Week 7: Logistic Regression

Data Science Certificate Program

Ryerson University

Bora Caglayan

12 Mar, 2015

Outline

- Assignment 1 Review
- Assignment 2 Discussion
- Logistic Regression
- Logistic Regression for Multiple Classes
- Comparison of Logistic Regression with kNN

Assignment 1 Review

Problem:

- Predict house sale prices in Manhattan based on certain characteristics of the houses.

Cleaning data:

- This dataset was relatively clean compared to some other datasets.
- Cleaning data is time consuming.

Some tips:

- For datasets with many dimension storage in a database might be more meaningful.
- Track the changes on the dataset and related documentation for longer term projects.

Assignment 1 Review

Model Building:

- Do you think linear regression can give sufficiently good results? Can this model substitute a real estate agent?
- Testing the model on a third independent dataset gives more accurate results in real scenarios.

Model Improvement:

- Did linear regression work?
- There are three ways to improve a model:
    - Increase number of instances
    - Increase number of attributes
    - Calibrate the model
- There are possible extensions:
    - Multilevel models: Suggest different models for different subsets of the instances.
    - Non-linear models

Assignment 1 Review

- Caution: Regression is used to show the relation between the dependent (target) variable and the input variable(s). If patterns change, the predicted relation might no longer be true. Some common causes for such changes might be:

- Sudden disruptive changes

- Lack of generalizability

Assignment 2

- Handwritten digit recognition with kNN

You are given a training and test dataset.

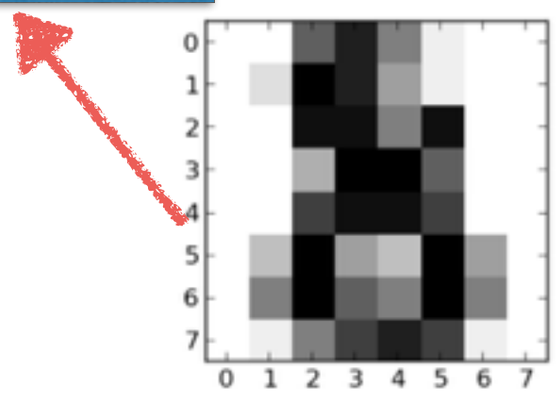
1. Pendigits.tra.csv
2. Pendigits.tes.csv

Each row contains 16 features (a 4x4 bitmap of a handwritten digit) and its correct label on the last column. Every feature is a numeric grayscale pixel value. 255 is white and 0 is black. Here is an example of a 8x8 scanned handwritten digit:

Every pixel is a feature.

Using kNN, predict the digits on the test dataset by training the model on the training dataset.

