UNION COLLEGE

CSC321 Data Mining & Machine Learning

Prof. Nick Webb
webbn@union.edu

Data

- A quick review of data and terminology
- We have a data set
- Each data set has a number of instances
- Each instance includes input, and output

Contact lens data

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Would translate to…

[[‘young’, ‘myope’, ‘no’, ‘reduced’, ‘none’],
 [‘young’, ‘myope’, ‘no’, ‘normal’, ‘soft’],
 [‘young’, ‘myope’, ‘yes’, ‘reduced’, ‘none’],
…
 [‘presbyopic’, ‘hypermetrope’, ‘yes’, ‘normal’, ‘none’]]

Predicting CPU performance

- Example: 209 different computer configurations

	Cycle time (ns)	Main memory (Kb)		Cache (Kb)	Channels		Performance
	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
…							
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

- Linear regression function

```
PRP = -55.9 + 0.0489 MYCT + 0.0153 MMIN + 0.0056 MMAX
      + 0.6410 CACH - 0.2700 CHMIN + 1.480 CHMAX
```

What about classification?

- We can use (almost) any regression technique to do classification
- Let's start with the most simple version:
  - Two-class problems
  - Learn a line that separates two classes
    - Called a decision boundary

Classifying iris flowers

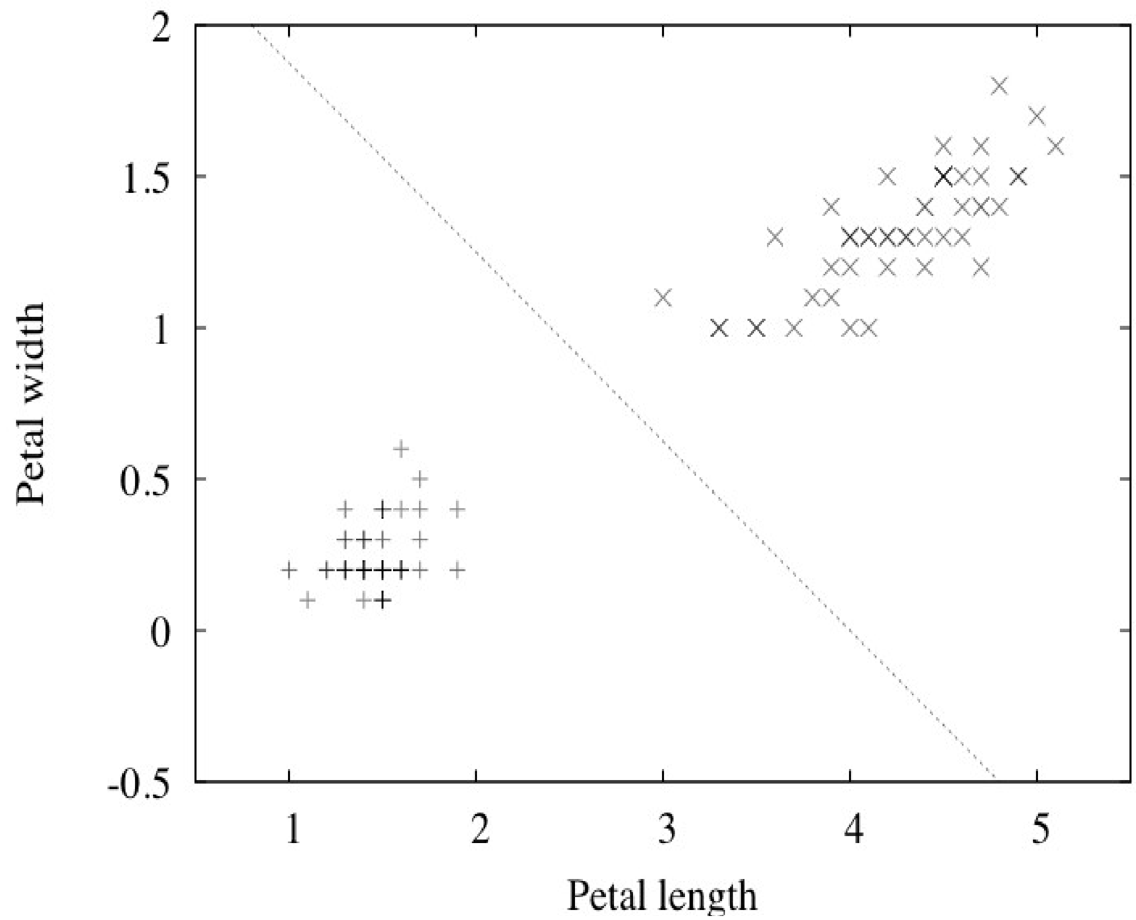
	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
…					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
…					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
…					

```
If petal length < 2.45 then Iris setosa
If sepal width < 2.10 then Iris versicolor
...
```

