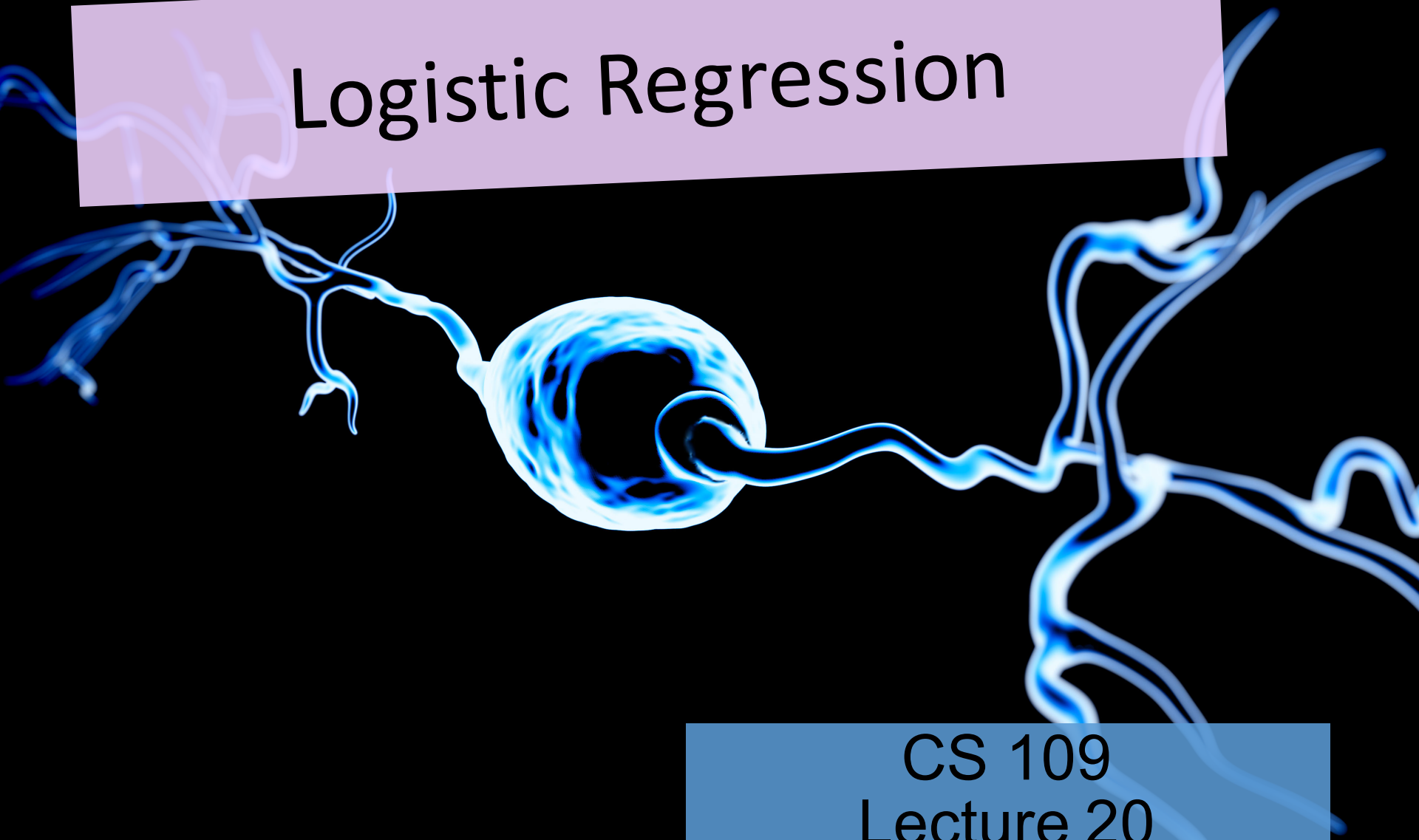
Logistic Regression

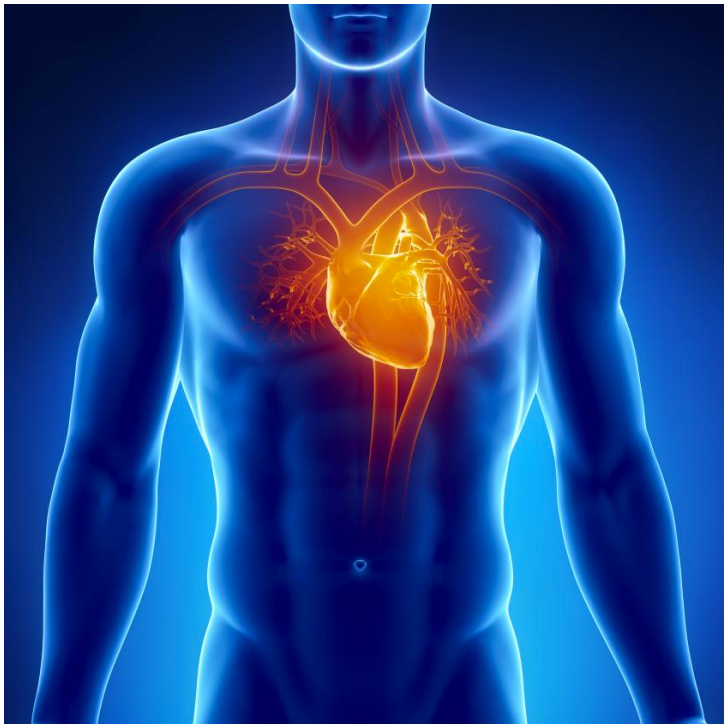


CS 109
Lecture 20
May 11th, 2016

Review

Classification Task

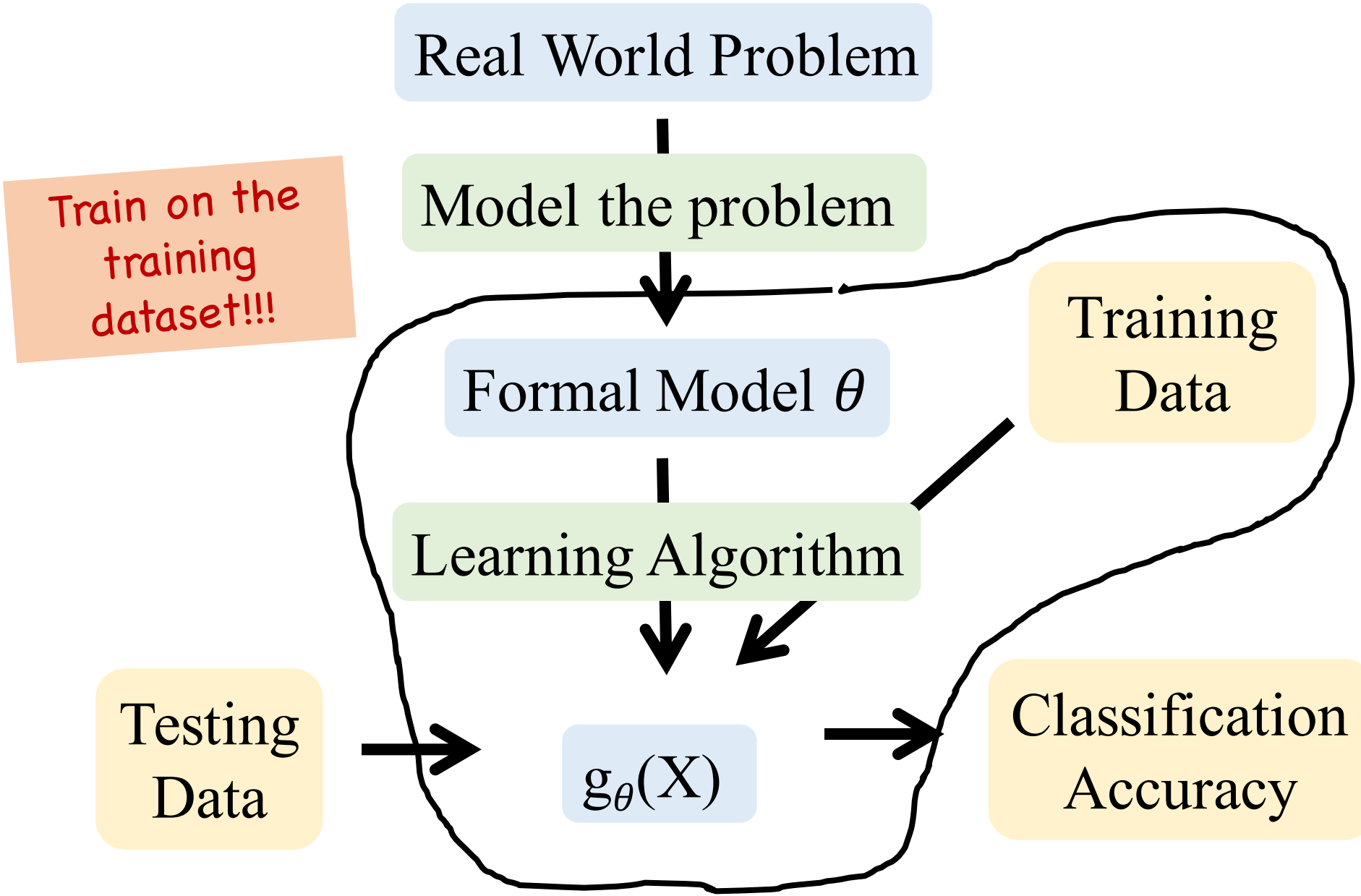
Heart



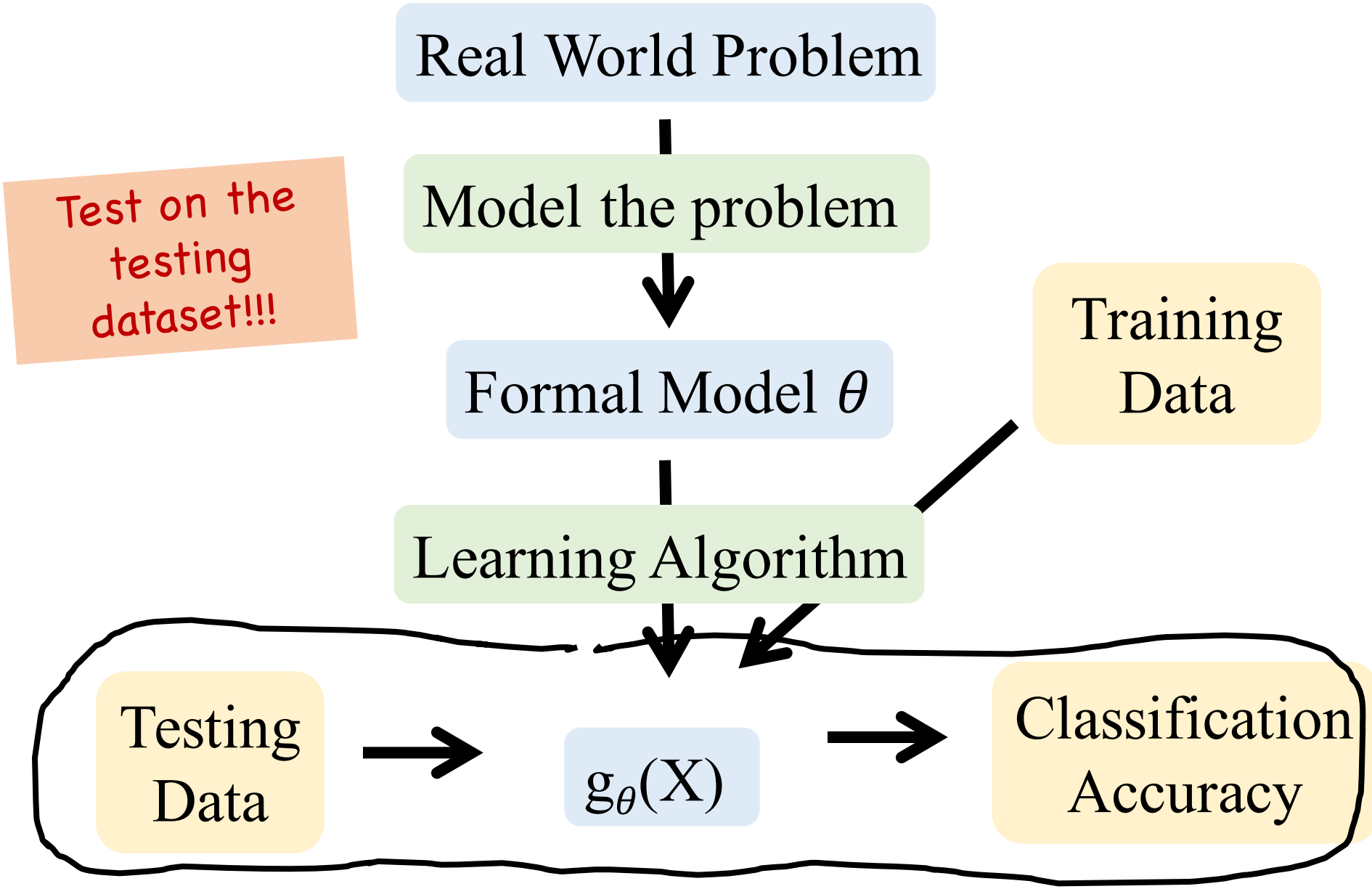
Ancestry

Netflix

Training


Real World Problem
Model the problem
Train on the training dataset!!!
Formal Model θ
Training Data
Learning Algorithm
Testing Data
$g_\theta(X)$
Classification Accuracy

Testing


Real World Problem
Model the problem
Formal Model θ
Learning Algorithm
Training Data
Test on the testing dataset!!!
Testing Data
$g_\theta(X)$
Classification Accuracy

Training a Learning Machine

- We consider statistical learning paradigm here
  - We are given set of *N* "training" *instances*
    - Each training instance is pair: ($<x_1, x_2, \ldots, x_m>$, y)
    - Training instances are *previously* observed data
    - Gives the output value *y* associated with each observed vector of input values $<x_1, x_2, \ldots, x_m>$
  - Learning: use training data to specify $g(\boldsymbol{X})$
    - Generally, first select a parametric form for $g(\boldsymbol{X})$
    - Then, estimate parameters of model $g(\boldsymbol{X})$ using training data
    - For classification, generally best choice of

$$\hat{Y} = g(\boldsymbol{X}) = \arg\max_{y} \hat{P}(Y \mid X) = \arg\max_{y} \hat{P}(X, Y)$$

Naïve Bayes Classifier

- Say, we have *m* input values **X** = <X_1, X_2, …, X_m>
  - Assume variables X_1, X_2, …, X_m are **<u>conditionally independent</u>** given Y