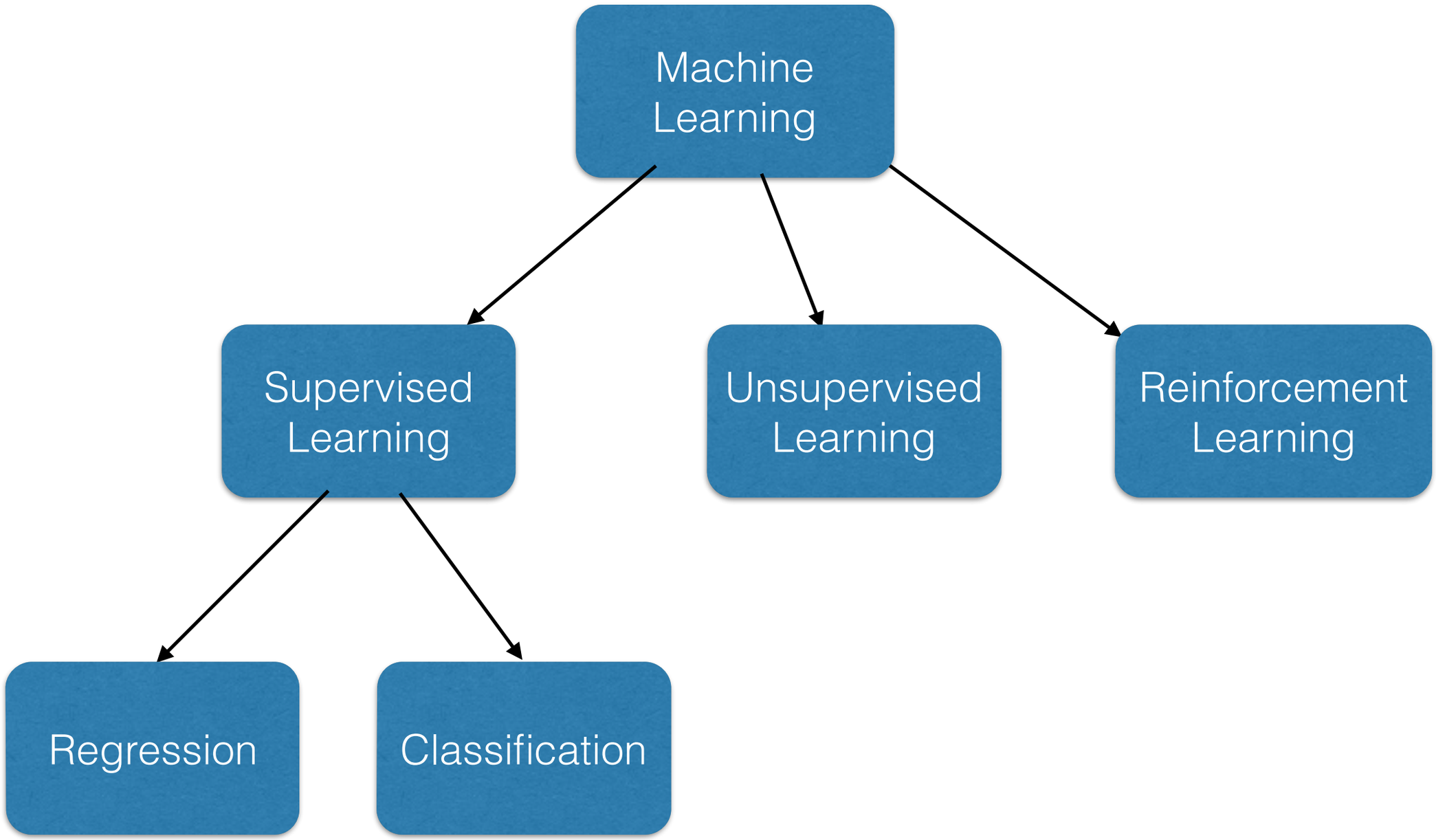
For humans



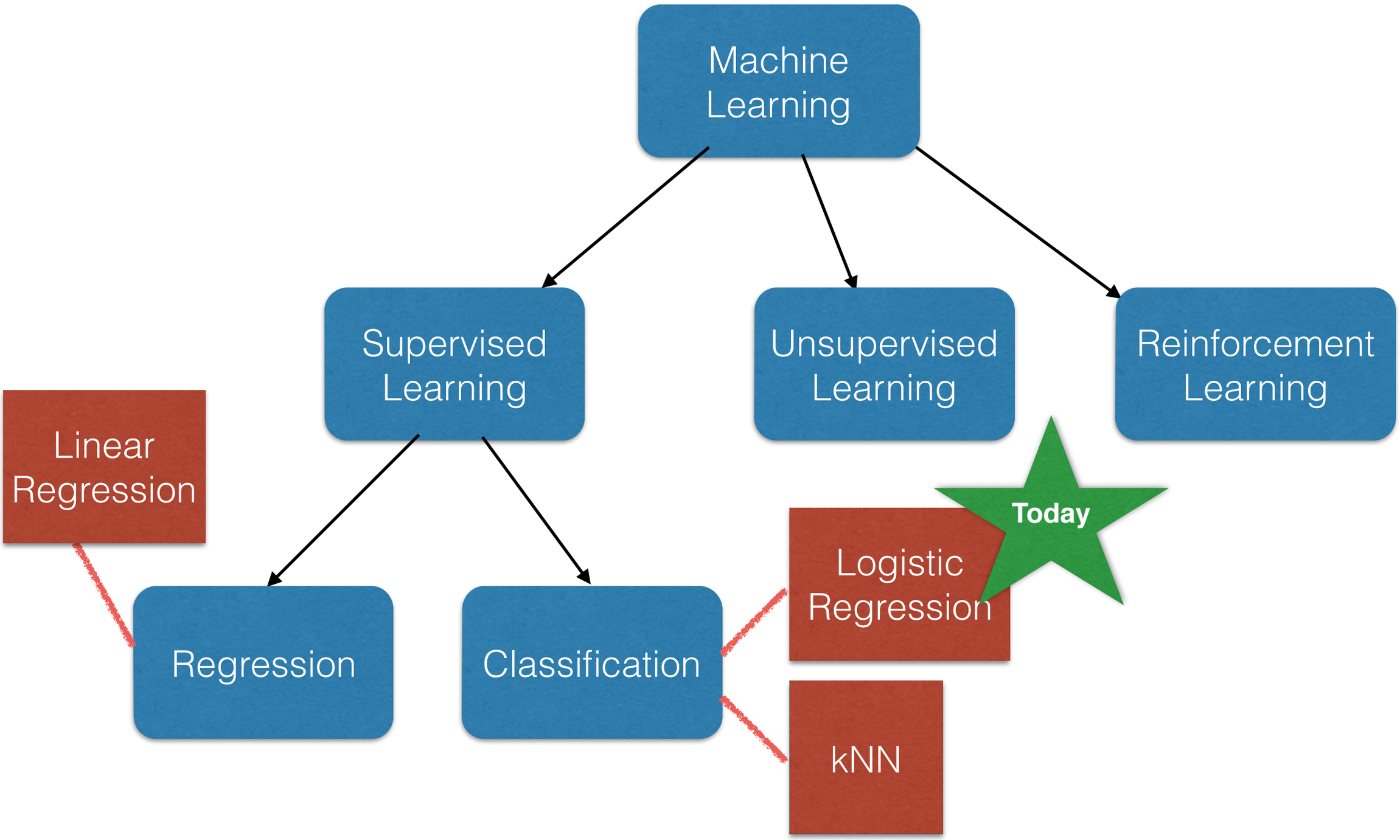
For computers

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Machine Learning Problems



Machine Learning Problems

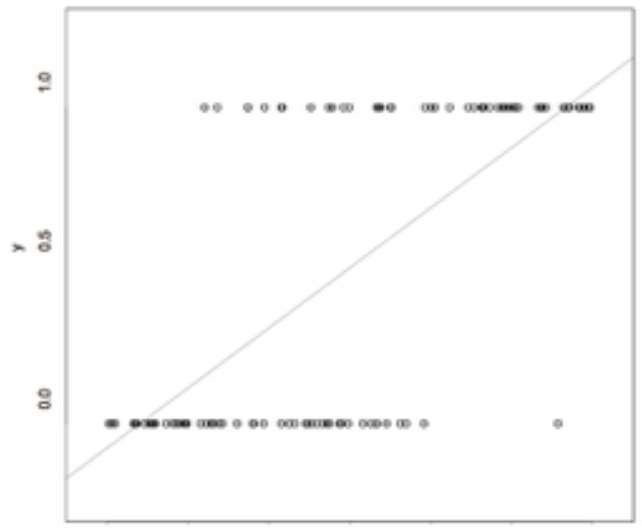


Logistic Regression

- In binary classification the target variable is a *dichotomous* variable:
  - Examples: {sunny, cloudy}, {faulty, safe} etc.

