Data Mining
Classification: Alternative Techniques

Lecture Notes for Chapter 5

Introduction to Data Mining

by

Tan, Steinbach, Kumar

These slides have been modified for the CS 4232/ 5232 course

Outline

- Other Classifier Methods
  - Nearest Neighbor Classifier
  - Bayesian Classifier
  - Logistic Regression Classifier
  - Neural Network Classifier
  - Support Vector Machines Classifier
- Ensemble Methods
  - Bagging
  - Boosting

Part IIA

Nearest neighbor Classifiers, Bayesian Classifiers and Logistic Regression Classifiers

Classifiers

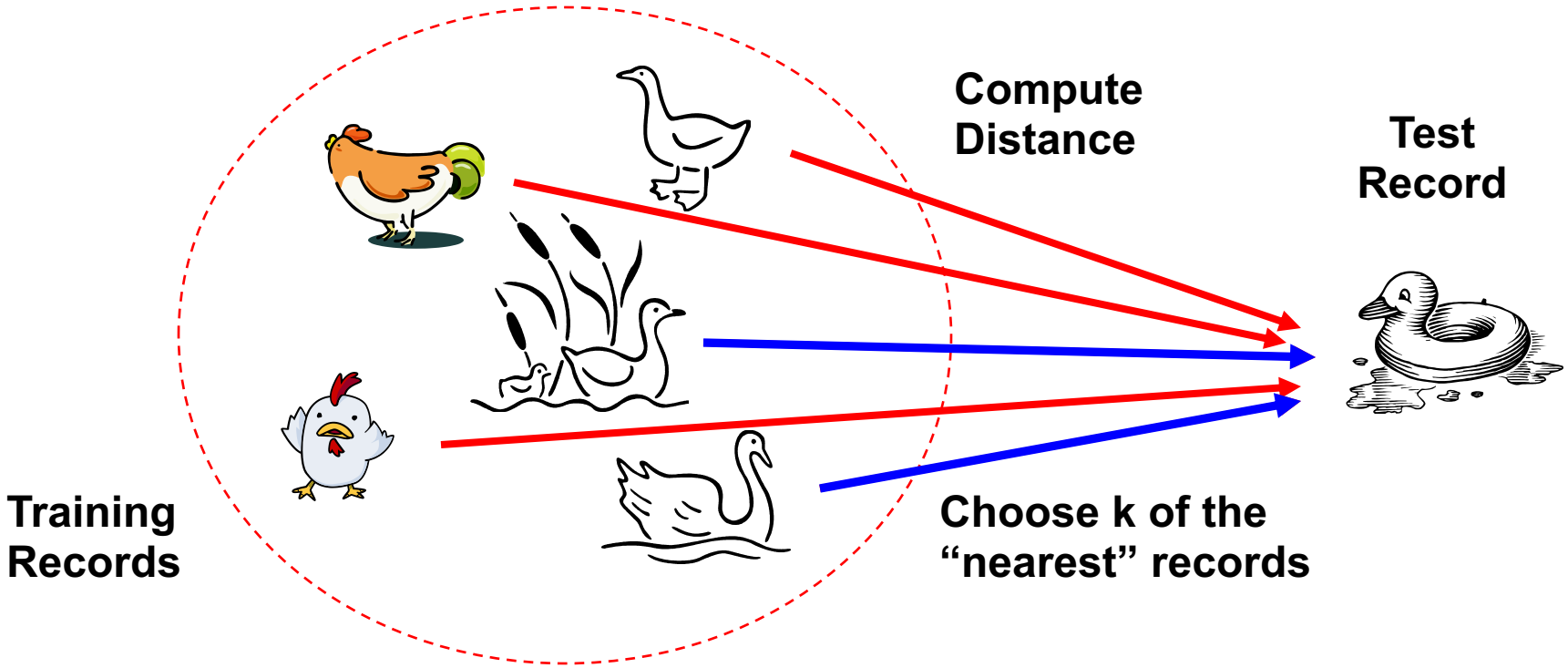
Nearest Neighbor Classifiers

Nearest Neighbor vs. Instance Based Classifiers

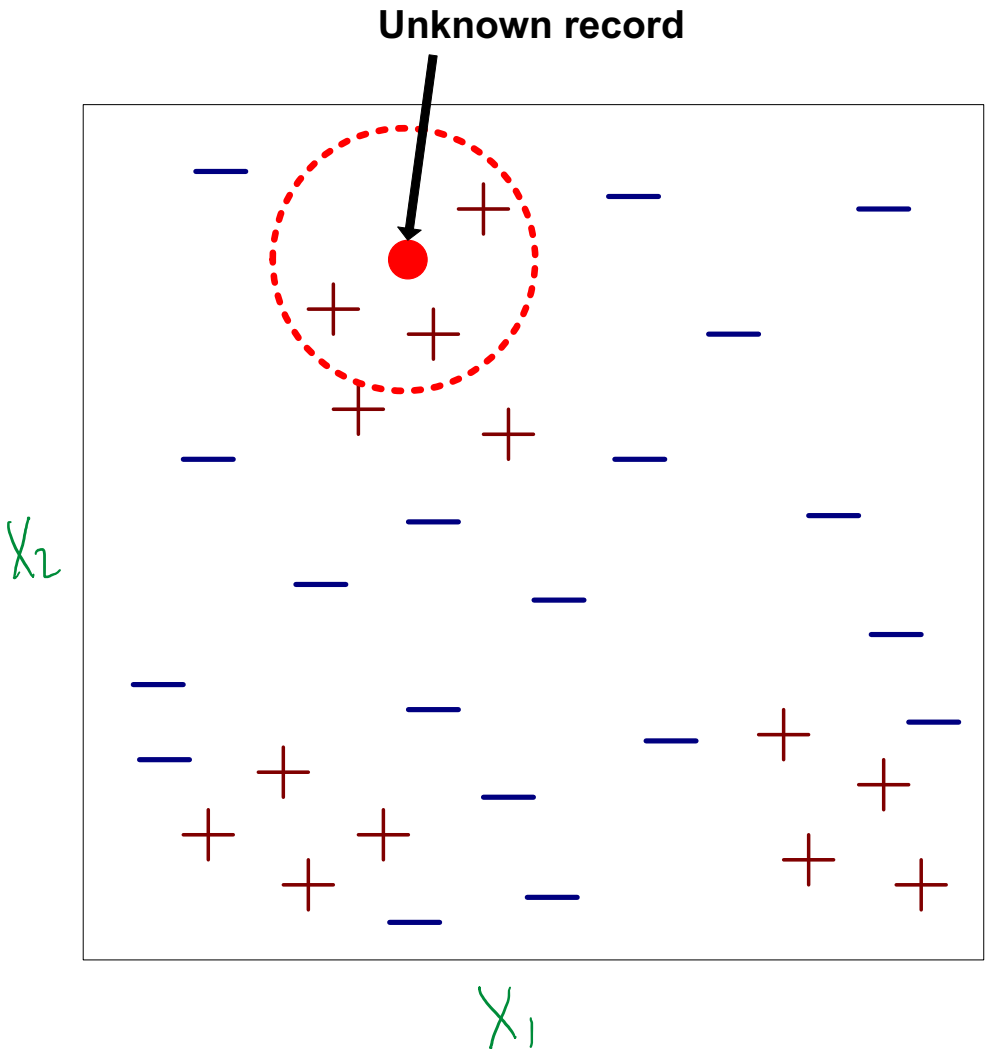
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