Two-class classification

	Two Class Classification	
$y \in \{0, 1\}$	**1 or Positive Class**	**0 or Negative Class**
Email	Spam	Not Spam
Tumor	Malignant	Benign
Transaction	Fraudulent	Not Fraudulent

Separating setosas from from versicolors



`2.0 – 0.5PETAL-LENGTH – 0.8PETAL-WIDTH = 0`

Logistic Regression

- Very similar to linear regression
- Input values (x) are combined with weights (coefficients) to predict output value (y)
- Instead of y being a real number, it should be a binary value

$$y = \frac{1.0}{1.0 + e^{-(b0 + b1 \times x1)}}$$

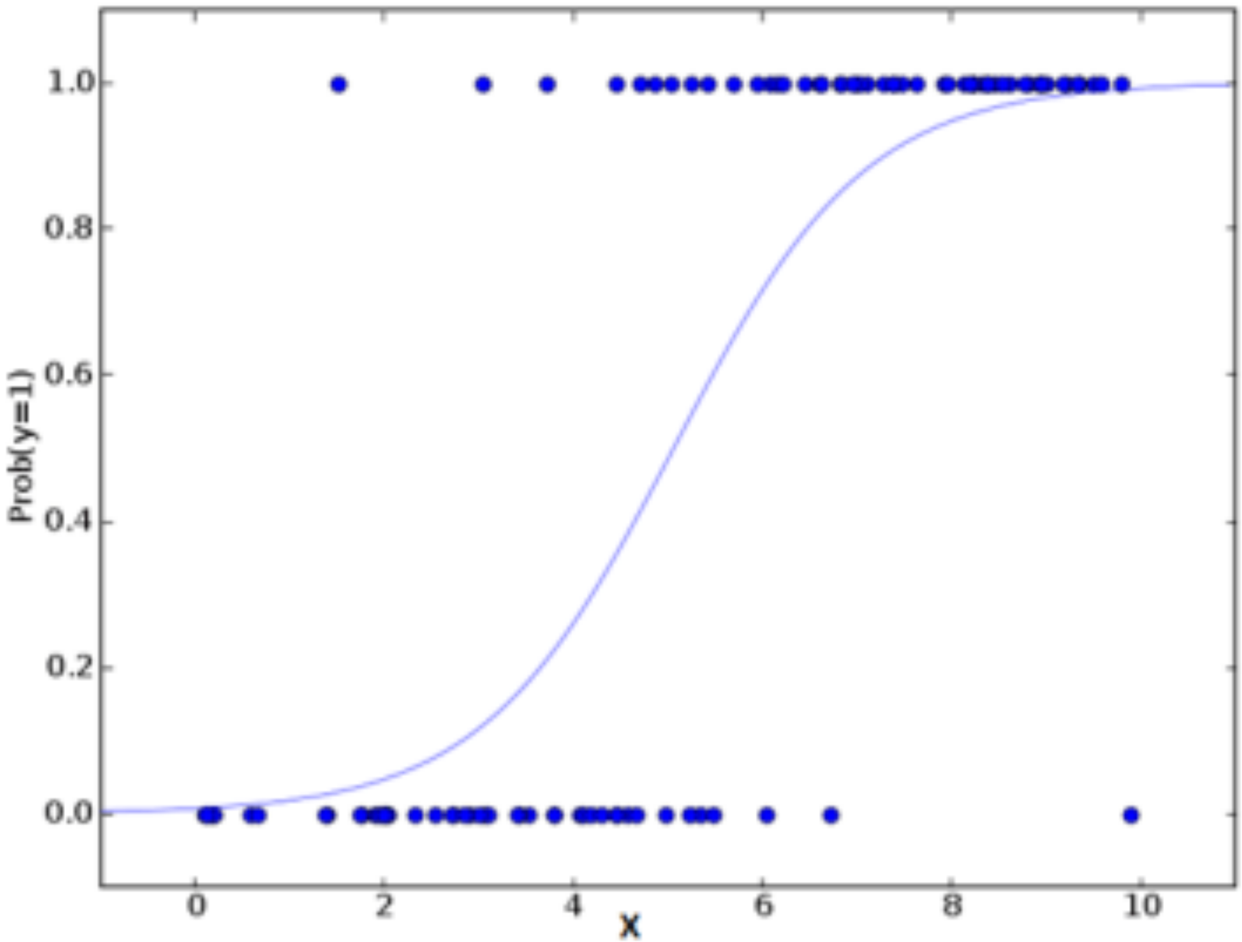
UNION COLLEGE

Logistic Regression

$$y = \frac{1.0}{1.0 + e^{-(b0 + b1 \times x1)}}$$

- Where
  - e is the base of the natural log
    - Euler’s number
  - y is the predicted value
  - b0 is the bias or intercept
  - b1 is the coefficient for the input value x1