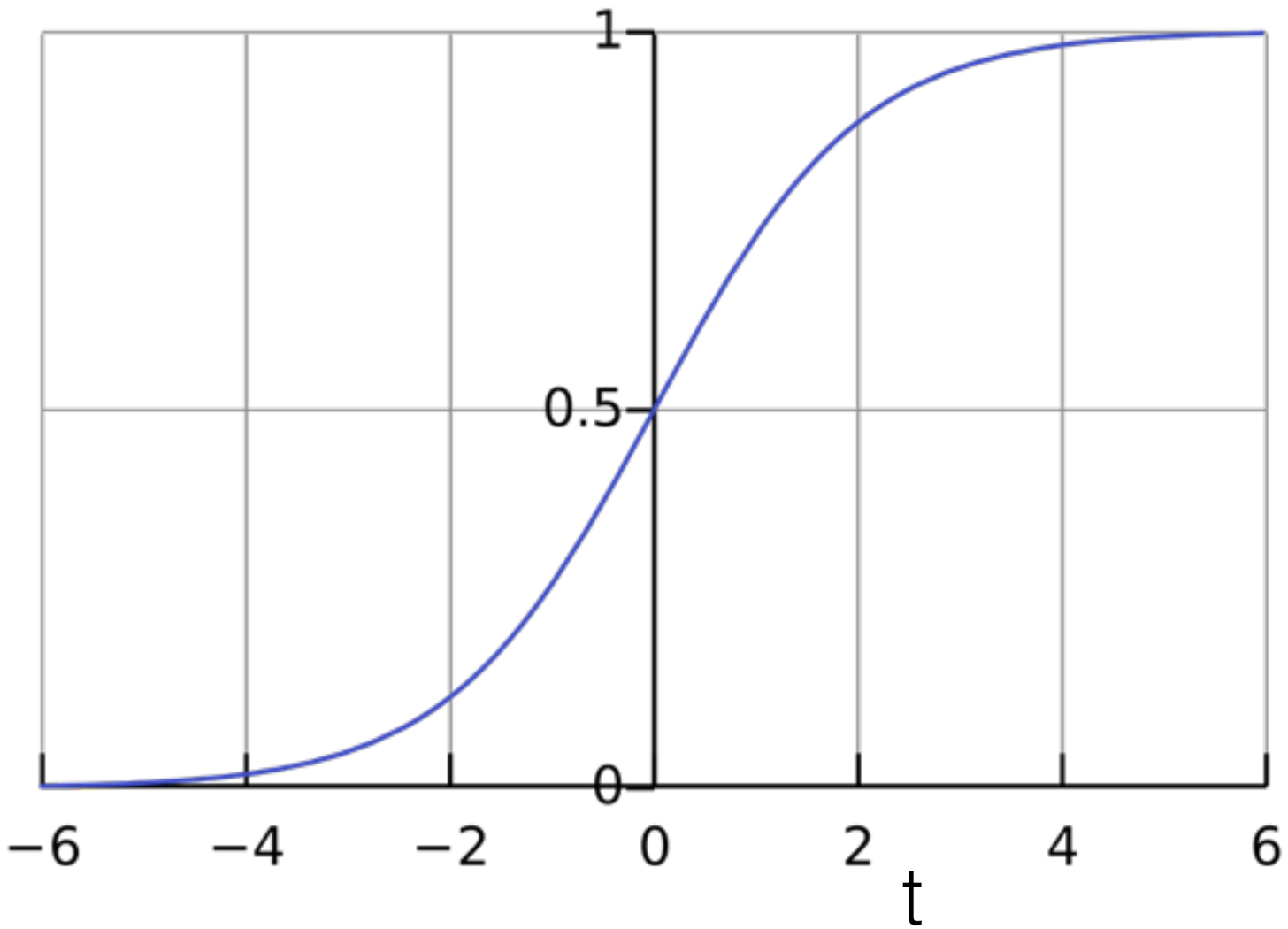
- Linear regression function is not a good discriminator function for such a scenario.

$$f(x) = \theta_1 x + \theta_0$$



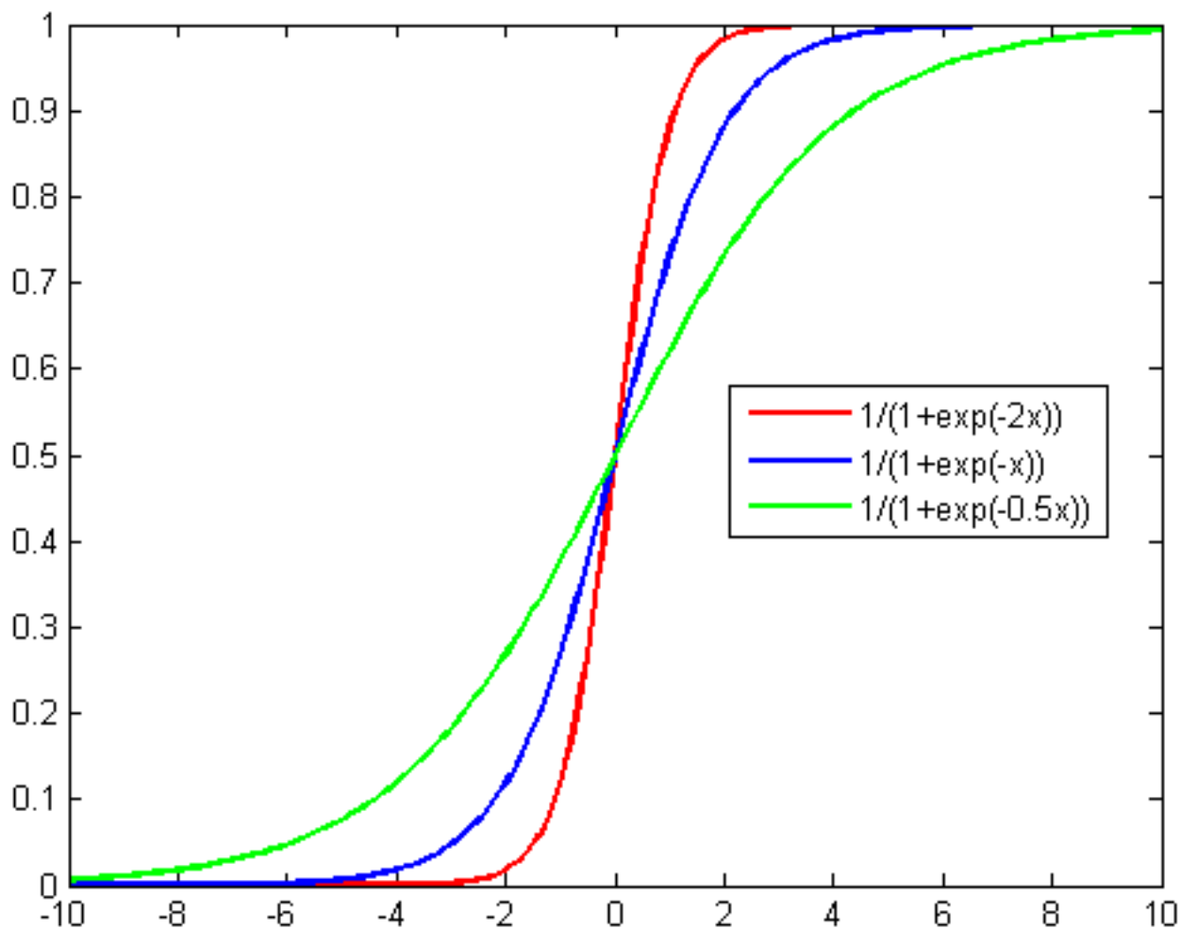
Logistic Function

$$f(t) = \frac{1}{1+e^{-t}} \qquad f(x) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}}$$