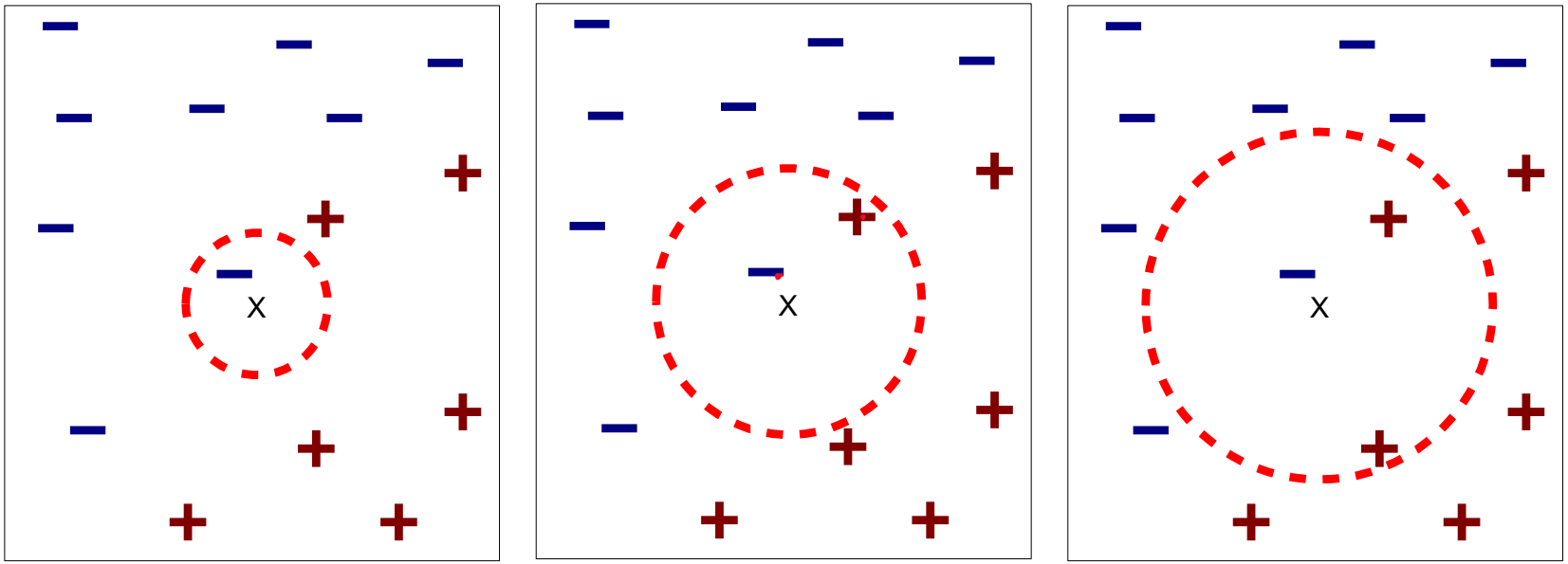
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance **Metric** to compute distance between records
  - The value of ***k***, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

KNN Classifier Algorithm

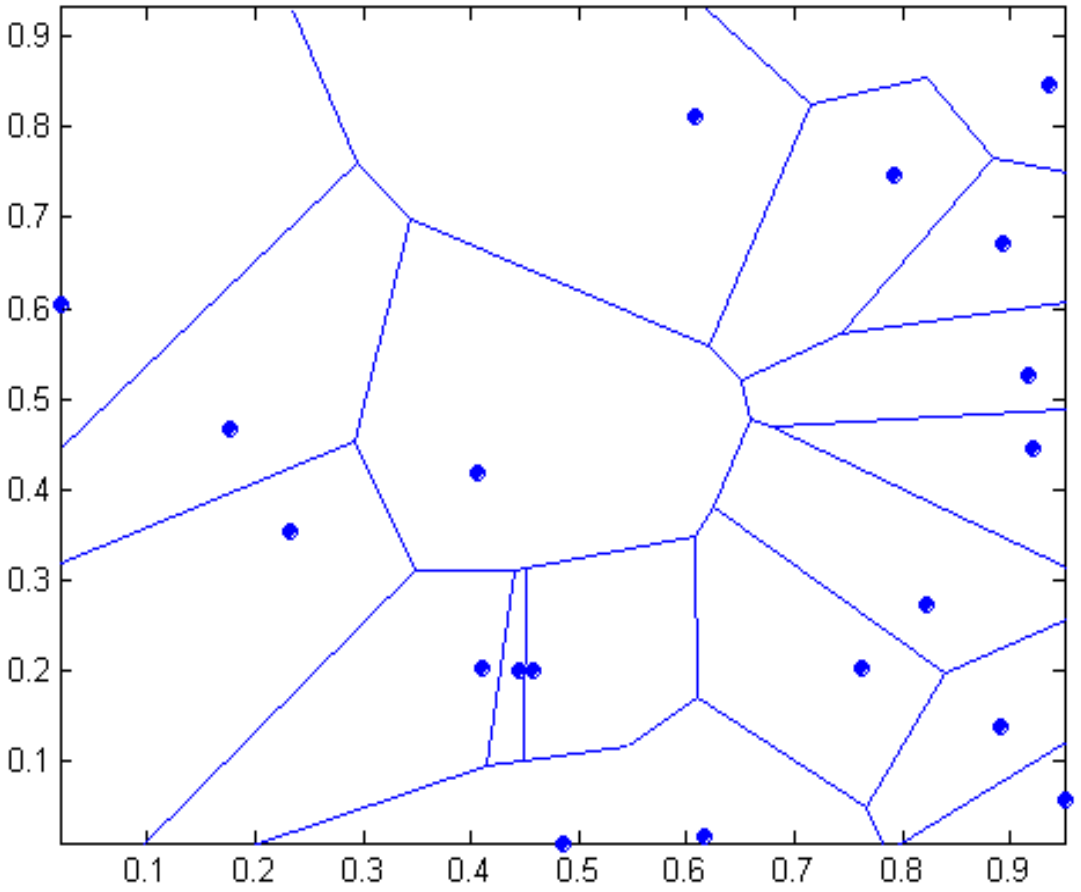
Algorithm 5.2 The k-nearest neighbor classification algorithm.

1: Let k be the number of nearest neighbors and D be the set of training examples.
2: **for** each test example $z = (\mathbf{x}', y')$ **do**
3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.
4: Select $D_z \subseteq D$, the set of k closest training examples to z.
5: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6: **end for**

1 nearest-neighbor

Voronoi Diagram

The points are given beforehand. Then we compute the Voronoi diagram. Each region has exactly a point called "seed". Each region contains the set of all points in the plane such that they are closer to the seed of that region.



Nearest Neighbor Classification

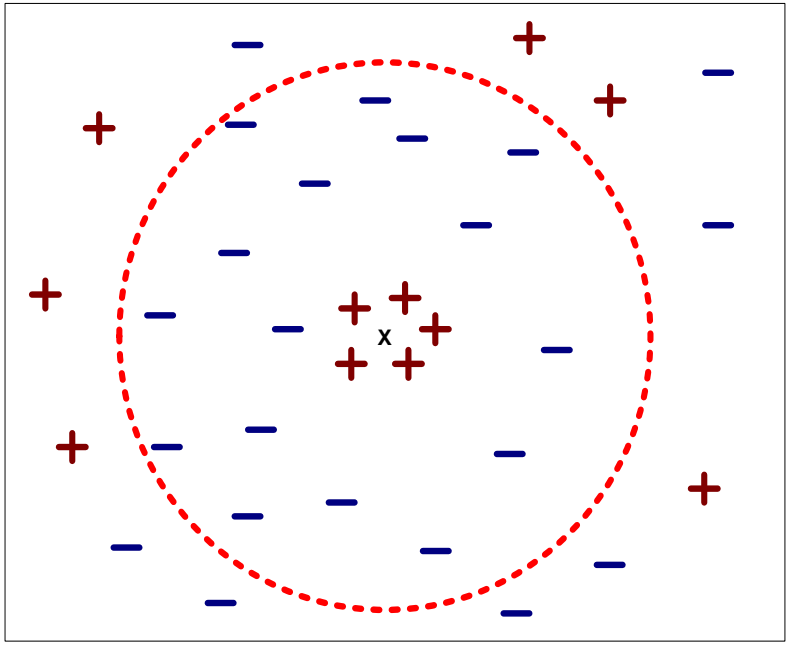
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points. Why?
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

- Can produce wrong predictions unless an appropriate proximity measure and data preprocessing steps are taken (normalization) What happens if we don't normalize the attributes?
- Suppose, for example that you have a dataset:

Id	Age	Wealth ($)	Class
1	67	170,000,000	1
2	45	67,000	0
3	23	2,500,000	0
4	43	169,000	1

- And you wish to classify (89, 78, 10000), then, if you don't normalize your data, then wealth will dominate the distance measure.