    - Really don't believe X_1, X_2, …, X_m are conditionally independent
    - Just an approximation we make to be able to make predictions
    - This is called the "Naive Bayes" assumption, hence the name
  - Predict Y using $\hat{Y} = \arg\max_y P(\boldsymbol{X}, Y) = \arg\max_y P(\boldsymbol{X} \mid Y) P(Y)$
    - But, we now have:

$$P(\boldsymbol{X} \mid Y) = P(X_1, X_2, \ldots X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$$ by conditional independence

  - Note: computation of PMF table is <u>linear</u> in *m* : O(*m*)
    - Don't need much data to get good probability estimates

Computing Probabilities from Data

- Various probabilities you will need to compute for Naive Bayesian Classifier (using MLE here):

$$\hat{P}(Y=0) = \frac{\#\text{instances in class} = 0}{\text{total } \#\text{ instances}}$$

$$\hat{P}(X_i = 0, Y = 0) = \frac{\#\text{instances where } X_i = 0 \text{ and class} = 0}{\text{total } \#\text{ instances}}$$

$$\hat{P}(X_i = 0 \mid Y = 0) = \frac{\hat{P}(X_i = 0, Y = 0)}{\hat{P}(Y = 0)} \qquad \hat{P}(X_i = 0 \mid Y = 1) = \frac{\hat{P}(X_i = 0, Y = 1)}{\hat{P}(Y = 1)}$$

$$\hat{P}(X_i = 1 \mid Y = 0) = 1 - \hat{P}(X_i = 0 \mid Y = 0)$$

$$\hat{y} = \arg\max_y P(\boldsymbol{X} \mid Y)P(Y) = \arg\max_y (\log[P(\boldsymbol{X} \mid Y)P(Y)])$$

$$\log P(\boldsymbol{X} \mid Y) = \log P(X_1, X_2, \ldots X_m \mid Y) = \log \prod_{i=1}^{m} P(X_i \mid Y) = \sum_{i=1}^{m} \log P(X_i \mid Y)$$

On biased datasets

Ancestry dataset prediction

East Asian
or
Ad Mixed American (Native + Early Immigrants)