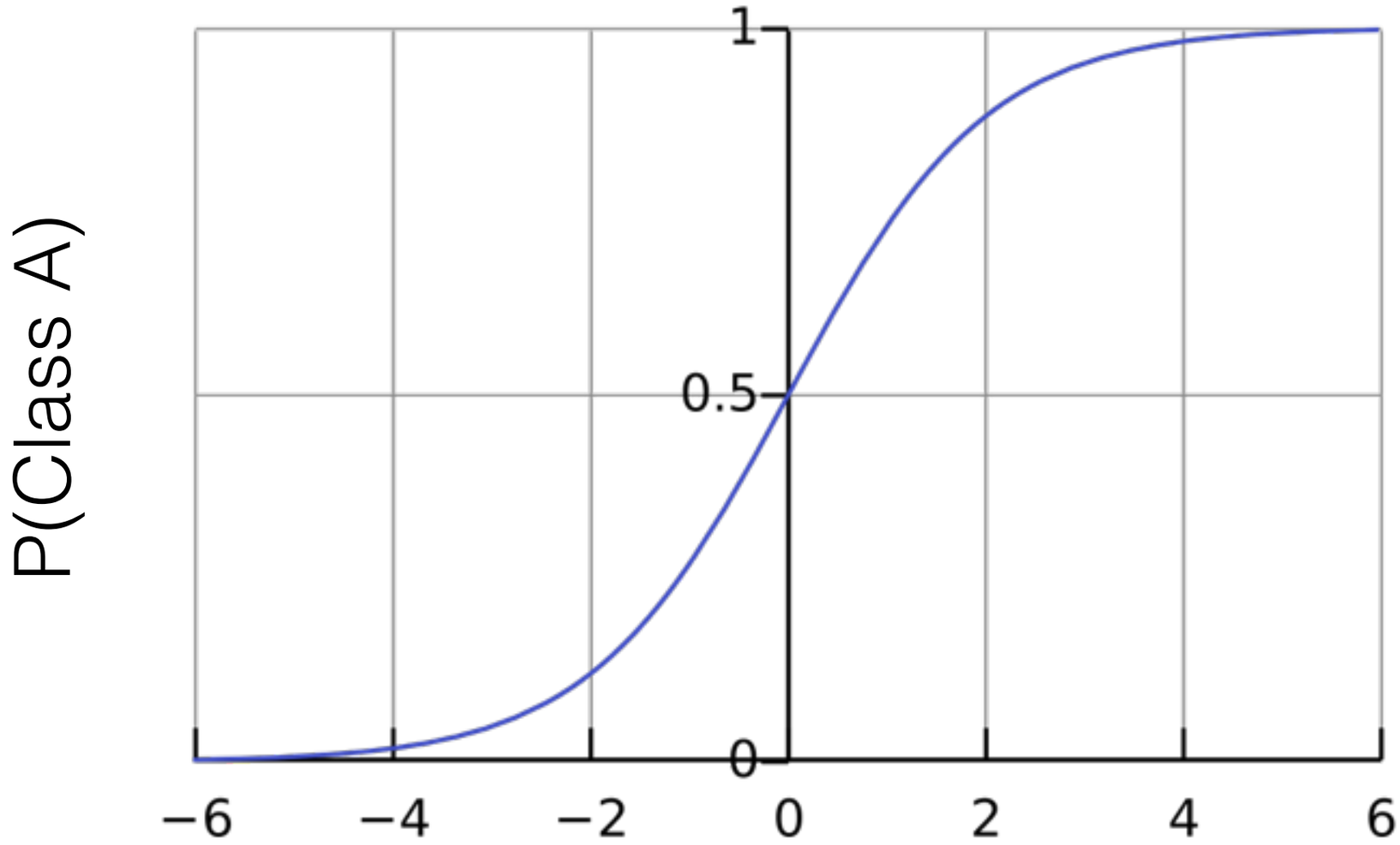


Logistic Function

Based on the theta coefficients the function shape changes. The intercept value changes the x value where there is maximum slope.



Logit Function

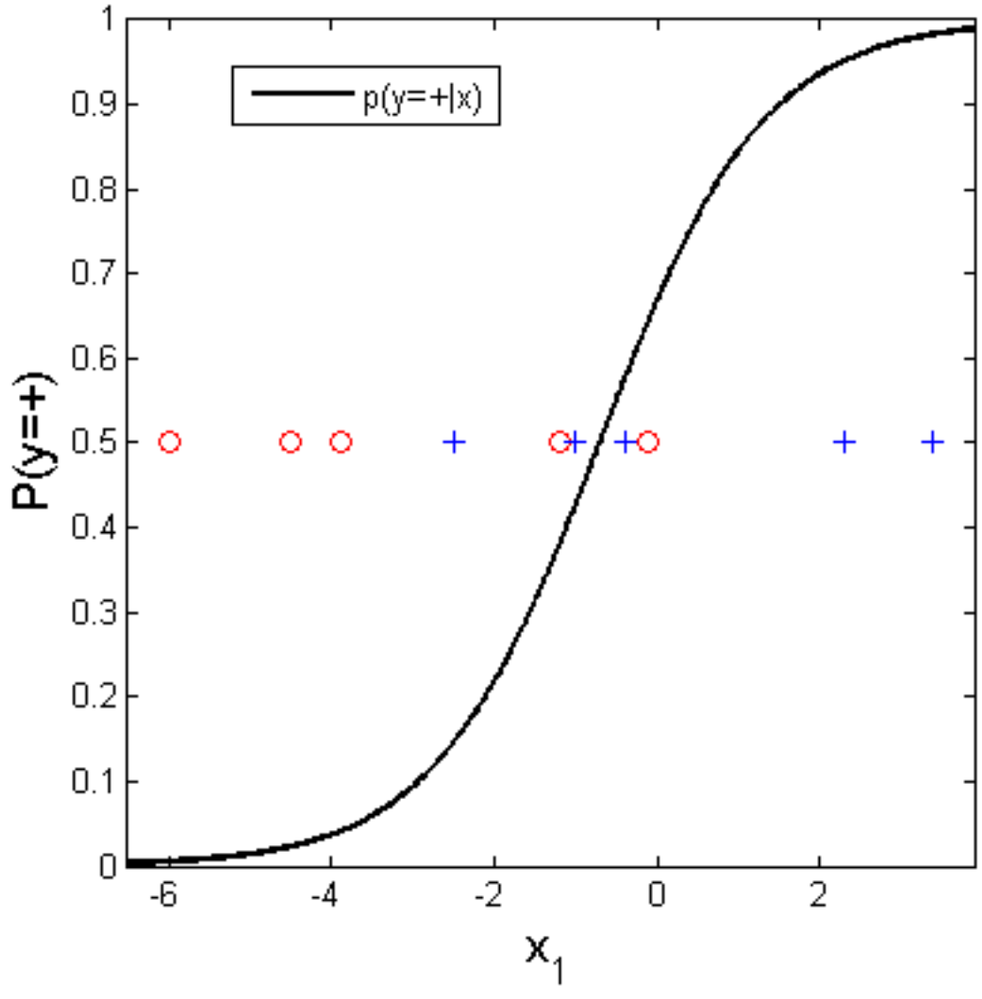


$$0 \leq p(classA) \leq 1$$

Logistic Function

The model predicts the new instance class based on the predicted probabilities.

Implication: In addition to prediction,
we can report the predicted probability for a class.



Univariate Logistic Regression

We start with a single input variable x and two classes a and b.

The goal: Build a model to predict the classes of new instances.