Nearest Neighbor Classification…

- What happens in this scenario?
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

- Scaling issues
  - **Attributes may have to be scaled** to prevent distance measures from being dominated by one of the attributes

Nearest Neighbor Classification…

- k-NN classifiers are **lazy learners**
  - They **do not build models explicitly** (unlike eager learners such as decision tree induction and rule-based systems)
  - They use specific training instances to make predictions without a model => Instance-based learning
    - Classifying unknown records can be expensive.
    - Need to compute individual proximity values
- k-NN classifiers make predictions based on local information
  - Susceptible to noise
- Can produce arbitrarily shaped decision boundaries
  - A more flexible model representation

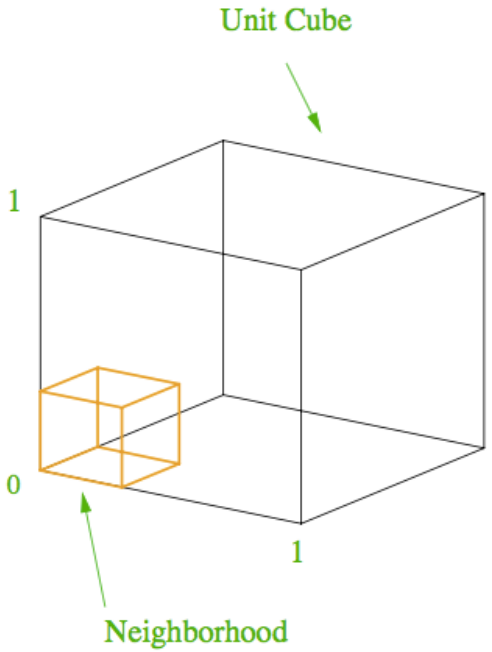
Nearest Neighbor Classification…

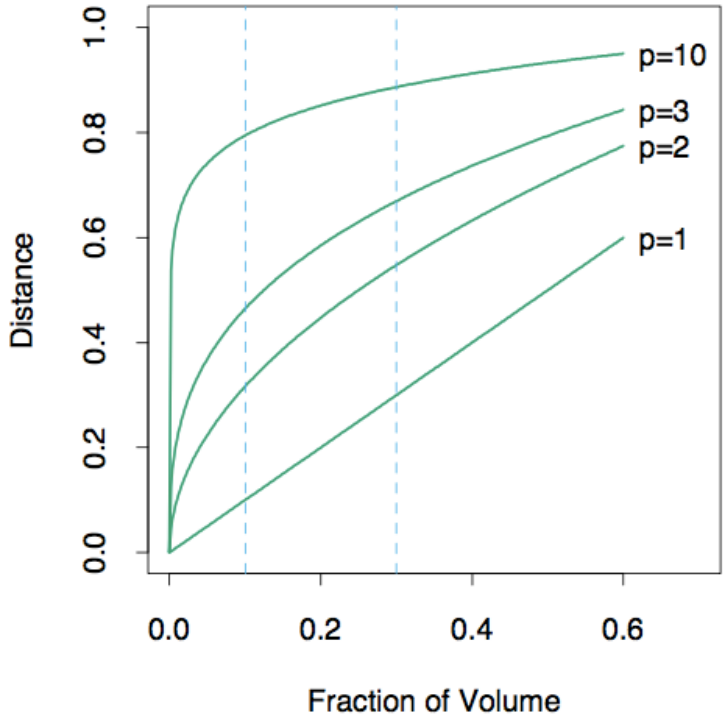
- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality

Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful

In the figure on the right *p* is the dimension of the space where the cube lives.
The plot says how large does each side of the yellow cube need to be in order to obtain a fraction x of the volume of the unit cube





The figure in this slide was taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Classifiers

Bayesian Classifiers

Probability Concepts Review

- Prior, conditional and joint probability for random variables
  - Prior probability: P(X)
  - Conditional probabilities: $P(X_1|X_2)$, $P(X_2|X_1)$
  - Joint probability: $X=(X_1,X_2)$, $P(X) = P(X_1,X_2)$
  - Independence: $P(X_2|X_1) = P(X_2)$, $P(X_1,X_2)= P(X_2)P(X_1)$
- Bayes theorem:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} \qquad \text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}$$

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time: P(S|M)
  - Prior probability of any patient having meningitis is 1/50,000: P(M)
  - Prior probability of any patient having stiff neck is 1/20: P(S)

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables
- **Given a record with attributes (Age=Senior, Wealth='> 150M')**
  - **Goal is to predict class 'Class'**
  - Specifically, we want to find the value of 'Class' such that…

Id	Age	Wealth ($)	Class
1	Senior	> 150 M	1
2	Senior	< 100K	0
3	Young Adult	< 175M and >= 100K	0
4	Adult	< 175M and >= 100K	1

Key Idea 1: MAP

It maximizes P(Class| Age=Senior, Wealth='>150M')

This is called MAP (*Maximum A Posteri*) estimation

Bayesian Classifiers

Id	Age	Wealth ($)	Class
1	Senior	> 150 M	1
2	Senior	< 100K	0
3	Young Adult	< 175M and >= 100K	0
4	Adult	< 175M and >= 100K	1

Key Idea 1: MAP

It maximizes P(Class| Age=Senior, Wealth='>150M')

Bayesian Classifiers

- In general:

- Given a record with attributes (X_1, X_2,…,X_p)
  - Goal is to predict class Y
  - Specifically, we want to find the value of Y that **maximizes** P(Y| X_1, X_2,…,X_p)

- Can we estimate P(Y| X_1, X_2,…,X_p) directly from data? How?

Bayesian Classifiers

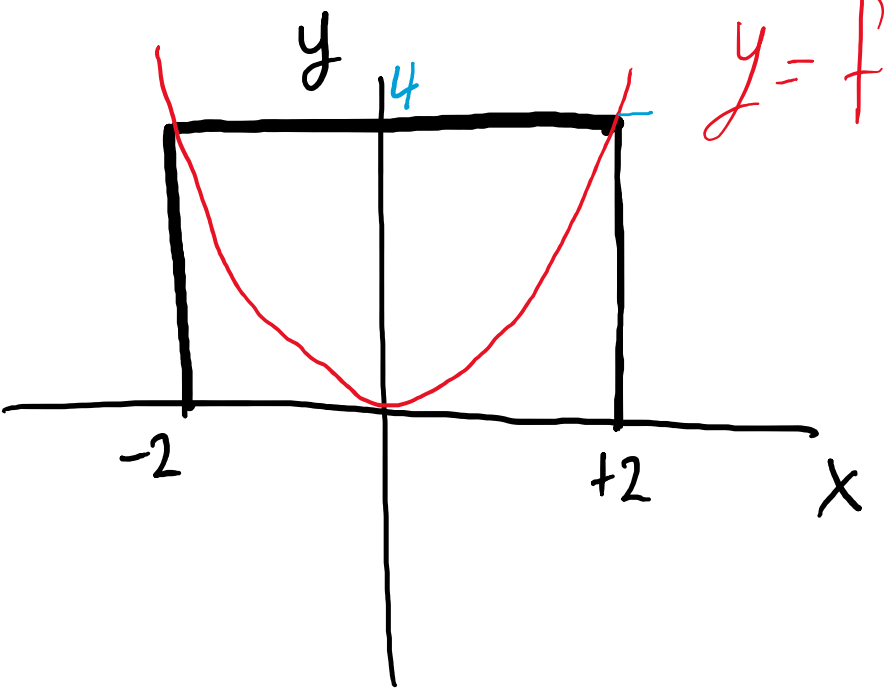
- Approach:
  - compute the posterior probability $P(Y \mid X_1, X_2, \ldots, X_p)$ for all values of Y using the Bayes theorem

$$P(Y \mid X_1, X_2, \ldots, X_p) = \frac{P(X_1, X_2, \ldots, X_p \mid Y) P(Y)}{P(X_1, X_2, \ldots, X_p)}$$

  - Choose value of Y that maximizes
    $P(Y \mid X_1, X_2, \ldots, X_p)$
  - Equivalent to choosing value of Y that maximizes
    $P(X_1, X_2, \ldots, X_p \mid Y)\, P(Y)$

Key Idea 2: Bayes theorem

- How to estimate $P(X_1, X_2, \ldots, X_p \mid Y)$?



$y = f(x) = x^2$

$\max_{x} f(x) = 4.$

s.t $x \in [-3, 2]$

$\arg\max_{x} f(x) = 2.$

s.t.

$x \in [-2, 2)$

Conditional Independence

- **X** and **Y** are **conditionally independent** given **Z** if **P(X|Y,Z) = P(X|Z)**

- Example: Arm length and reading skills
  - A young child has shorter arm length and limited reading skills, compared to adults.
  - If **age is fixed**, there is no apparent relationship between arm length and reading skills.