Is the ancestry dataset biased?

Yes!

It is much easier
to write a binary classifier
when learning ML
for the first time

Learn Two Things From This

1. What classification with DNA Single Nucleotide Polymorphisms looks like.

2. That genetic ancestry paints a more realistic picture of how we are mixed in many nuanced ways.

3. The importance of choosing the right data to learn from. Your results will be as biased as your dataset.

Know it so you can beat it!

Ethics in Machine Learning
is a whole new field

End Review

Notation For Today

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid function

$$\theta^T \mathbf{x} = \sum_{i=1}^{n} \theta_i x_i$$

$$= \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Weighted sum
(aka dot product)

$$\sigma(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Sigmoid function of
weighted sum

Step 1: Big Picture

From Naïve Bayes to Logistic Regression

- Recall the Naive Bayes Classifier
  - Predict $\hat{Y} = \arg\max_y P(\boldsymbol{X}, Y) = \arg\max_y P(\boldsymbol{X} \mid Y) P(Y)$
  - Use assumption that $P(\boldsymbol{X} \mid Y) = P(X_1, X_2, \ldots X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$
  - We are really modeling joint probability P(**X**, Y)
- But for classification, really care about P(Y | **X**)
  - Really want to predict $\hat{y} = \arg\max_y P(Y \mid \boldsymbol{X})$
  - Modeling full joint probability P(**X**, Y) is equivalent
- Could we model P(Y | **X**) directly?
  - Welcome our friend: logistic regression!