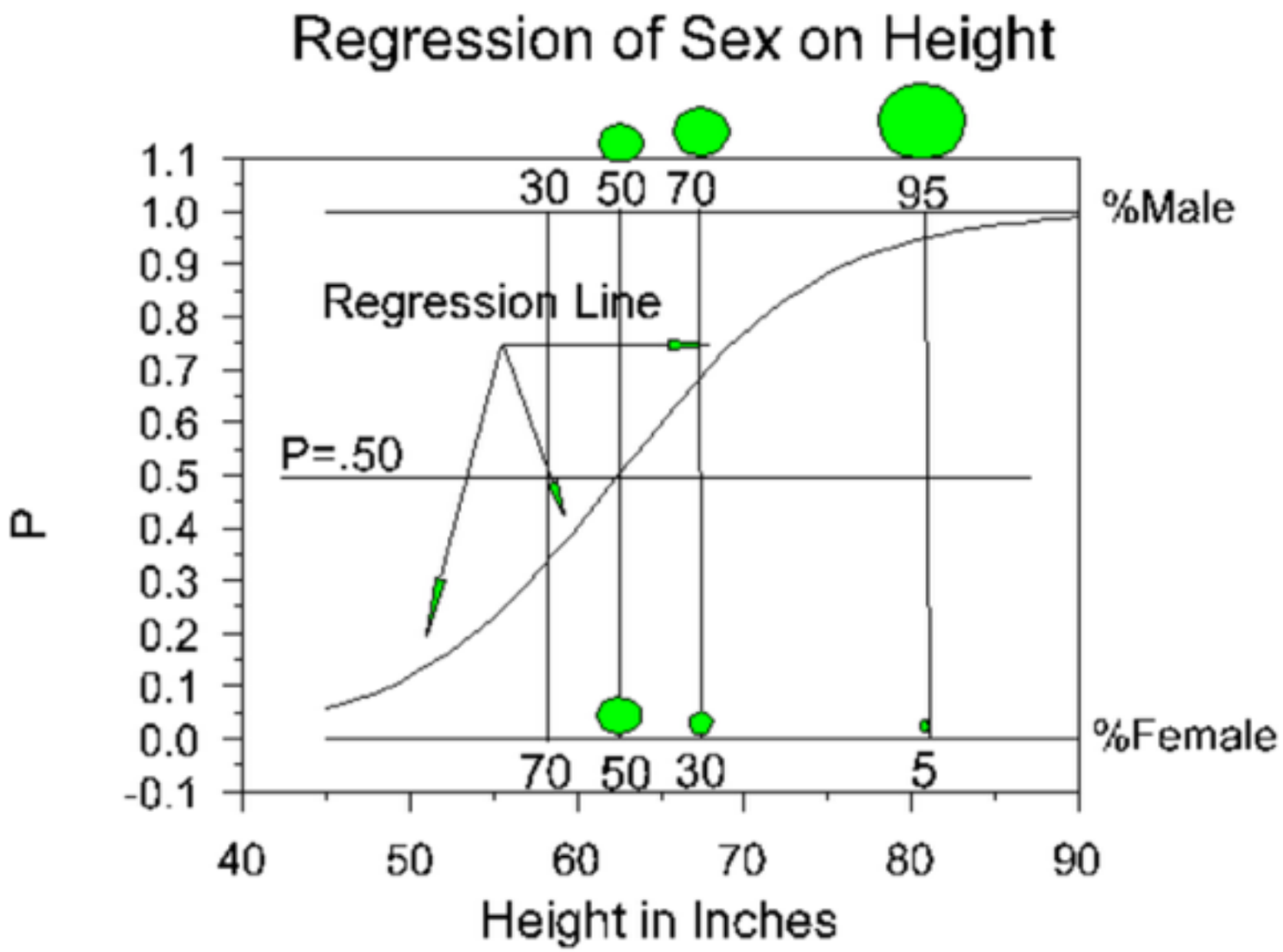
$$p(ClassA) + p(classB) = 1$$

$$p(ClassA) = \frac{1}{1 + e^{-(\theta_1 x + \theta_0)}}$$

$$p(ClassB) = -\frac{1}{1 + e^{-(\theta_1 x + \theta_0)}}$$

Univariate Logistic Regression

An example:



Univariate Logistic Regression

$\frac{\partial}{\partial \theta_j} J(\theta)$ has a single minimum

Finding the parameters:

Goal is finding the theta values that minimizes the cost function. We define the following **convex** function;

For m instances:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

If prediction and actual classes are same the cost is 0 otherwise the cost is 1.

Afterwards, we can use gradient descent to estimate the cost.

Multivariate Logistic Regression

We start with *N* input variables and two classes namely A and B.

The goal: Build a model to predict the classes of new instances.

$$p(ClassA) + p(classB) = 1$$

$$p(ClassA) = \frac{1}{1 + e^{-\sum_{i=1}^{M}(\theta_i x_i) + \theta_0}}$$

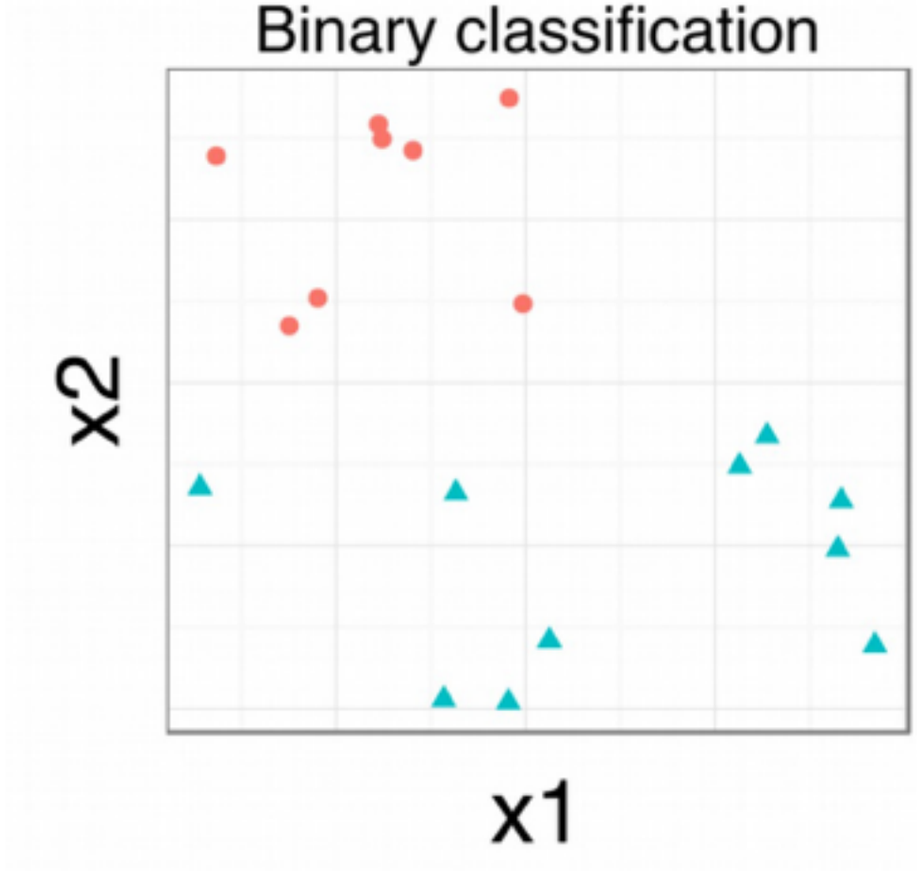
$$p(ClassB) = -\frac{1}{1 + e^{-\sum_{i=1}^{M}(\theta_i x_i) + \theta_0}}$$

