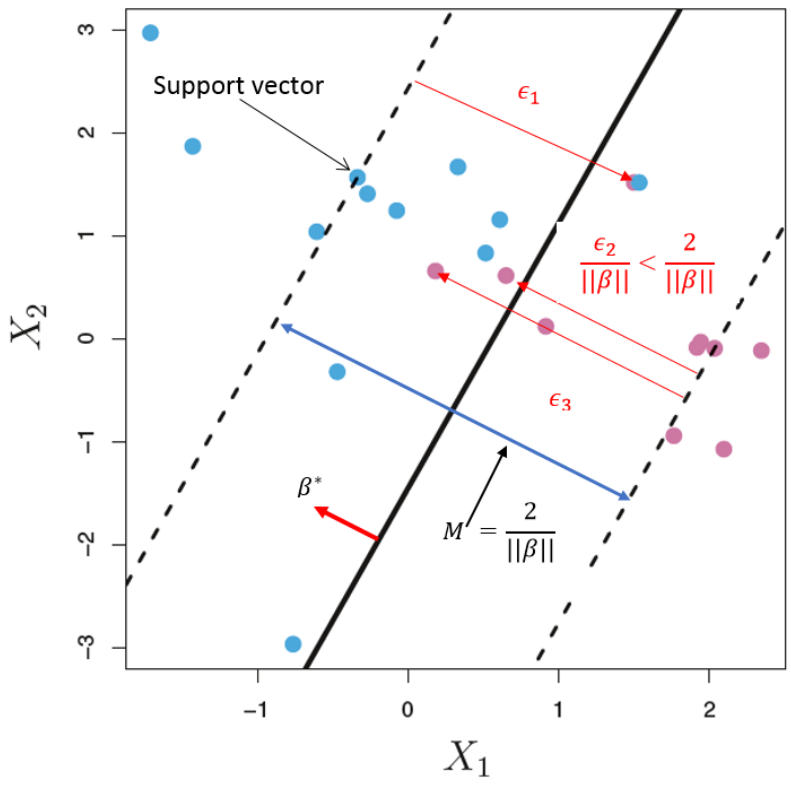
- What is we cannot find a perfect seperating hyperplane?



- We allow some violation and find a near perfect seperation, allowing some training points to be on the other side of the margin or other side of the hyperplane, is called Support Vector Classfier.

Computing Soft margine classifier

- when we cannot separate two classes perfectly, we allow some violations and obtain a better classifier (soft margin classifier).



where $\epsilon_1, \epsilon_2, \ldots$ are the slack variables.

Computing Soft margine classifier

- We can formulate this as a constrained optimization problem

$$\underset{\beta,\beta_0}{\textit{minimize}}\ ||\beta||^2$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \epsilon_i,\ \forall i$$

$$\epsilon \geq 0,\ \sum_i \epsilon_i \leq C$$

Computing Soft margine classifier

- We can formulate this as a constrained optimization problem

$$\underset{\beta,\beta_0}{minimize}\ ||\beta||^2$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \epsilon_i,\ \forall i$$
$$\epsilon \geq 0,\ \sum_i \epsilon_i \leq C$$

- computationally it is convinient to re-express as

$$\underset{\beta,\beta_0}{minimize}\ \frac{1}{2}||\beta||^2 + C\sum_i \epsilon_i$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \epsilon_i,\ \forall i, \epsilon \geq 0,$$

  where C is the amount of penalty (cost) assigned for the slack variables.
- We can estimate the paramters considering the Lagrange function and taking derivates.