Logistic Regression Assumption

- Model *conditional* likelihood P(Y | **X**) directly
  - Model this probability with *logistic* function:

$$P(Y = 1|\mathbf{X}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$$

  - For simplicity define $x_0 = 1$ so $z = \theta^T \mathbf{x}$
  - Since $\mathrm{P}(Y = 0 \mid \mathbf{X}) + \mathrm{P}(Y = 1 \mid \mathbf{X}) = 1$:

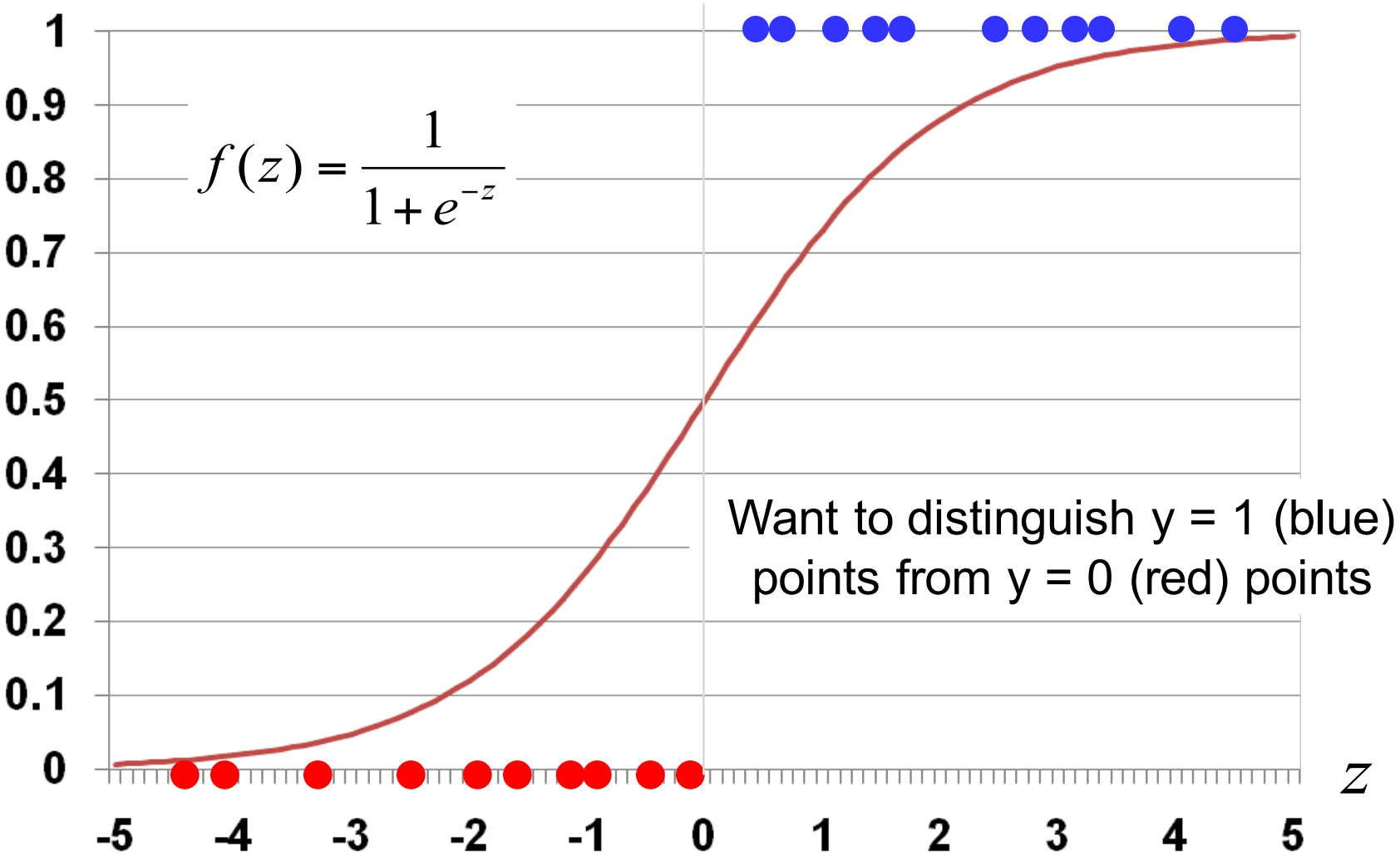
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Recall:
Sigmoid function