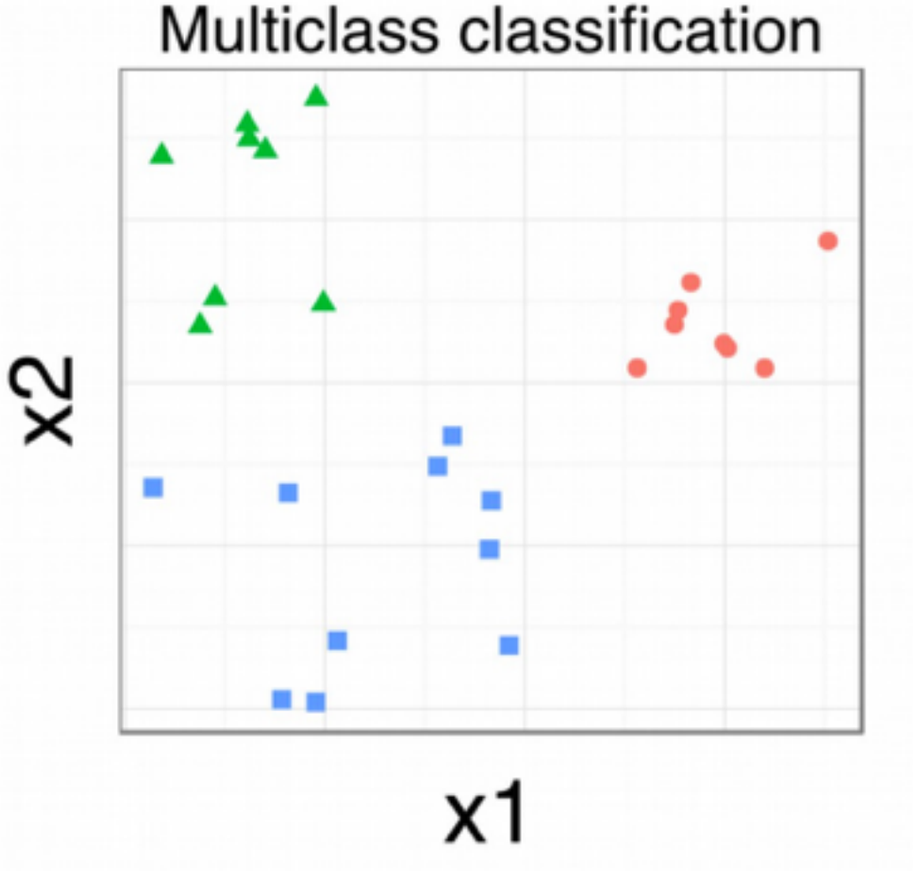
Multivariate Logistic Regression

- The model has same similar types of problems as linear regression.
- Strong assumptions about the relations between the input variables and the target variables.
- Collinearity among the input variables.
- Over-fitting and under-fitting the training samples.

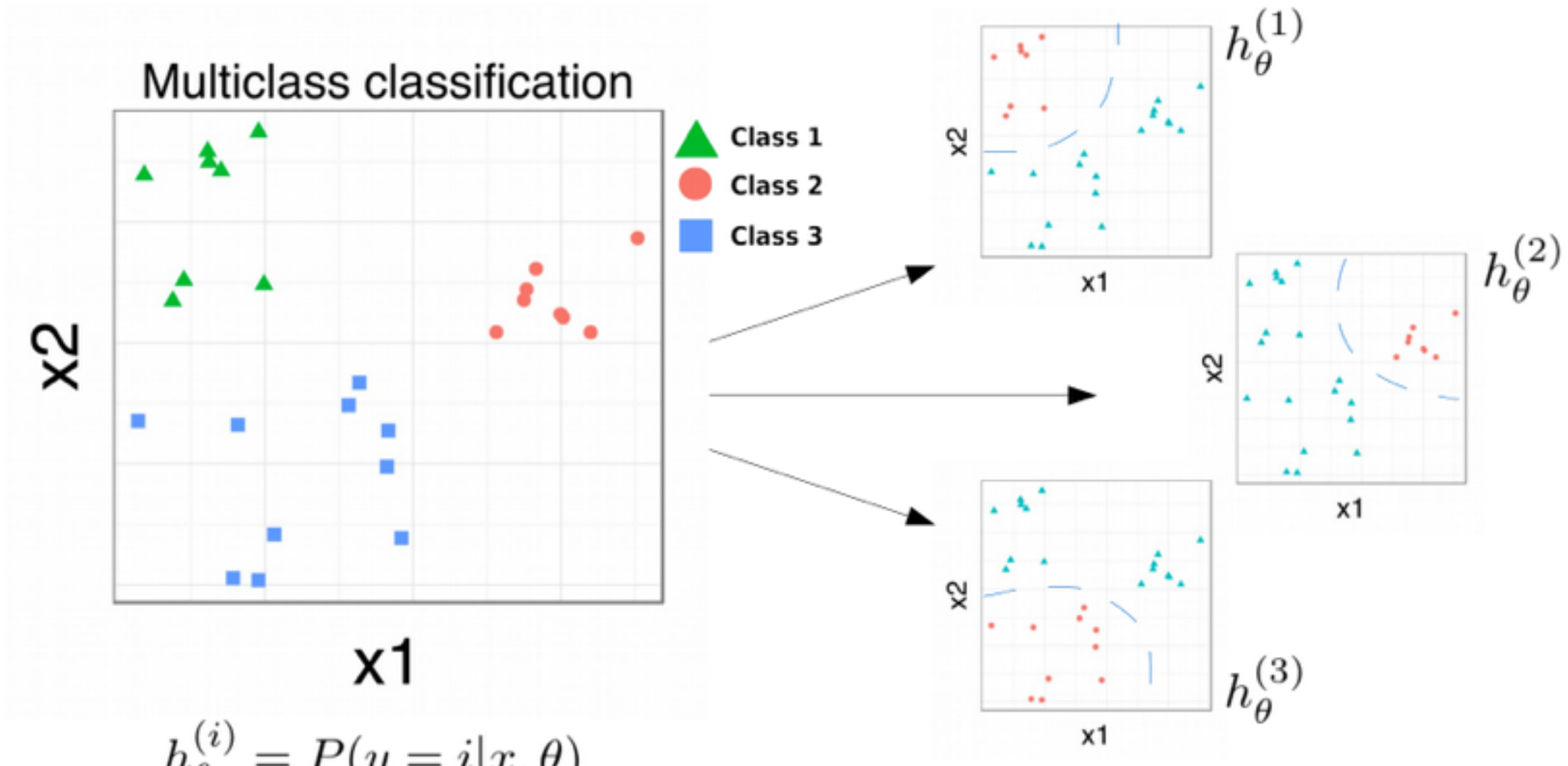
Logistic Regression with More than Two Classes

Idea: For m classes we define m-1 binary classification problems.