Computing Soft margine classifier

- The solution for β can be obtained by solving the optimization problem $(**)$, $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$ Now, the solution function $f(x) = x^T \beta + \beta_0$ can be written as

$$\hat{f}(x) = x^T \hat{\beta} + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i x_i + \beta_0$$

Computing Soft margine classifier

- The solution for β can be obtained by solving the optimization problem $(**)$, $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$ Now, the solution function $f(x) = x^T \beta + \beta_0$ can be written as

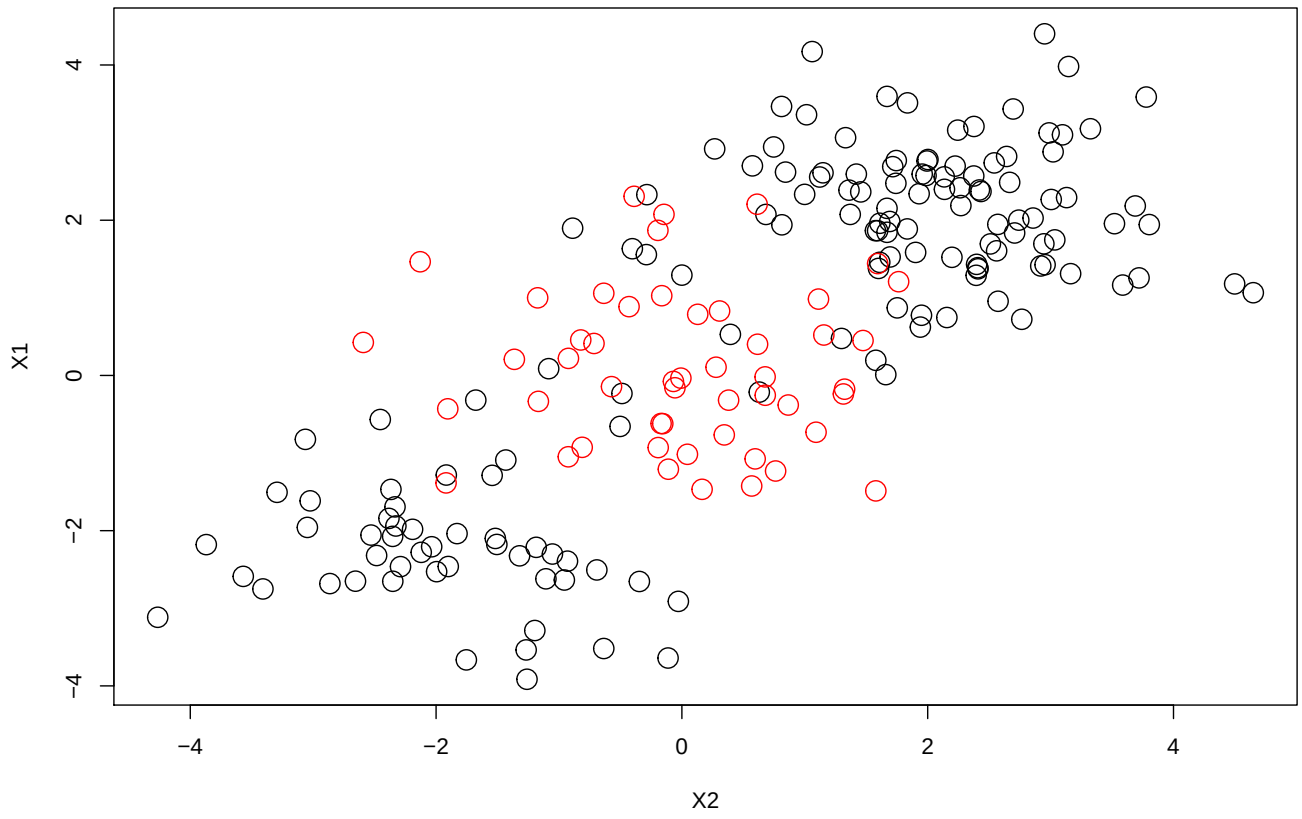
$$\hat{f}(x) = x^T \hat{\beta} + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i x_i + \beta_0$$

Using feature transformation,

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i \kappa(x, x_i) + \hat{\beta}_0$$