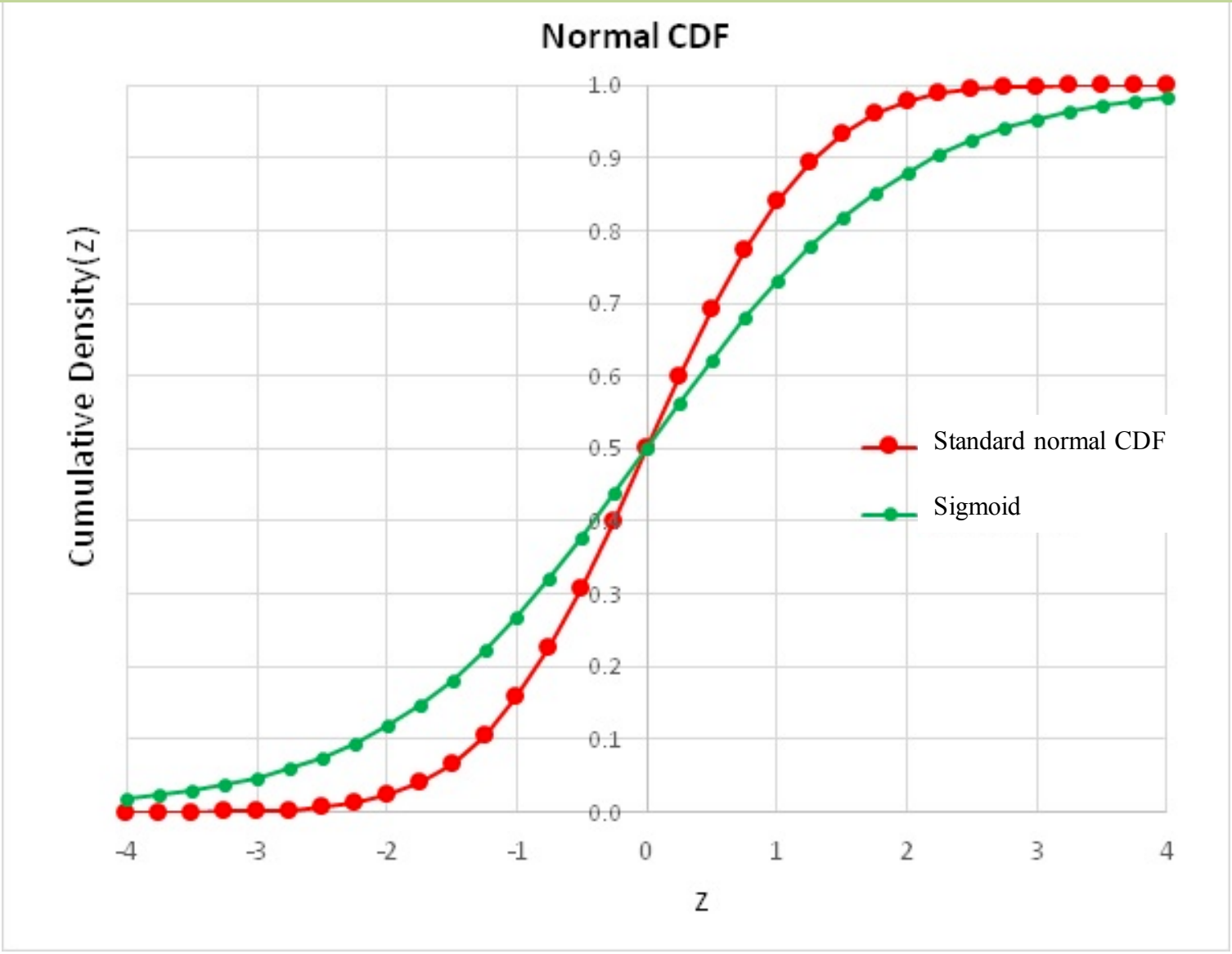
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The Sigmoid Function



Note: inflection point at z = 0. $f(0) = 0.5$

The Sigmoid Function



What is in a Name

Regression Algorithms

Linear Regression 

Classification Algorithms

Naïve Bayes 

Logistic Regression 

Awesome classifier, terrible name

If Chris could rename it he would call it: Sigmoidal Classification

Training Data

Assume IID data:

N training datapoints

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots (\mathbf{x}^{(n)}, y^{(n)})$$

$$m = |\mathbf{x}^{(i)}|$$

Each datapoint has m features and a single y

Logistic Regression

1 Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

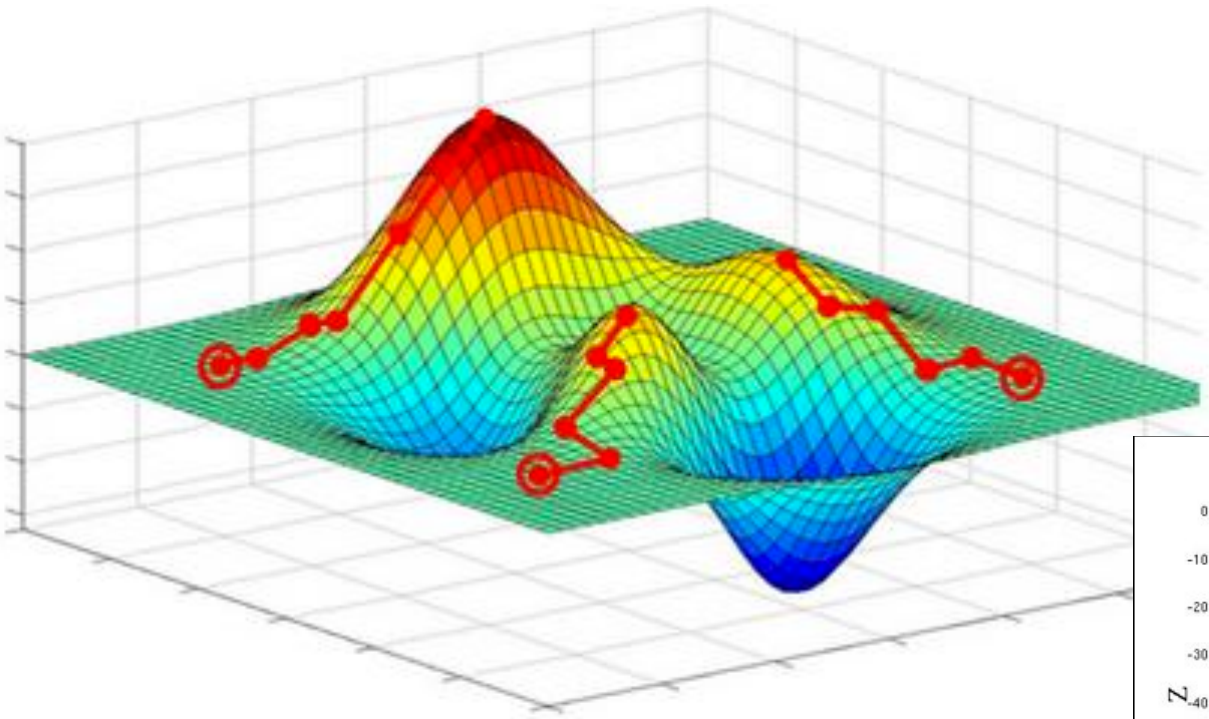
2 Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