P(Arm Length, Reading Skill) ≠ P(Arm Length) P(Reading Skill)

  - Arm length and reading skills are conditionally independent **given age**

P(Arm Length, Reading Skill | Age) = P(Arm Length | Age) P(Reading Skill | Age)

Naïve Bayes Classifier

- **Assume independence among attributes** X_i when class is given. In other words, the **attributes X_i are conditionally independent given the Class attribute** Y:

$$P(X_1, X_2, \ldots, X_p \mid Y_j) = P(X_1 \mid Y_j) P(X_2 \mid Y_j) \cdots P(X_p \mid Y_j)$$

$$= \prod_{i=1}^{p} P(X_i \mid Y_j)$$

Conditional Independence Assumption

Key Idea 3: Conditional Ind. Assumption

  - Can estimate $P(X_i \mid Y_j)$ for all X_i and Y_j **from the data**

  - New point is classified to Y_j if $P(Y_j) \prod P(X_i \mid Y_j)$ is maximal

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(Y) = N_c/N$
  - where N_c is the number of instances belonging to class Y, and N is the total number of instances in the training set
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10
- For **discrete attributes**:

$$P(X_i \mid Y_k) = |X_{ik}| / N_{ck}$$

  - where $|X_{ik}|$ is number of instances having attribute X_i and belonging to class Y_k, and N_{ck} is the number of instances belonging to class Y_k.
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For **continuous attributes**:
  - Discretization: Partition the range into bins
    - Replace continuous value with bin value (i.e., its corresponding discrete interval)
    - $P(X_i|Y=y)$: Compute the fraction of training records belonging to class Y=y that fall within the corresponding interval for X_i.
  - Probability density estimation:
    - Assume attribute follows a **normal distribution**
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(X_i|Y=y)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(X_i \mid Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(X_i \mid Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (X_i, Y_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Class Exercise

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given the Test Record:

X = (Refund = No, Marital Status = Divorced, Income = 120K)

Predict its class using a Naive Bayes Classifier

$$P(X_i \mid Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

Bayes rule:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Naïve Bayes Assumption:

$$P(X_1, X_2, \ldots, X_p \mid Y_j) = P(X_1 \mid Y_j)P(X_2 \mid Y_j) \cdots P(X_p \mid Y_j)$$

$$= \prod_{i=1}^{p} P(X_i \mid Y_j)$$

Example of Naïve Bayes Classifier

Given a Test Record:

X = (Refund = No, Divorced, Income = 120K)

Predict its class using a Naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes)=1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No) = 1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/3
P(Marital Status=Divorced|Yes) = 1/3
P(Marital Status=Married|Yes) = 0
P(No) = 7/10
P(Yes)=3/10

For taxable income:
If class=No: sample mean = 110
sample variance = 2975
If class=Yes: sample mean = 90
sample variance = 25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Divorced| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 1/7 × 0.0072 = 0.0006

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Divorced| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 1/3 × 1.2 × 10^{-9} = 4 × 10^{-10}

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Issues with the Naïve Bayes Classifier

Consider the same table, but with Tid=7 deleted:

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

P(Refund=Yes|No) = 2/6
P(Refund=No|No) = 4/6
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes)=1
P(Marital Status=Single|No) = 2/6
P(Marital Status=Divorced|No) = 0
P(Marital Status=Married|No) = 4/6
P(Marital Status=Single|Yes) = 2/3
P(Marital Status=Divorced|Yes) = 1/3
P(Marital Status=Married|Yes) = 0
P(No) = 7/10
P(Yes)=3/10

For taxable income:
If class=No: sample mean = 91
sample variance = 685
If class=Yes: sample mean = 90
sample variance = 25

Given **X = (Refund=Yes, Divorced, Income=120K),**
$P(X|Evade = No) = 2/6 \times 0 \times 0.0083 = 0$
$P(X|Evade = Yes) = 0 \times 1/3 \times (1.2 \times 10^{-9}) = 0$

Naïve Bayes cannot classify X!

Issues with the Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of levels of A_i

p: prior probability of the class

m: parameter

N_c: number of instances in the class c

N_{ic}: number of instances having attribute value A_i in class c

If you don't know p, then choose p:=1 / (Num. different possible values that Ai can assume)

Issues with the Naïve Bayes Classifier

Suppose that we use Laplace's Estimator to compute P(X | Evade):

$$\text{Laplace: } P(\text{M. Status} = \text{Div.} \mid \text{Evade} = \text{No}) = \frac{N_{\text{M. Status=Div., Evade=No}} + 1}{N_{\text{Evade=No}} + \text{Num. Values M. Status}} = \frac{0+1}{6+3} = \frac{1}{9}$$

$$\text{Laplace: } P(\text{Refund} = \text{Yes} \mid \text{Evade} = \text{Yes}) = \frac{N_{\text{Refund=Yes, Evade=Yes}} + 1}{N_{\text{Evade=Yes}} + \text{Num. Values Refund}} = \frac{0+1}{3+2} = \frac{1}{5}$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given **X = (Refund=Yes, Divorced, Income=120K),**

P(X|Evade = No) = 2/6 × **(1/9)** × 0.0083 = 0.0003

P(X|Evade = Yes) = **(1/5)** × 1/3 × (1.2 × 10^{-9}) = 0

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to **irrelevant** attributes
- **Independence assumption** may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

Classifiers

Logistic Regression

Logistic Regression: Motivation

- Consider the default dataset, for predicting if a person will default on his/her debt based on other variables like age, balance, marital status, etc.

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

Default Dataset

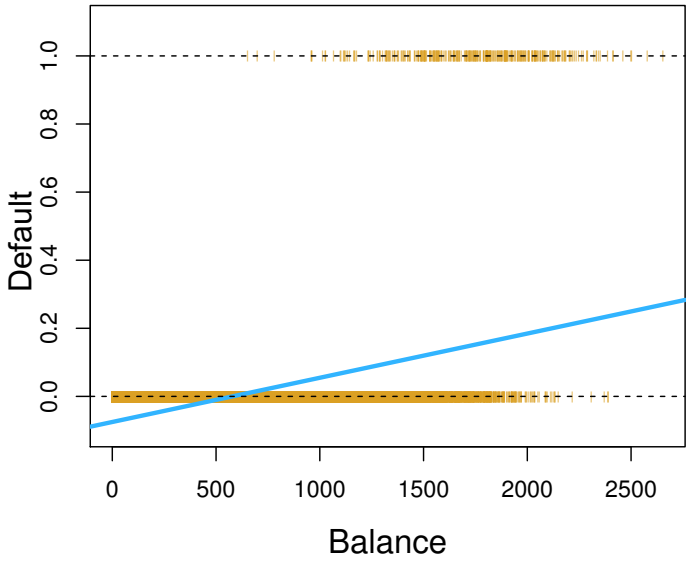
- Suppose that we want to classify people, if we know their balance.

Logistic Regression: Motivation

- Basic idea:
  - Assume that there are two classes 1 (default) and 0
  - We want to use a linear function to predict default, as we did in linear regression:

$$\beta_0 + \beta_1 Balance$$

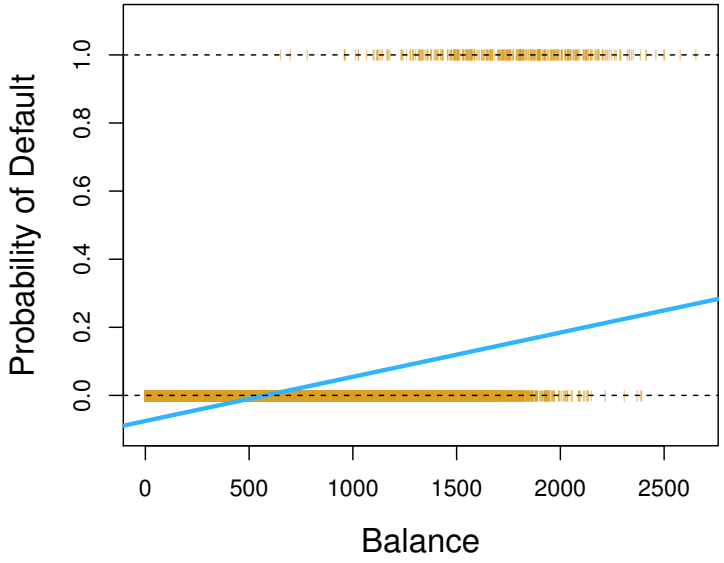
- Problems:
  - A linear function like that returns a continuous number instead of discrete classes (0 and 1)



Logistic Regression: Motivation

- Assuming that there are two classes 1 and 0, estimate: $p(X) = p(Default = 1 \mid X) = \beta_0 + \beta_1 Balance$
  - How can we estimate that probability?
    - We could use **linear regression**