Logistic Regression with More than Two Classes



$$h_\theta^{(i)} = P(y = i|x, \theta)$$

where $i \in \{1, 2, 3\}$

Stepwise Logistic Regression

Problem: Select minimum number of attributes for the model to avoid curse of dimensionality.

Strategies:

- **Forward selection**
  - Start with no variables in the model.
  - Test the addition of each variable using a chosen model comparison criterion
  - Add the variable (if any) that improves the model the most.
  - Repeat this process until none improves the model.

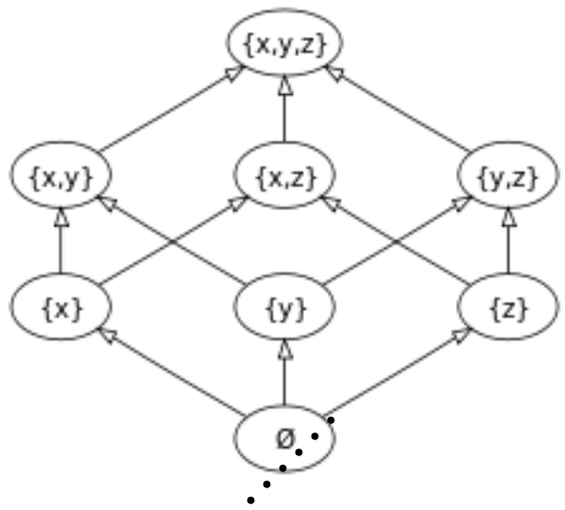
- **Backward elimination**
  - Start with all candidate variables.
  - Test the deletion of each variable using a chosen model comparison criterion.
  - Delete the variable (if any) that improves the model the most by being deleted.
  - Repeat this process until no further improvement is possible.

Possible Problems: Depends on the model performance criteria, biased tests.

Finding the Best Subset

N variables may have $2^N - 1$ combinations (same as power sets).

If there is only a few candidate variables every subset of the variables can be evaluated exhaustively in the model.



Regularized Logistic Regression

The goal:

Regularization term

Keep coefficients small to reduce sudden changes.

We change the cost function to reduce the coefficient values.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \quad + \frac{\lambda}{2m}\sum_{i=1}^{m} \theta_m$$

Idea: The values of the coefficients are added to the cost. **Very high parameters would increase the cost function value**.

Remember: The goal of logistic regression is minimizing the cost function.

Summary

Logistic regression model

- **Definition:** Binary classification using numeric attributes. Can be extended to multiple classes.

- **Exploratory analysis:** Check correlations, scatter plot all the pairs.

- **Preprocessing:** Changing scales, *feature selection, feature extraction*

- **Algorithms:** Gradient descent variations

- **Experiment:** k-fold cross validation, random split

- **Performance criteria:** recall, precision, confusion matrix

- **Advantages:** Probabilistic output

- **Disadvantage:** Strong assumptions about the data.

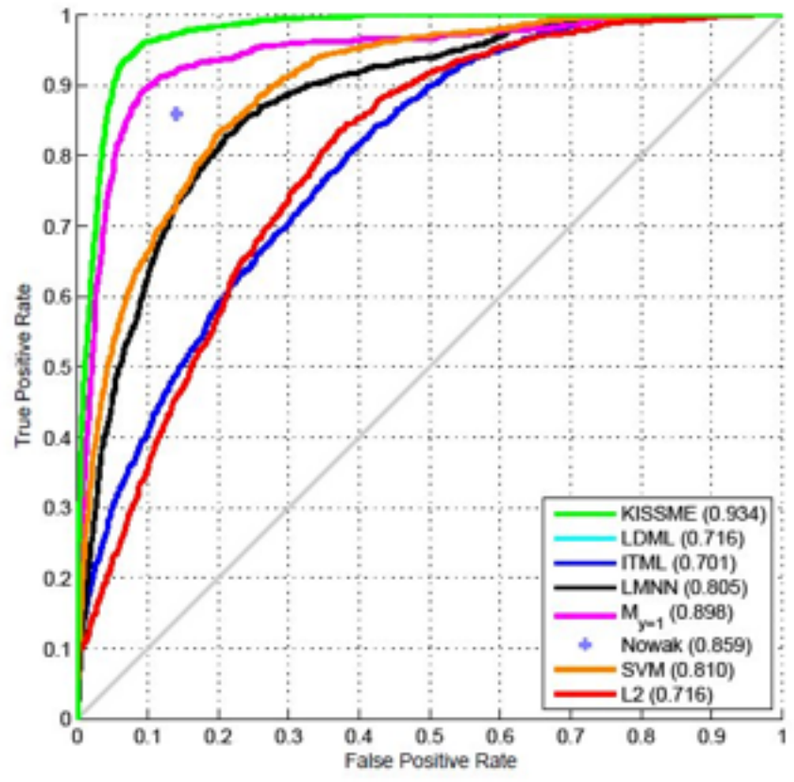
Comparison of Logistic Regression with kNN

- **Remember:** Every machine learning model has some *weaknesses* and *strengths*.
- Both models are used to build classification models widely. Their execution can be paralleled very easily.
- Training cost of kNN is very low compared to logistic regression. Training samples are stored in memory.
- Computation cost is very low for logistic regression model during evaluation. Suitable for real-time prediction scenarios.
- Memory requirement of kNN can be very high since all of the training dataset is stored for comparison.
- kNN has less assumptions about the distributions in the data.
- kNN better generalizes to many class prediction problems.

Machine Learning Benchmark Example

- The best way to evaluate multiple learning models is benchmark them with carefully designed experiments.