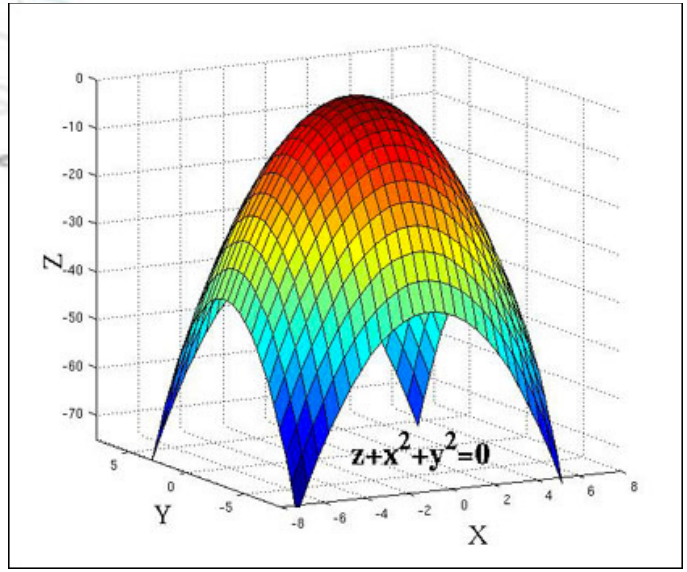
3 Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Gradient Ascent



Logistic regression
LL function is convex



Walk uphill and you will find a local maxima
(if your step size is small enough)

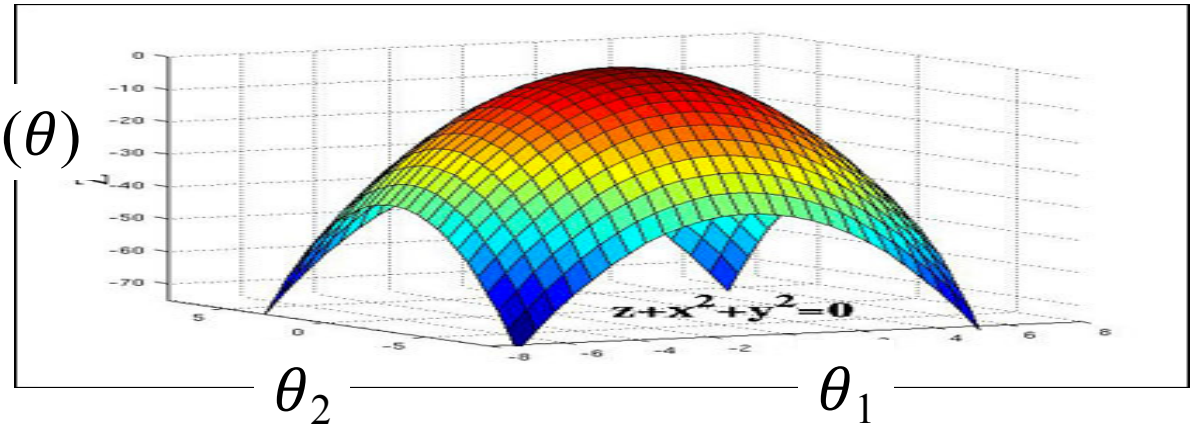
Gradient Ascent Step

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Do this for all thetas!



Logistic Regression Training

Initialize: $\beta_j = 0$ for all $0 \le j \le m$

// "epochs" = number of passes over data during learning

for (i = 0; i < epochs; i++) {

  Initialize: gradient[j] = 0 for all $0 \le j \le m$

  // Compute "batch" gradient vector

  for each training instance ($\langle x_1, x_2, \ldots, x_m \rangle$, y) in data {

    // Add contribution to gradient for each data point

    for (j = 0; j <= m; j++) {

      // Note: x_j below is j-th input variable and $x_0 = 1$.

      gradient[j] += $x_j (y - \frac{1}{1+e^{-z}})$ where $z = \sum_{j=0}^{m} \beta_j x_j$

    }

  }

  // Update all θ_j. Note *learning rate* η is pre-set constant

  θ_j += η * gradient[j] for all $0 \le j \le m$

}

Classification with Logistic Regression

- Training: determine parameters θ_j (for all $0 \le j \le m$)
  - After parameters θ_j have been learned, test classifier
- To test classifier, for each new (test) instance ***X***:
  - Compute: $p = P(Y = 1 \mid \boldsymbol{X}) = \dfrac{1}{1 + e^{-z}}$, where $z = \theta^T \mathbf{x}$
  - Classify instance as: $\hat{y} = \begin{cases} 1 & p > 0.5 \\ 0 & \text{otherwise} \end{cases}$
  - Note about evaluation set-up: parameters θ_j are **not** updated during "testing" phase

Step 2: How Come?

Logistic Regression

1 Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

2 Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3 Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

How did we get that LL function?

Log Probability of Data

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Implies

$$P(Y = y|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})^y \cdot \left[1 - \sigma(\theta^T \mathbf{x})\right]^{(1-y)}$$

For IID data

$$L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = \mathbf{x}^{(i)})$$
$$= \prod_{i=1}^{n} \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\theta^T \mathbf{x}^{(i)})\right]^{(1-y^{(i)})}$$

Take the log

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

How did we get that gradient?

Sigmoid has a Beautiful Slope

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x)?$$

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1 - z]$$

True fact about sigmoid functions

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x) = \frac{\partial}{\partial z}\sigma(z) \cdot \frac{\partial z}{\partial \theta_j}$$

Chain rule!