Key Idea 1: Estimate the probability of the class

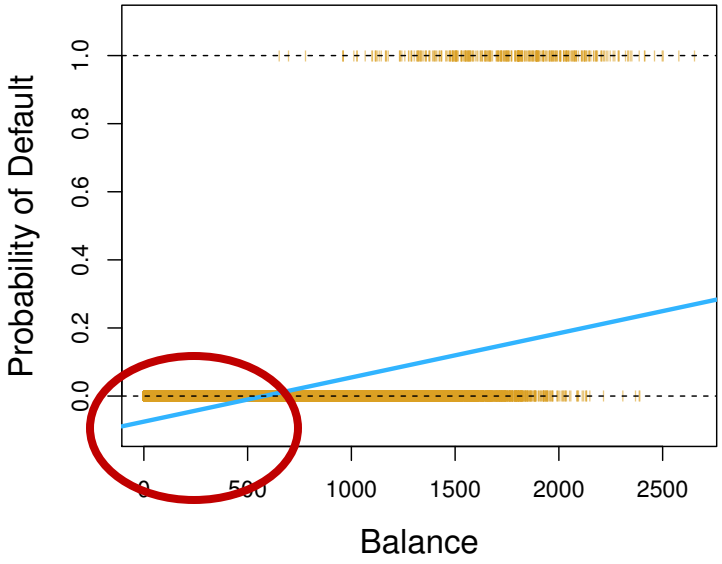


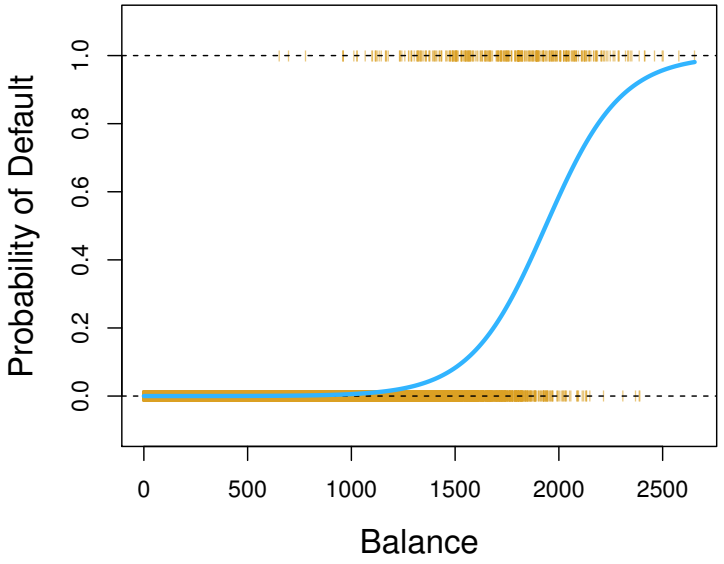
Logistic Regression: Motivation

- If we use Linear Regression, what are the potential problems?

$$p(X) = p(\text{Default} = 1 \mid X) = p(\text{Default} = 1 \mid \text{Balance})$$
$$= \beta_0 + \beta_1 \text{Balance}$$

For certain values of X, we can get negative probabilities



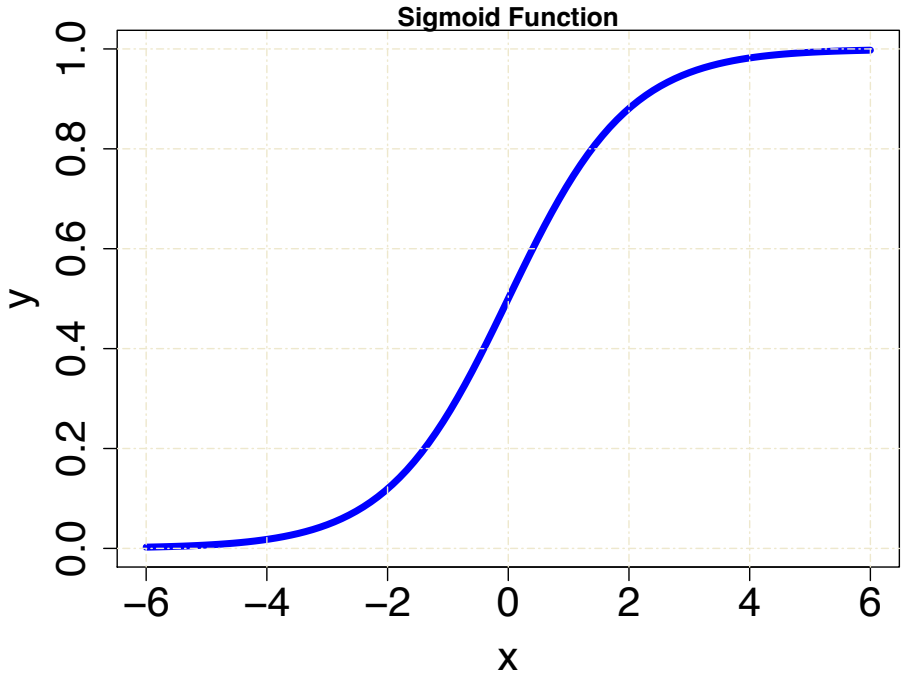


How can we solve the problem of negative probabilities?

Logistic Regression: Motivation

- To address this issue of negative probabilities, use a sigmoid function *f*, which has range in [0,1]:

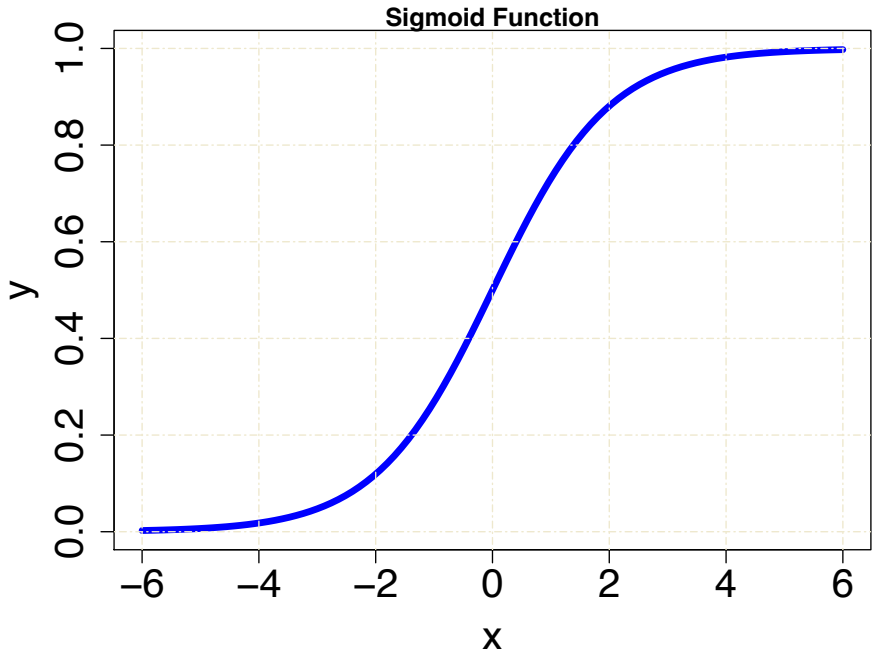
$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



Logistic Regression: Motivation

- The sigmoid function has a nice property:
  - It's differentiable, and its derivative has a nice form:

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



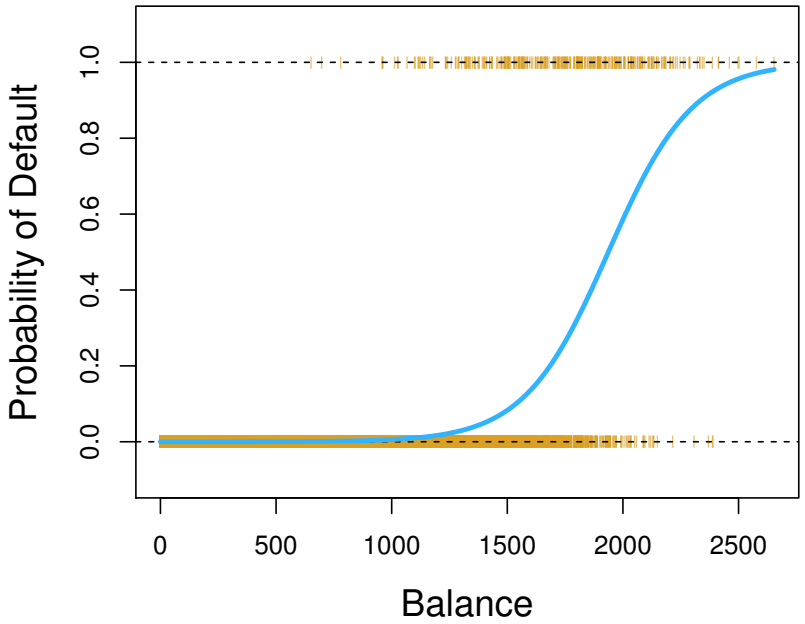
$$\begin{aligned}
\frac{d\sigma(x)}{dx} &= \frac{d}{dx}\left(\frac{e^x}{1+e^x}\right) \\
&= \frac{e^x(1+e^x) - (e^x)^2}{(1+e^x)^2} \\
&= \frac{e^x}{(1+e^x)^2} \\
&= \sigma(x) * (1-\sigma(x))
\end{aligned}$$

Logistic Regression: Motivation

- If we use the sigmoid function:

$$
\begin{aligned}
p(\text{Default} = 1 \mid X) &= p(\text{Default} = 1 \mid \text{Balance}) \\
&= \sigma(\text{Balance}) \\
&= \frac{1}{1 + e^{-(\beta_0 + \beta_1 \text{Balance})}}
\end{aligned}
$$