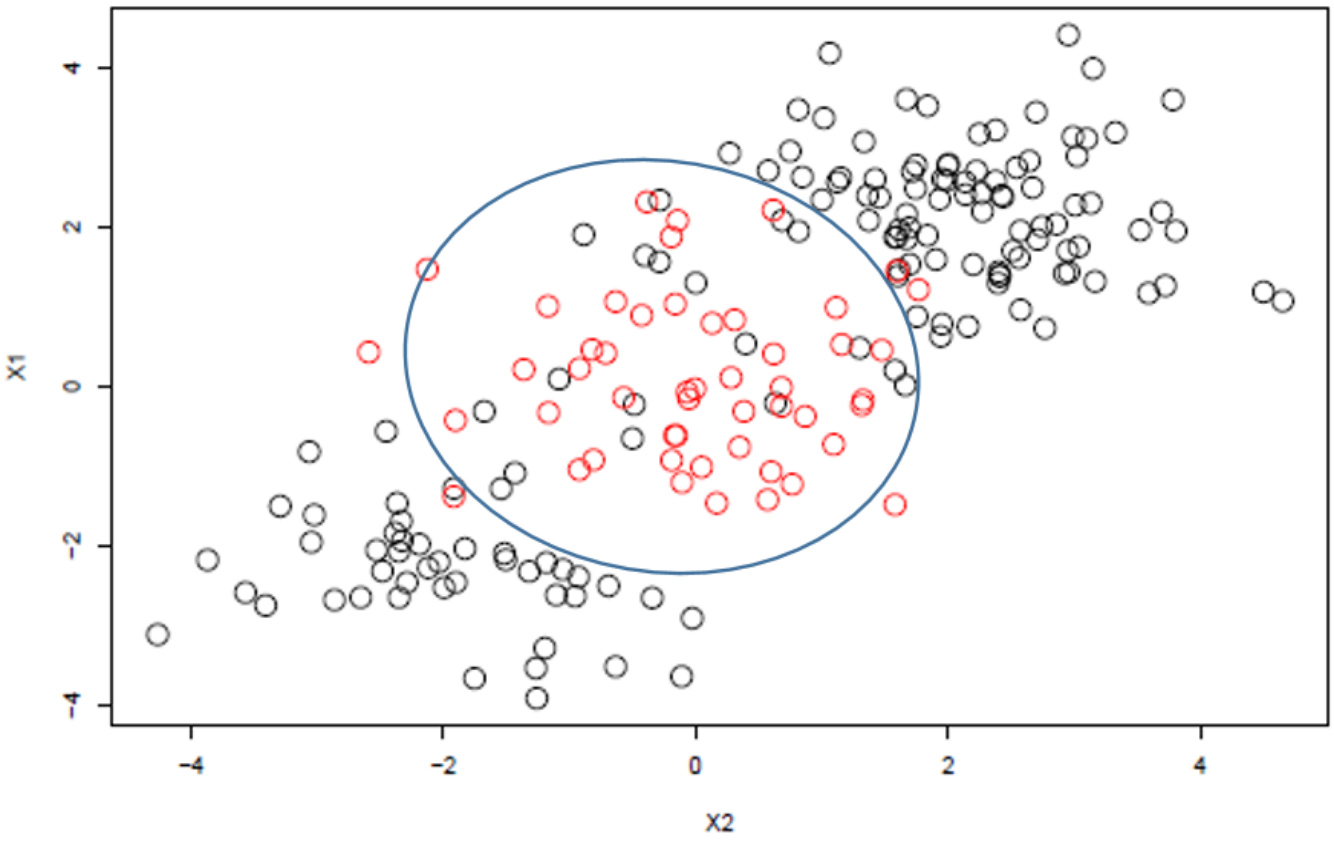
where $\kappa(x, x_i)$ is the kernel function.