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x) = \sigma(\theta^T x)[1 - \sigma(\theta^T x)]x_j$$

Plug and chug

Sigmoid, you should be a ski hill

Gradient Update

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x}]$$

Imagine only one data point

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

$$= \left[\frac{y - \sigma(\theta^T x)}{\sigma(\theta^T x)[1 - \sigma(\theta^T x)]} \right] \sigma(\theta^T x)[1 - \sigma(\theta^T x)] x_j$$

$$= \left[y - \sigma(\theta^T x) \right] x_j$$

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

For many data points

Logistic Regression

1 Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

2 Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

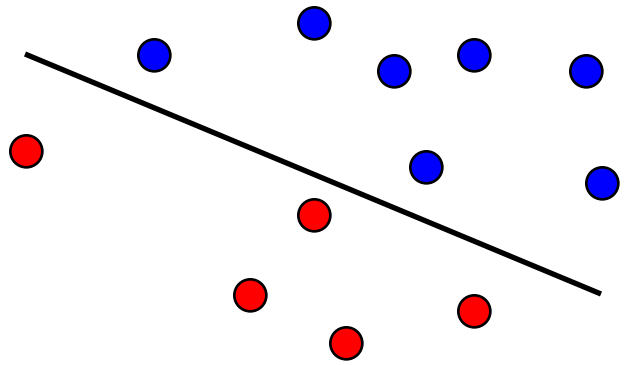
3 Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Step 3: Philosophy

Discrimination Intuition

- Logistic regression is trying to fit a **line** that separates data instances where $y = 1$ from those where $y = 0$



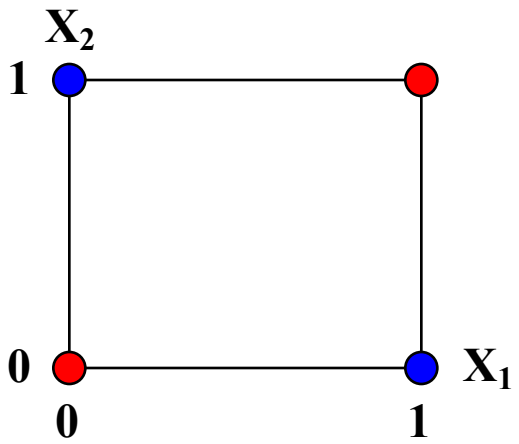
$$\theta^T \mathbf{x} = 0$$

$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- We call such data (or the functions generating the data) "**linearly separable**"
- Naïve bayes is linear too as there is no interaction between different features.

Some Data Not Linearly Seperable

- Some data sets/functions are not separable
  - Consider function: $y = x_1$ XOR x_2
  - Note: y =1 iff **one** of either x_1 or x_2 = 1



  - Not possible to draw a line that successfully separates all the y = 1 points (blue) from the y = 0 points (red)
  - Despite this fact, logistic regression and Naive Bayes still often work well in practice

Logistic Regression vs Naïve Bayes

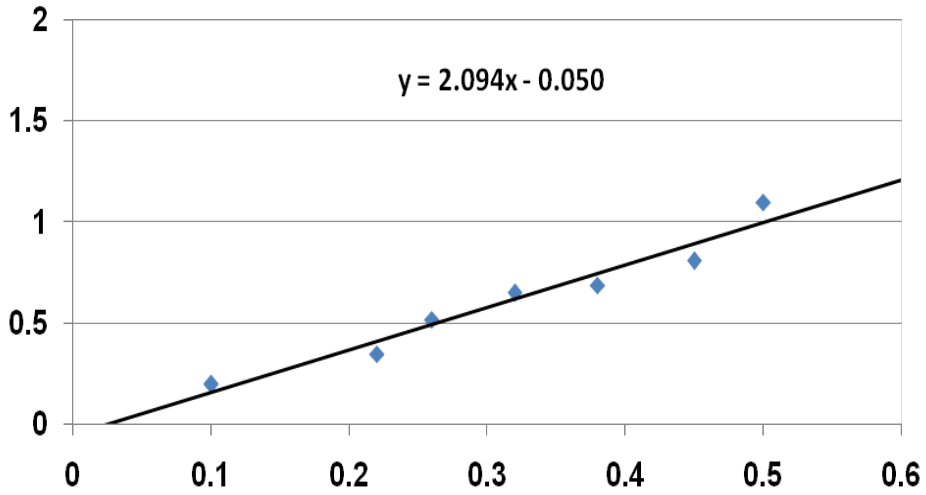
- Compare Naive Bayes and Logistic Regression
  - Recall that Naive Bayes models P(**X**, Y) = P(**X** | Y) P(Y)
  - Logistic Regression directly models P(Y | **X**)
  - We call Naive Bayes a "generative model"
    - Tries to model joint distribution of how data is "generated"
    - I.e., could use P(X, Y) to generate new data points if we wanted
    - But lots of effort to model something that may not be needed
  - We call Logistic Regression a "discriminative model"
    - Just tries to model way to discriminate y = 0 vs. y = 1 cases
    - Cannot use model to generate new data points (no P(X, Y))
    - Note: Logistic Regression can be generalized to more than two output values for y (have multiple sets of parameters β_j)

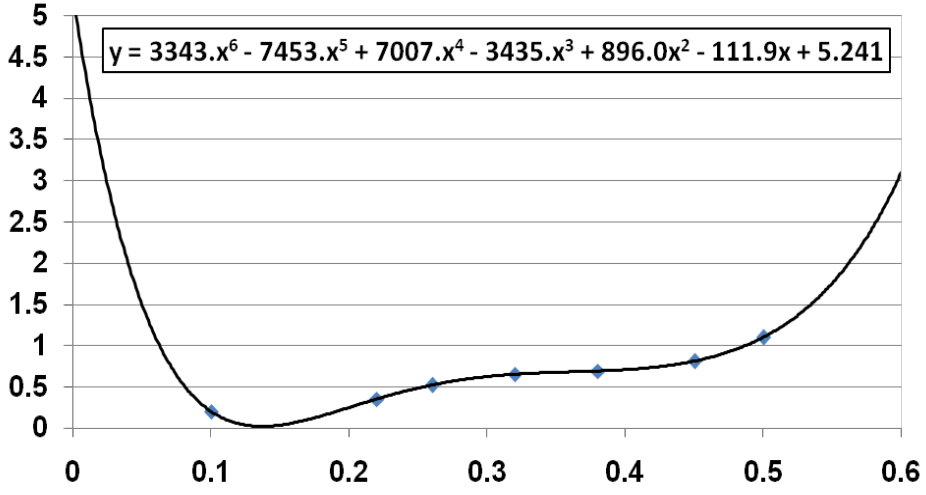
Choosing an Algorithm?