Key Idea 2: Sigmoid of a polynomial



Logistic Regression

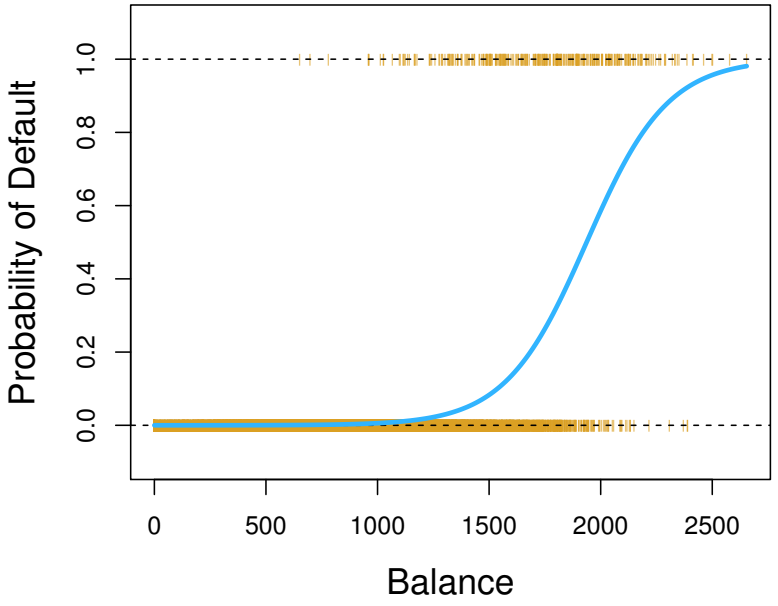
- To address this issue of negative probabilities, use a logistic function f, which has range from [0,1]:

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} = \sigma(\beta_0 + \beta_1 x).$$

- This is equivalent to saying: $\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$

- And also to:

$$\ln\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$



We replaced Balance with X in these formulas

Logistic Regression

- This is equivalent to saying:

odds

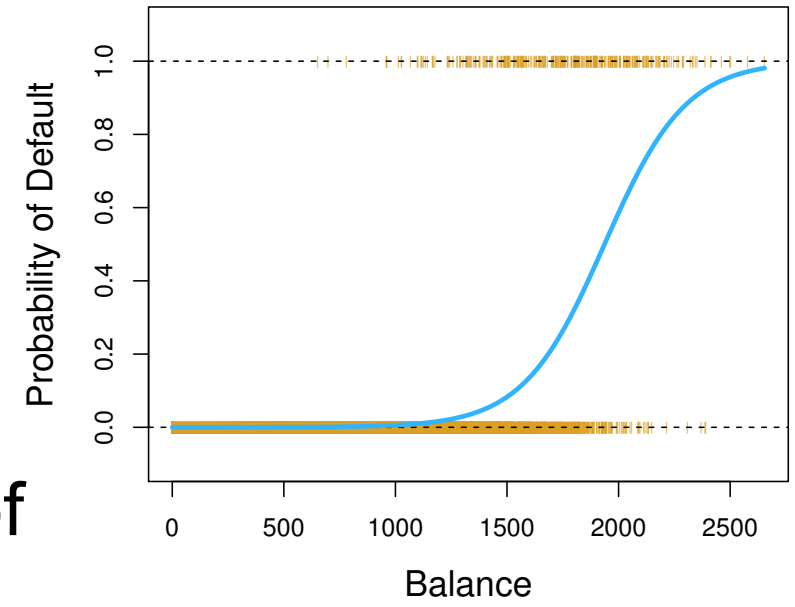
$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

- And also to:

Log odds

$$\ln\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

- What is the interpretation of the coefficients in linear regression?



We replaced Balance with X in these formulas

Logistic Regression: Prediction

- Once you have found β_0, β_1 , it is very easy to classify a new record. For example, if

$$\beta_0 = -10.7, \beta_1 = 0.0055$$

- Then, if we see a person with a balance of 1,000, our prediction will be:

$$p(\text{Balance}) = \frac{1}{e^{-(\beta_0+\beta_1 \text{Balance})}} = \frac{1}{1+e^{-(-10.7+0.0055\times 1000)}} = 0.006$$

- And since this number is less than 0.5, we conclude that our model's prediction is that this person won't default.

Logistic Regression

- How do we fit a logistic regression model? In other words, how do we find the β_0, β_1 coefficients in the following formula?

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

- Just like with linear regression, we must define a loss function $L(X) = L(\beta_0, \beta_1)$

And then use gradient descent or another optimization algorithm to find "good" values for β_0, β_1

Logistic Regression: Loss Function

- Remember the dataset for this section:

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

- Suppose that we have two classifiers that output probabilities:

Id	Predicted Probability of Default
1	0.6
2	0.2
3	0.7

Classifier 1

Id	Predicted Probability of Default
1	0.15
2	0.98
3	0.99

Classifier 2

Which performs better on the training set?

Classifier 2

Logistic Regression: Loss Function

- A loss function that captures the intuition from before is the **binary cross-entropy:**

$$L(C) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log C(x_i) + (1 - y_i) \log (1 - C(x_i))$$

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

- **L(C) is the loss of classifier C on the dataset *D* being evaluated**
- **n is the number of rows in the dataset D (here 3)**
- **y_i is the true class of the ith row in the dataset**
- **$C(x_i)$ is the prediction of classifier C on the ith row.**

Id	Predicted Probability of Default
1	0.6
2	0.2
3	0.7

Classifier 1

Id	Predicted Probability of Default
1	0.15
2	0.98
3	0.99

Classifier 2

L(C1) = (-1/3) * (log2(1 - 0.6) + log2(0.2) + log2(0.7))
= 1.39

L(C2) = (-1/3) * (log2(1 - 0.15) + log2(0.98) + log2(0.99))
= 0.009

Classifier 2 has a smaller loss; therefore, it is better!

Whenever we use the word “entropy”, we use log base 2. Tensorflow v2 uses ln (base e) instead.

Logistic Regression: Loss Function

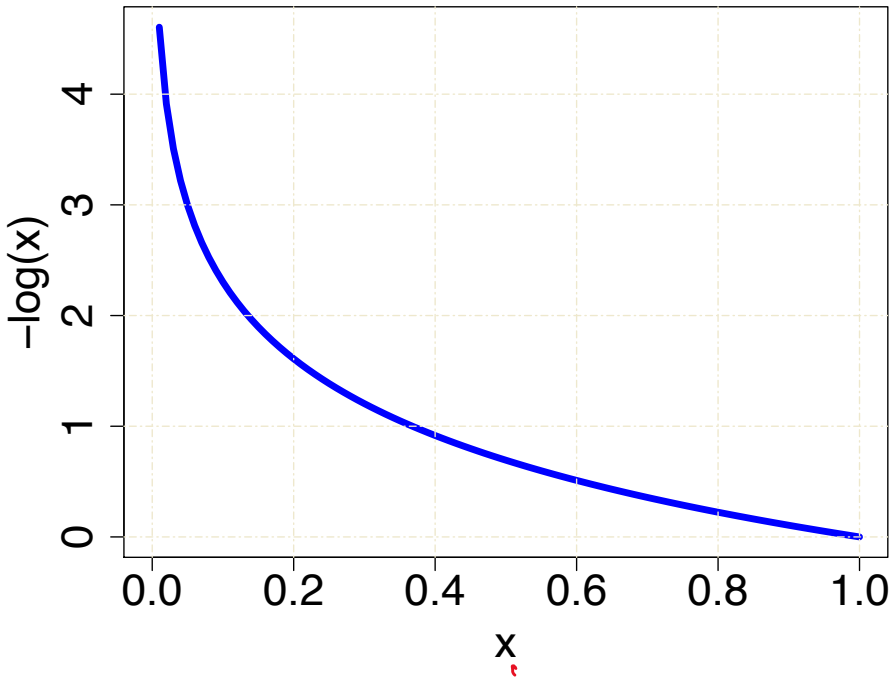
- Why does the **binary cross entropy** work?

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

$$L(C(x)) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log C(x_i) + (1 - y_i) \log\left(1 - C(x_i)\right)$$

Id	Predicted Probability of Default
1	0.6
2	0.2
3	0.7

Classifier 1



- If the sample has true label 1 and the classifier predicts a number close to one, we want the loss to be: Small
- If the sample has true label 1 and the classifier predicts a number close to zero, we want the loss to be **Large**
- Analogously if the true label is 0.