Classification with Non-Linear Decision Boundaries



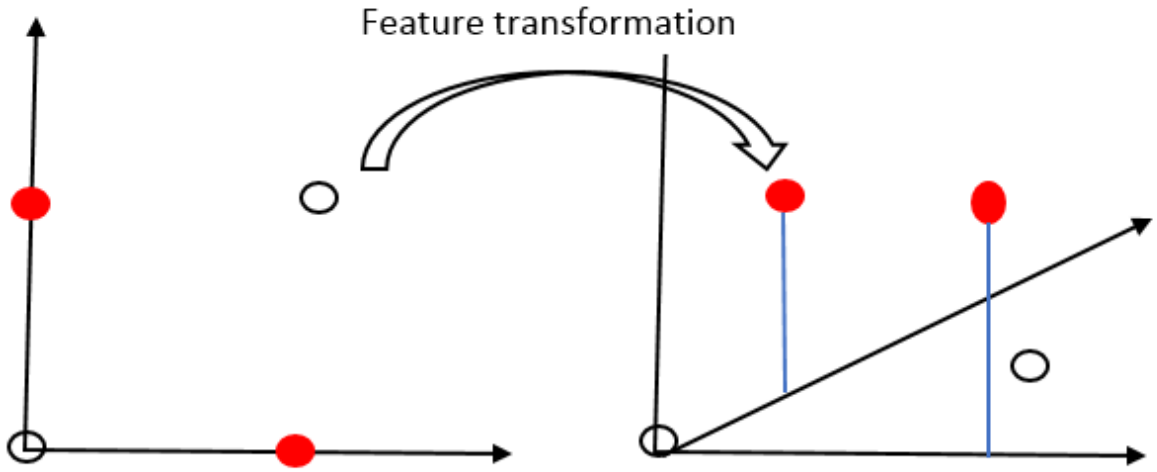
- $K-$ nearest neighbors or SVM with *non-linear decision boundary* would be better.

Classification with Non-Linear Decision Boundaries



Kernel Functions

- What if when we cannot find a seperating hyperplane in given feature space?



- We try to get them to another space such that it allows to classify them, using some kind of transformation, is called the kernel function ($K(x, x_i)$)

$$K(x, x') = h(x) \cdot h(x')$$

Kernel Functions

- The most popular kernel functions ($K(x, x_i)$)

1. linear kernel
$$K(x, x') = x \cdot x'$$
2. polynomial kernel
$$K(x, x') = (1 + x \cdot x')^d$$
3. radial kernel
$$K(x, x') = exp(-\gamma||x - x'||^2), \gamma > 0$$
4. Sigmoid kernel
$$K(x, x') = \tanh(\kappa_1 x \cdot x' - \kappa_2)$$

Suport Vector Machines

- SVM is an extension of the support vector classifier, enlarging the feature space using **kernels**.
- SVM allows to enlarge the feature space using *kernels* in a way that leads to efficient computations.
- When we combine non-linear terms with support vector classifier, it is called Support Vector Machine.