- ROC Curve Comparison



References

- There is an ongoing digit recognizer competition: http://www.kaggle.com/c/digit-recognizer

- An advanced image classification example: http://deeplearning.cs.toronto.edu/

- Logistic regression mathematical details (check part 2): http://cs229.stanford.edu/notes/cs229-notes1.pdf

Week 7 Application Part

Preparation

```
library("RWeka") # for datasets
library("ROCR") # visualize performance of classifiers
library("caret") # for confusion matrix
library("e1071") # may be needed for caret
diabetes <- read.arff(system.file("arff", "diabetes.arff",
                                  package = "RWeka"))
```

Solution of the Optional Problem From Last Week

In this part, we will go through the R code steps based on the solution for lab 6.

Dynamically Create R Formulas

You do not need to write variable names every time on regression models.

```
formula_text <- paste(names(diabetes[9]),"~",
                      paste(names(diabetes[1:8]), collapse="+")
                      )
formula <- as.formula(formula_text)
formula
```

```
class ~ preg + plas + pres + skin + insu + mass + pedi + ag
```

Fit a logistic Regression Model

Remember: fit checks your model on the same data.

```
diabetes$class <- as.integer(diabetes$class) - 1
fit <- glm(formula,data=diabetes,family=binomial())
summary(fit) # display fit results
```

```
Call:
glm(formula = formula, family = binomial(), data = diabetes

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5566  -0.7274  -0.4159   0.7267   2.9297

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.4046964  0.7166359 -11.728  < 2e-16 ***
preg         0.1231823  0.0320776   3.840 0.000123 ***
```

Predict using Random Split

```
rn_train <- sample(nrow(diabetes),
                   floor(nrow(diabetes)*0.7))
train <- diabetes[rn_train,]
test <- diabetes[-rn_train,]
fit <- lm(formula, data=train)
```

Predict using Random Split

```
test$scores <- predict(fit, type="response",
                       newdata=test)

pred<-prediction(test$scores, test$class)
```

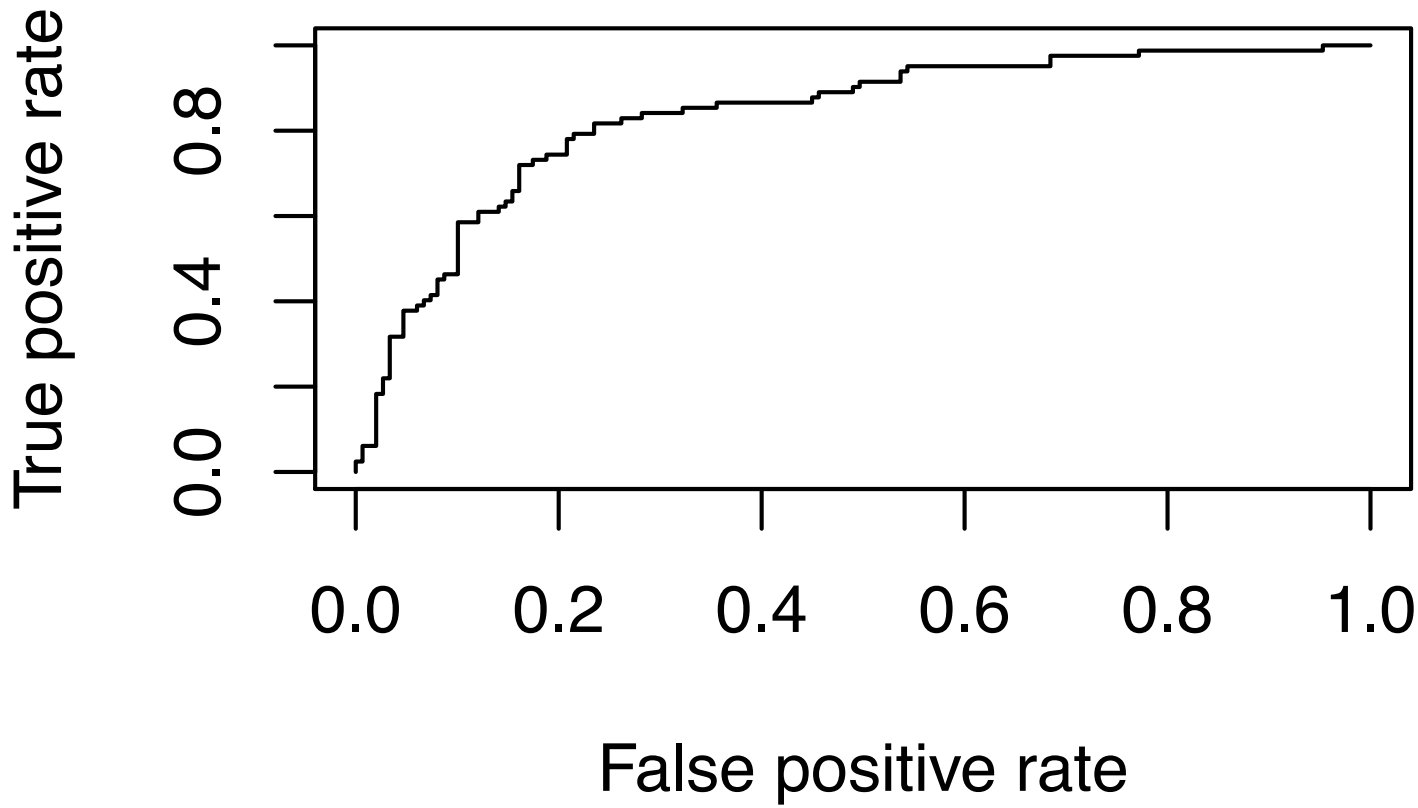
Create a Confusion Matrix

```
c <- confusionMatrix(as.integer(test$scores > 0.5),
                     test$class)
c$table
```

```
          Reference
Prediction   0   1
         0 134  35
         1  15  47
```

Plot the ROC Curve

```
perf<-performance(pred,"tpr","fpr")
plot(perf, lty=1)
```



Using rWeka For Logistic Regression

```
diabetes <- read.arff(system.file("arff", "diabetes.arff",
                                  package = "RWeka"))
weka_fit <- Logistic(formula,
                     data = diabetes)
evaluate_Weka_classifier(weka_fit, numFolds = 10)
```

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         600               78
Incorrectly Classified Instances       168               21
Kappa statistic                          0.494
Mean absolute error                      0.3101
Root mean squared error                  0.396
Relative absolute error                 68.2213 %
Root relative squared error             83.0867 %
Coverage of cases (0.95 level)          99.349 %
```

Lab Preparation

```
library("RWeka") # rweka (embedded Weka software)

diabetes <- read.arff(system.file("arff", "diabetes.arff",
                                  package = "RWeka"))
```

Lab problems:

Using standard mtcars dataset

1. Create a formula am~hp+mpg+gear+carb.
2. Create a logistic regression model using the parameters.
3. Check the performance of the logistic regression model and show ROC curve and confusion matrix.
4. Using rweka interface compare the performance of knn with logistic regression for the same problem.
5. (Optional) Play with the model parameters and attributes to increase the performance of your model.