Logistic Regression: Loss Function

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

- OK, we have the binary cross entropy function for logistic regression.

$$L(C(x)) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log C(x_i) + (1-y_i)\log\left(1 - C(x_i)\right)$$

- In the case of logistic regression, remember that the output is given by

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

- So the binary cross entropy is really:

$$L(\beta_0,\beta_1) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}} + (1-y_i)\log\left(1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}}\right)$$

Logistic Regression: Loss Function

- So the binary cross entropy loss for logistic regression is

$$L(\beta_0, \beta_1) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}} + (1-y_i)\log\left(1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}}\right)$$

- What do we do after we have our loss function?
  - **Find the gradient**
  - **Then use an optimization method like gradient descent or Newton's method (better).**

Logistic Regression: Gradient

- With some vector calculus, we can find the gradient of the binary cross-entropy loss for logistic regression

$$L(\beta_0, \beta_1) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}} + (1-y_i)\log\left(1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}}\right)$$

- The gradient is

$$\nabla L(\beta_0, \beta_1) = H^T(p-y)$$

Where

- $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_n})$ **is the vector of classes (1 or 0 each)**
- $\boldsymbol{p} = (\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n})$ **is the vector with the probabilities (of default) for each example**
- **H is the nx2 matrix containing 1s in the first column, and the balances of the examples in the second column.**

Logistic Regression: Gradient

- The gradient is

$$\nabla L(\beta_0, \beta_1) = H^T (p - y)$$

H^T: $(1+p) \times n$; $(p - y)$: $n \times 1$

Where

- $y = (y_1, y_2, \ldots, y_n)$ is the vector of classes (1 or 0 each)
- $p = (p_1, p_2, \ldots, p_n)$ is the vector with the probabilities (of default) for each example
- H is the nx2 matrix containing 1s in the first column, and the balances of the examples in the second column.

- In the example before:

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

$$H = \begin{pmatrix} 1 & 2500 \\ 1 & 535 \\ 1 & 3000 \end{pmatrix}_{n \times (1+p)} \quad p = \begin{pmatrix} \frac{1}{1+e^{-(\beta_0+\beta_1(2500))}} \\ \frac{1}{1+e^{-(\beta_0+\beta_1(535))}} \\ \frac{1}{1+e^{-(\beta_0+\beta_1(3000))}} \end{pmatrix}_{n \times 1} \quad y = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}_{n \times 1}$$

Logistic Regression in R

```
df.name <- data.frame(balance = c(2500, 535, 3000),
                      default = c(0,1,1))

model <- glm(default ~ ., data = df.name, family =
"binomial")
```

```
```{r}
model <- glm(default ~ ., data = df.name, family = "binomial")
summary(model)
```
```

```
Call:
glm(formula = default ~ ., family = "binomial", data = df.name)

Deviance Residuals:
      1       2       3
-1.3706  0.5136  1.1529

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.3719254  3.6760762   0.645    0.519
balance     -0.0007714  0.0014624  -0.527    0.598

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3.8191  on 2  degrees of freedom
Residual deviance: 3.4716  on 1  degrees of freedom
AIC: 7.4716

Number of Fisher Scoring iterations: 4
```

These are

β_0, β_1

```
Command Window

>> logistic_reg(H,z, x0, lambda, max_iter)

num_iter =

   100000000


ans =

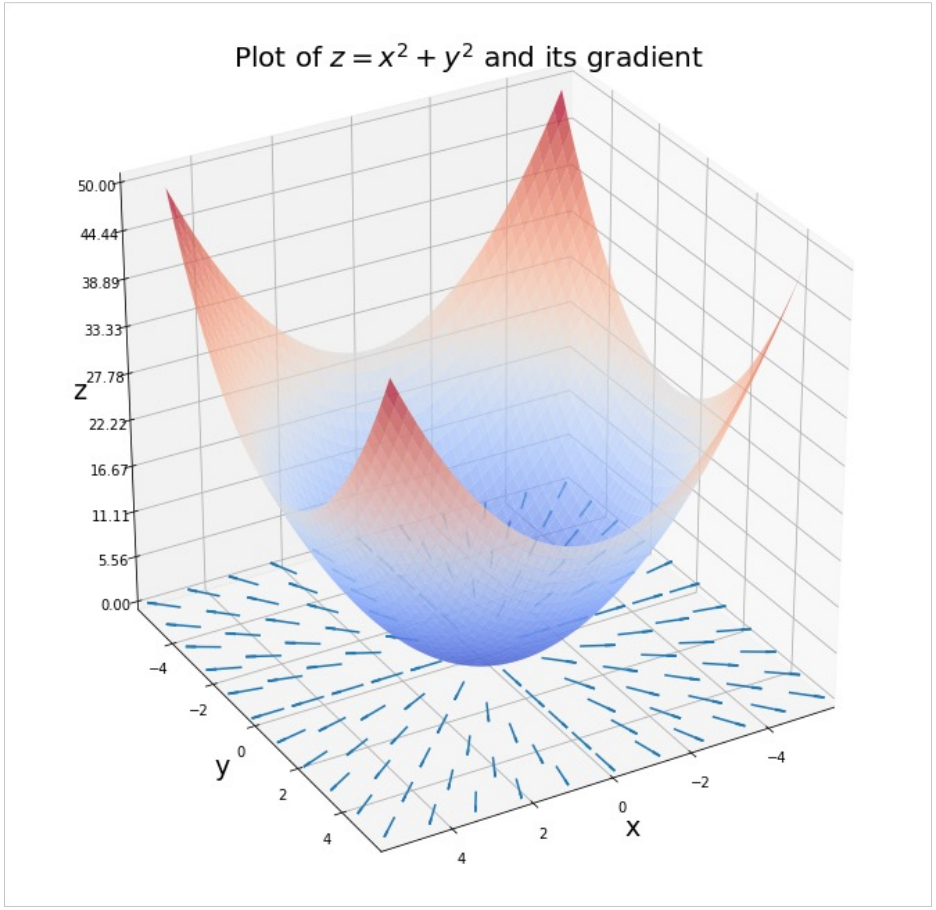
    2.3374
   -0.0008
```

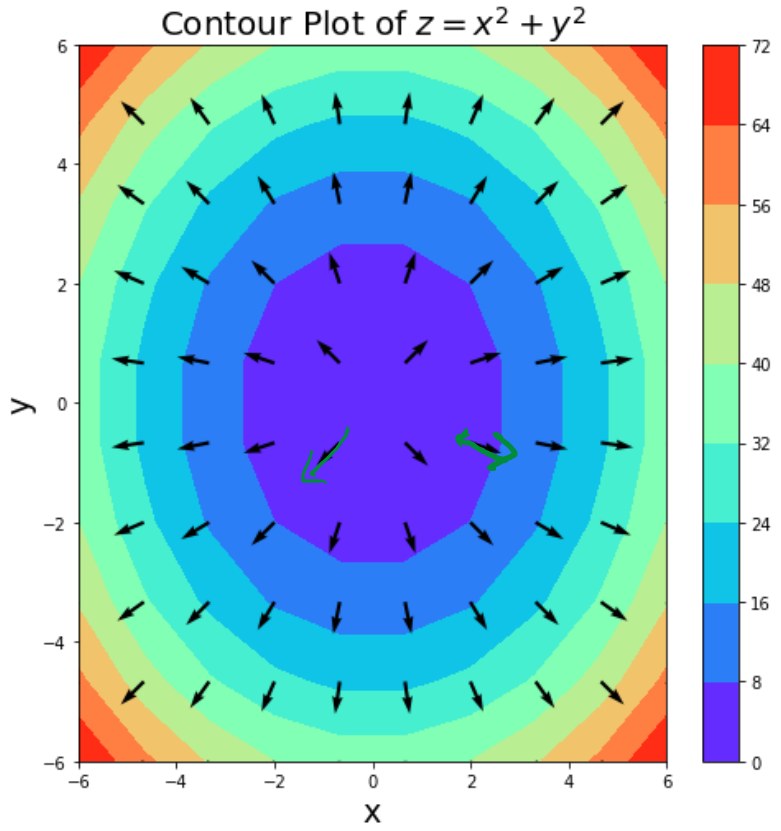
Gradient Descent

In these slides we will present a very brief review of the Gradient Descent method

Gradient of a Function

- Remember the gradient is $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$





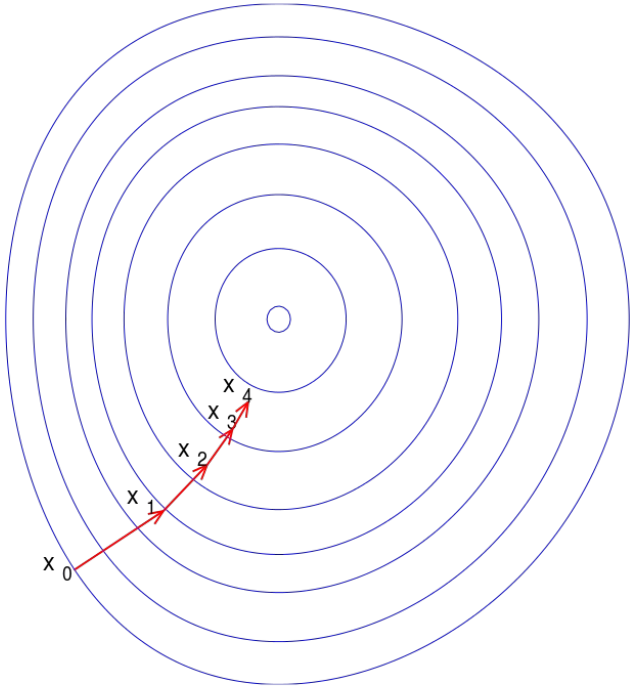
(General) Gradient Descent Algorithm

- **Input**: $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}, x_0 \in \mathbb{R}^n, \lambda \in \mathbb{R}$ (Initial guess; Learning rate)
- **Output**: A local minimum of f

- For k = 1, 2, …:
  - Compute the gradient ∇f
  - Take the step

$$x_{k+1} = x_k - \lambda(\nabla f)(x_k)$$

  - Test for convergence. If yes, then exit.