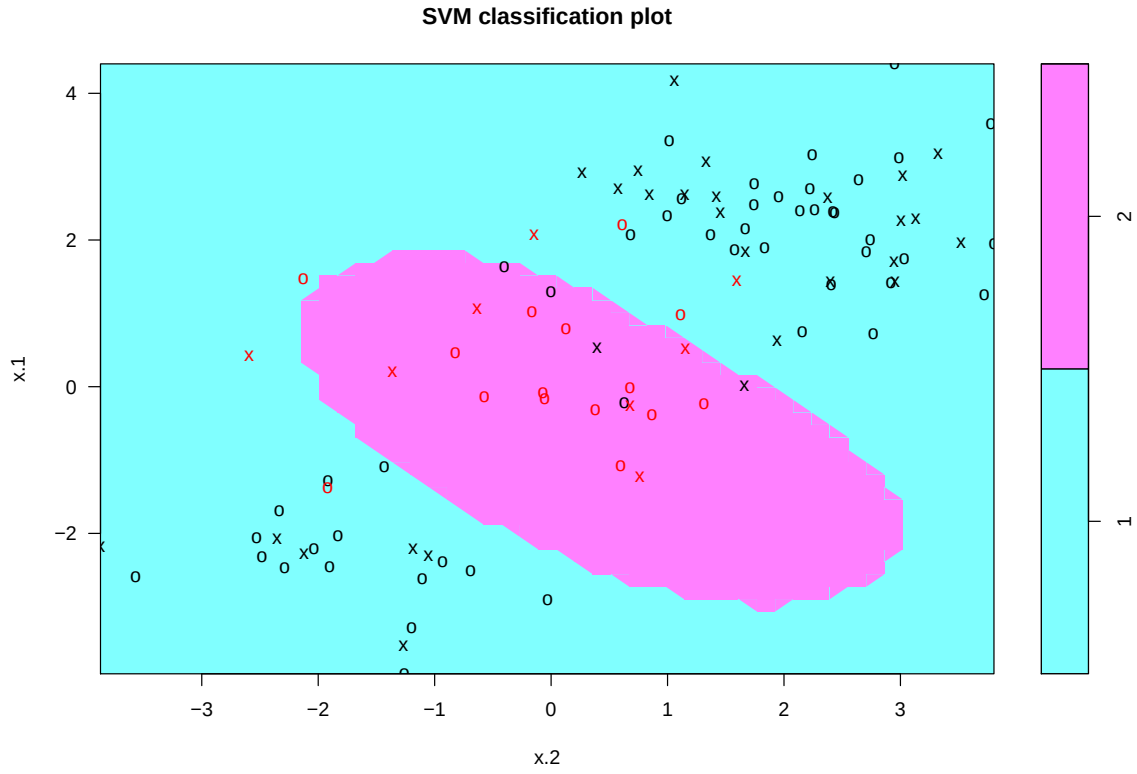
Radial kernel with different C

Consider 200 observations and 2 variables Using redial kernel

```
set.seed(1)
library(e1071)
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
train=sample(200,100)
tune.out=tune(svm, y~., data=dat[train,], kernel="radial",
              ranges=list(cost=c(0.1,1,10,100,1000)))
bestmod=tune.out$best.model
```

Using Radial kernel

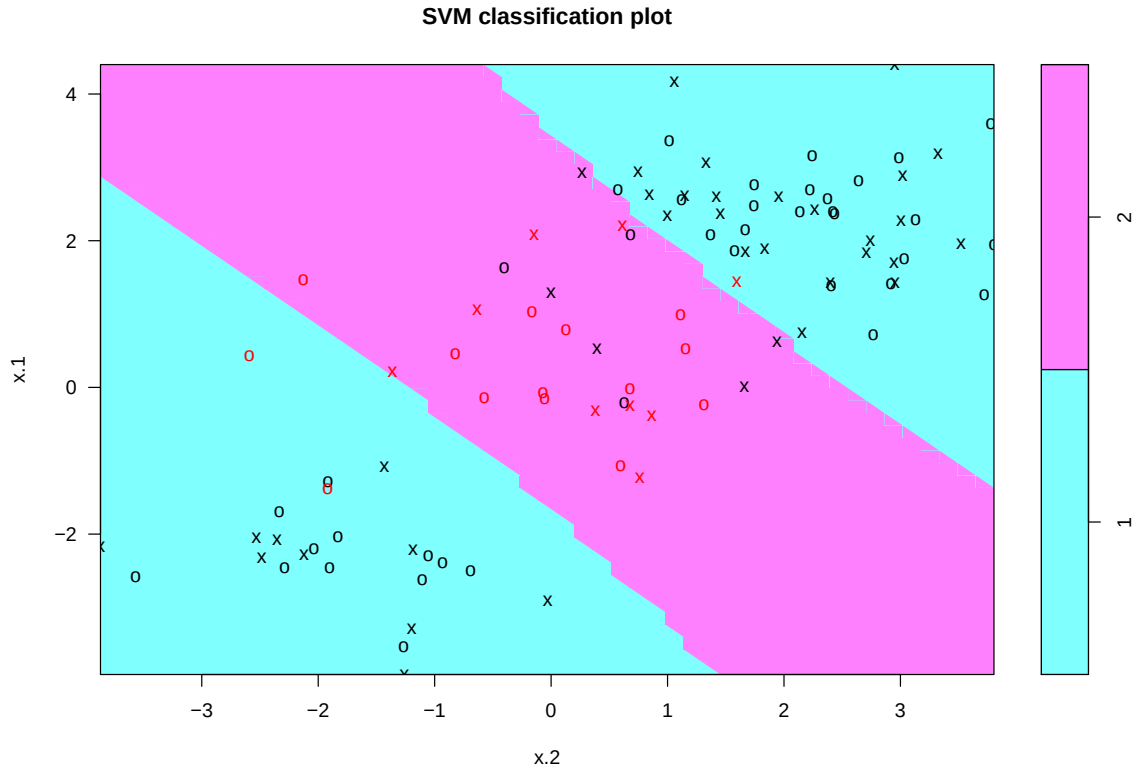
```
plot(bestmod, dat[-train,], cex=2)
```