- Many trade-offs in choosing learning algorithm
  - Continuous input variables
    - Logistic Regression easily deals with continuous inputs
    - Naive Bayes needs to use some parametric form for continuous inputs (e.g., Gaussian) or "discretize" continuous values into ranges (e.g., temperature in range: <50, 50-60, 60-70, >70)
  - Discrete input variables
    - Naive Bayes naturally handles multi-valued discrete data by using multinomial distribution for $P(X_i \mid Y)$
    - Logistic Regression requires some sort of representation of multi-valued discrete data (e.g., multiple binary features)
    - Say $X_i \in \{A, B, C\}$. Not necessarily a good idea to encode X_i as taking on input values 1, 2, or 3 corresponding to A, B, or C.

Good ML = Generalization

- Goal of machine learning: build models that ***generalize*** well to predicting new data
  - "Overfitting": fitting the training data too well, so we lose generality of model
    - Example: linear regression vs. Newton's interpolating polynomial





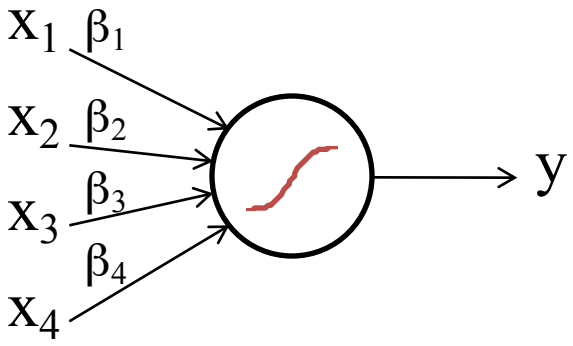
    - Interpolating polynomial fits training data perfectly!
    - Which would you rather use to predict a new data point?

To Consider with Logistic Regression

- Logistic Regression can more easily overfit training data than Naive Bayes
  - Logistic Regression is not modeling whole distribution, it is just optimizing prediction of Y
  - Overfitting can is problematic if distributions of training data and testing data differ a bit
  - There are methods to mitigate overfitting in Logistic Regression
    - Called “regularizers”
    - Use Bayesian priors on parameters$_j$ rather than just maximizing conditional likelihood (like MAP)
    - Many others!

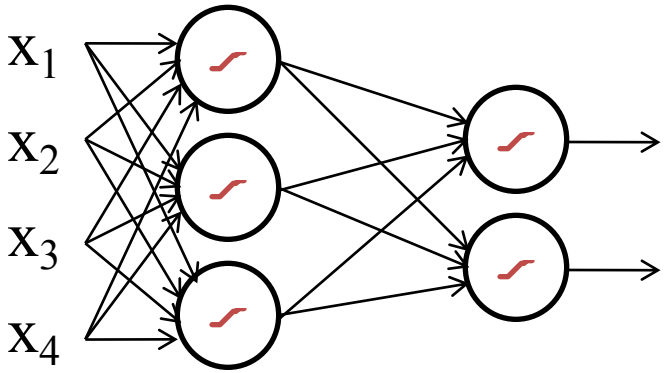
Logistic Regression and Neural Networks

- Consider logistic regression as:



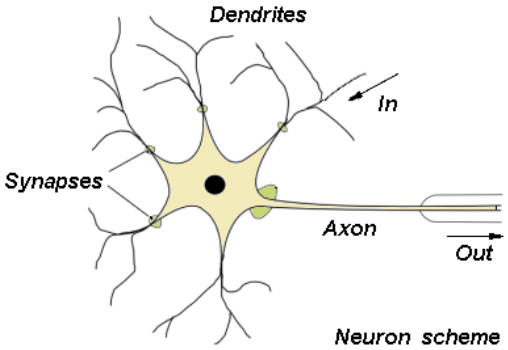
Logistic regression is same as a one node neural network

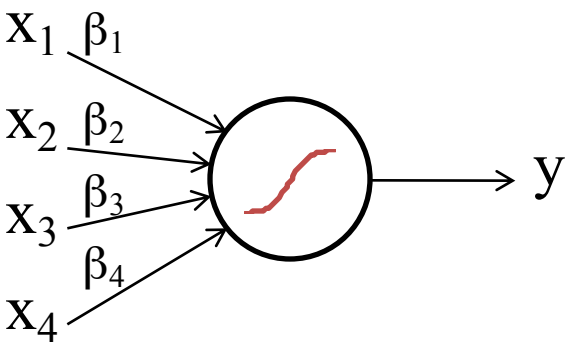
- Neural network



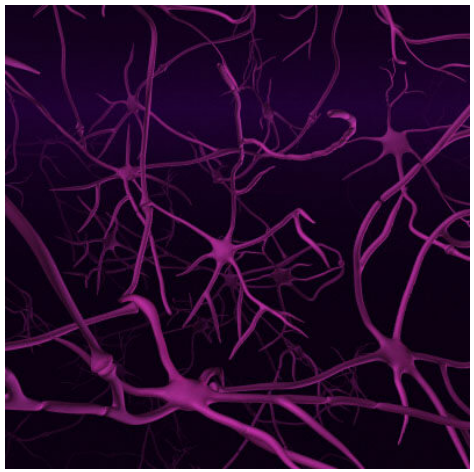
Biological Basis for Neural Networks

- A neuron

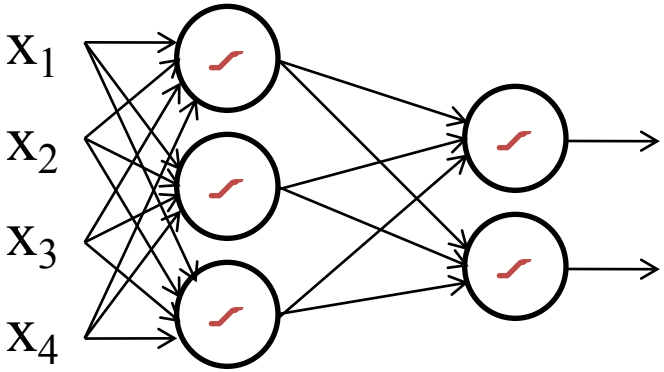




- Your brain



Actually, it's probably someone else's brain



Next up: Neural Networks!