The gradient is always normal to the level sets

Gradient Descent Algorithm for Linear Regression

- **Input**: $E(x) : \mathbb{R}^n \longrightarrow \mathbb{R}, x_0 \in \mathbb{R}^n, \lambda > 0 \in \mathbb{R}, \epsilon > 0 \in \mathbb{R}$ (Initial guess; Learning rate; Tolerance)
- **Output**: A local minimum of f

- For k = 0, 1, 2, …:
  - Compute the gradient at x_k

$$\nabla E(x_k) = 2(H^T \cdot H) \cdot x_k - 2H^T z$$

  - Take the step

$$x_{k+1} = x_k - \lambda(\nabla E)(x_k)$$

  - Test for convergence:
    - If $\|x_{k+1} - x_k\| < \epsilon$ for a small fixed epsilon, then exit.

Gradient Descent Algorithm for Logistic Regression

- **Input**: $E(x) : \mathbb{R}^n \longrightarrow \mathbb{R}, x_0 \in \mathbb{R}^n, \lambda > 0 \in \mathbb{R}, \epsilon > 0 \in \mathbb{R}$ (Initial guess, Learning rate, Tolerance)
- **Output**: A local minimum of f

- For k = 0, 1, 2, …:
  - Compute the gradient at x_k

$$\nabla E(w) = H^T(p(x_k) - y)$$

p depends on x_k

  - Take the step

$$x_{k+1} = x_k - \lambda(\nabla E)(x_k)$$

  - Test for convergence:
    - If $\|x_{k+1} - x_k\| < \epsilon$ for a small fixed epsilon, then exit.

Where $y = (y_1, y_2, \ldots, y_n)$ is the vector of classes (1 or 0 each), $p = (p_1, p_2, \ldots, p_n)$ is the vector with the probabilities (of default) for each example, and H is the nx2 matrix containing 1s in the first column, and the balances of the examples in the second column.

Gradient Descent for Logistic Regression: Step by Step

$$p(\text{default} \mid \text{balance}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \text{balance})}}$$

id	Married	income	balance	default
1	No	23,000	2500	no
2	Yes	57,000	535	yes
3	Yes	29,000	3000	yes

$$x_0 = (\beta_0^{(0)}, \beta_1^{(0)})^T = (0,1)^T \qquad \lambda = 0.1 \qquad \epsilon = 10^{-12}$$

Iteration 1:

1) Compute the gradient of L (the loss) at x_0 = (1,1)

$$\nabla E(x_0) = H^T(p(x_0) - y)$$

$$H = \begin{pmatrix} 1 & 2500 \\ 1 & 535 \\ 1 & 3000 \end{pmatrix} \qquad H^T = \begin{pmatrix} 1 & 1 & 1 \\ 2500 & 535 & 3000 \end{pmatrix}$$

$$p(x_0) = \begin{pmatrix} \frac{1}{1+e^{-(0+1\cdot 2500)}} \\ \frac{1}{1+e^{-(0+1\cdot 535)}} \\ \frac{1}{1+e^{-(0+1\cdot 3000)}} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Gradient Descent for Logistic Regression: Step by Step

$$\nabla E(x_0) = H^T(p(x_0) - y) = \begin{pmatrix} 1 & 1 & 1 \\ 2500 & 535 & 3000 \end{pmatrix} \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right)$$

$$= \begin{pmatrix} 1 \\ 2500 \end{pmatrix}$$

2) Take the step

$$x_1 = x_0 - \lambda \nabla E(x_0)$$

$$= \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 1 \\ 2500 \end{pmatrix}$$

$$= \begin{pmatrix} -0.1 \\ -249 \end{pmatrix}$$

Gradient Descent for Logistic Regression: Step by Step

Iteration 2:

1) Compute the gradient of L (the loss) at x_1 = (-0.1,-249)

$$\nabla E(x_1) = H^T(p(x_1) - y)$$

$$= \begin{pmatrix} 1 & 1 & 1 \\ 2500 & 535 & 3000 \end{pmatrix} \left(\begin{pmatrix} \frac{1}{1+e^{-(-0.1-249\cdot 2500)}} \\ \frac{1}{1+e^{-(-0.1-249\cdot 535)}} \\ \frac{1}{1+e^{-(-0.1+-249\cdot 3000)}} \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right)$$

2) Take the step, and so on…

Gradient Descent for Logistic Regression

```
function [x] = logistic_reg(H, z, x0,lambda, max_iter)
%grad_reg Finds the logistic regression coefficients
%   Finds the logistic regression coefficients by using the
%   gradient method, using a step of lambda
%   and running no more than max_iter iterations.

% Set the initial point
x        = x0;
tol      = 1e-7;
% The error can be whatever as long as it is > 1e-7
error    = 1;
% Store the errors for plotting
errors   = zeros(max_iter,1);
num_iter = 0;

while error > tol && num_iter < max_iter
    % Residual or error
    residual = sigmoid(H*x) - z; % this is (p - y) in the slides

    % Compute the gradient at x = (beta0, beta1)'
    gradient = H' * residual; % H is phi in the slides

    % Take a step
    prev = x;
    x = x - lambda * gradient;

    % Compute the successive error
    %error = norm(prev-x,2);%(z - H*x)' * (z - H*x);
    error = residual' * residual

    % Increase the number of iterations
    num_iter = num_iter + 1;

    % Store the error
    errors(num_iter) = error;
end
```

Note: Logistic regression is usually implemented using a *different* algorithm called *iterative reweighted least squares* or *IRLS*. That other algorithm is based on Newton's method and uses the Hessian of the loss function (which is a generalization of the second derivative) to improve the convergence rate.

Side Note: Cross-Entropy

- The **cross entropy** of two discrete probability distributions *q* relative to *p* is defined as:

$$H(p,q) = -\sum_{x \in \text{Values}(x)} p(x) \log q(x)$$

- where values(x) is the set of values where either p(x) and q(x) are non-zero.

Logistic Regression: Additional Material

- **The following slides until the end of the section of Logistic Regression are optional**. They present an alternate way of deriving a loss function for logistic regression using the **log-likelihood function**.

Logistic Regression: Additional Material

- To fit a logistic regression model to your data, you need to find β_0, β_1, which are the ones that maximize the likelihood function:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

$$= \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

$p(x_i)$ is the probability of default

- where y_i is the class (1 or 0) of the ith row.
- For this, you can use any solver. These solvers may use optimization algorithms like gradient descent or Newton's method.

Logistic Regression: Additional Material

- To maximize the likelihood function:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

$$= \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

- You can instead minimize the (-1)*log-likelihood:

$$\log \ell(\beta_0, \beta_1) = -\sum_{i=1}^{n} \{y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))\}$$

- Using gradient descent $\nabla_{\beta_0, \beta_1} \log \ell = \sum_{i=1}^{n} (p(x_i) - y_i)\phi_i$

$$\phi_i = (1, balance)$$

Logistic Regression: Additional Material

- To minimize the negative log-likelihood:

$$\log \ell(\beta_0, \beta_1) = -\sum_{i=1}^{n} \{y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))\}$$

- You can use gradient descent with:

$$\nabla_{\beta_0, \beta_1} \log \ell = \sum_{i=1}^{n} (p(x_i) - y_i)\phi_i \qquad \phi_i = (1, balance)$$

- Or equivalently:

$$\nabla E(w) = \phi^T (p - y)$$

Where $y = (y_1, y_2, \ldots, y_n)$ is the vector of classes (1 or 0 each), $p = (p_1, p_2, \ldots, p_n)$ is the vector with the probabilities (of default) for each example, and ϕ is the nx2 matrix containing 1s in the first column, and the balances of the examples in the second column.

If ϕ is an nx2 matrix, and the vector (p-y) has dimensions nx1, then the gradient has dimensions 2x1