Using Polynomial kernel

Using polynomial kernel with degree $= 2$, and $C = 1$

```
SVM.poly=svm(y~., data=dat[train,],
                kernel="polynomial",degree=2,cost=1)
plot(SVM.poly, dat[-train,], cex=2)
```



Comparison - Radial kernel Vs. Polynomial kernel

- 12% of the observations are misclassified by this SVM with radial kernel.
- Using radial kernel - confusion matrix

```
     pred
true  1  2
   1 72  5
   2  7 16
```

- 12% of the observations are misclassified using SVM with second degree polynomial kernel
- Using polynomial kernel - confusion Matrix

```
     pred
true  1  2
   1 68  9
   2  3 20
```

SVM in High Dimensions

Ridge penalty

The optimization problem using ridge penalty:

$$\underset{\beta_0,\beta}{minimize} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}||\beta||^2$$

- $\sum_{i=1}^{N}[1 - y_i f(x_i)]_+ =$ loss and $\frac{\lambda}{2}||\beta||^2 =$ penalty
- Tuning parameter (λ) is estimated by cross validation.

SVM in High Dimensions

Ridge penalty

The optimization problem using ridge penalty:

$$\underset{\beta_0,\beta}{minimize} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}||\beta||^2$$

- $\sum_{i=1}^{N}[1 - y_i f(x_i)]_+ =$ loss and $\frac{\lambda}{2}||\beta||^2 =$ penalty
- Tuning parameter (λ) is estimated by cross validation.

Lasso penalty

The optimization problem using lasso penalty:

$$\underset{\beta_0,\beta}{minimize} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}||\beta||$$

SVM over Logistic Regression

- If you train SVM well, SVM avoids overfitting since it has a ragularization paramter.

SVM over Logistic Regression

- If you train SVM well, SVM avoids overfitting since it has a ragularization paramter.
- But, if you do not train SVM well, it will cause overfitting.

SVM over Logistic Regression

- If you train SVM well, SVM avoids overfitting since it has a ragularization paramter.
- But, if you do not train SVM well, it will cause overfitting.
- In logistic regression, we consider the logit of probability is linear, but SVM provides a non-linear decision boundaries to handle non-linear relations.

SVM over Logistic Regression

- If you train SVM well, SVM avoids overfitting since it has a ragularization paramter.
- But, if you do not train SVM well, it will cause overfitting.
- In logistic regression, we consider the logit of probability is linear, but SVM provides a non-linear decision boundaries to handle non-linear relations.
- SVM is a convex optimization problem.

References

Witten, D., Simon, N. (2017), Supervised Learning, Summer Institute for Statistics of Big Data, Univ. of Washington.

Hastie, T., Tibshirani, R., Friedman, J.(2001). The elements of statistical learning, Vol. 1, Springer series in statistics New York.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2014). An Introduction to Statistical Learning: with Applications in R.

Koshiba, Y., & Abe, S. (2003). Comparison of L1 and L2 support vector machines. In Neural Networks, Vol. 3, pp. 2054-2059.

Zhu, J., Rosset, S., Tibshirani, R., Hastie, T. J. (2004). 1-norm support vector machines. In Advances in neural information processing systems (pp. 49-56).

Wang, L., Zhu, J., Zou, H. (2006). The doubly regularized support vector machine. Statistica Sinica, 589-615.