Logit Function



Logistic Regression

- y will be a real value predicted in the range (0,1)
- We will need to round it to an integer to determine which class it predicts
- We will also need to learn those coefficients, b0 and b1.
- We'll use stochastic gradient descent

Stochastic Gradient Descent

- Same algorithm as before
  - Combining learning rate and epochs
  - Minimizing our (error) function

b1(t+1) = b1(t) + learning rate * error * x1

Stochastic Gradient Descent

- We also need to estimate the coefficient b0

b0(t+1) = b0(t) + learning rate * error

Logistic Regression

- Once we've learned our coefficients
- We make a prediction, using

$$y = \frac{1.0}{1.0 + e^{-(b0 + b1 \times x1)}}$$

- And then we round y, to either 1 or 0

Measuring Performance

- So we have a method of performing simple logistic regression
- How well does it do?
- We need a way of measuring performance
- AND we need something to compare it to
  - Another simple machine learning algorithm

UNION COLLEGE

ZeroR

- For regression, use the mean of the output variable

- For classification, use the most frequently occurring class

Measuring Performance

- Calculate the accuracy

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \text{ x } 100$$

- When we get above 2 classes there are other things we might want to look at to help us understand performance

Confusion Matrix

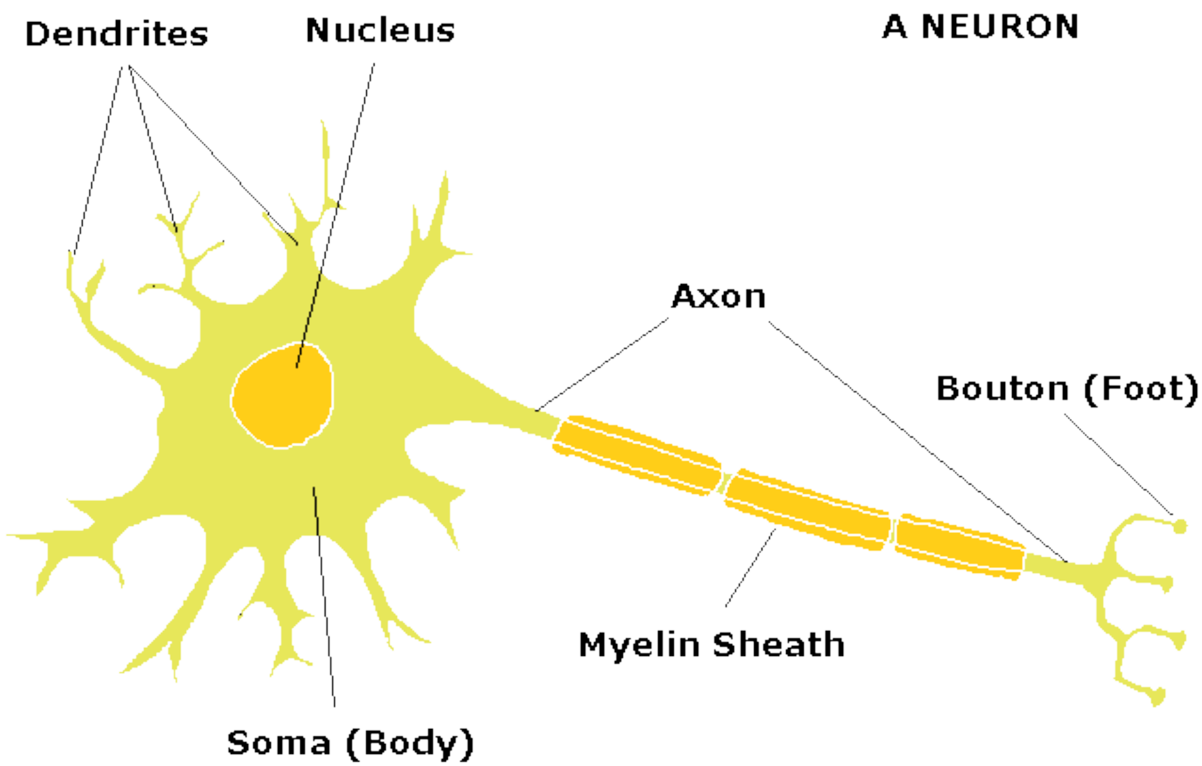
n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Neural Models

- So I said we wouldn't do deep learning
  - And we wont
- But I will introduce neural networks
- Starting with the most simple model
  - A single neuron

UNION COLLEGE

A Neuron



How it works

- We have inputs (dendrites)…
- ...with weights
- We have an output…
- ...determined by an activation...
- ...which is transformed into an output value by a transfer function

Neural Model

- Activation

$$\text{activation} = \text{bias} + \sum_{i=1}^{n} weight_i \times x_i$$

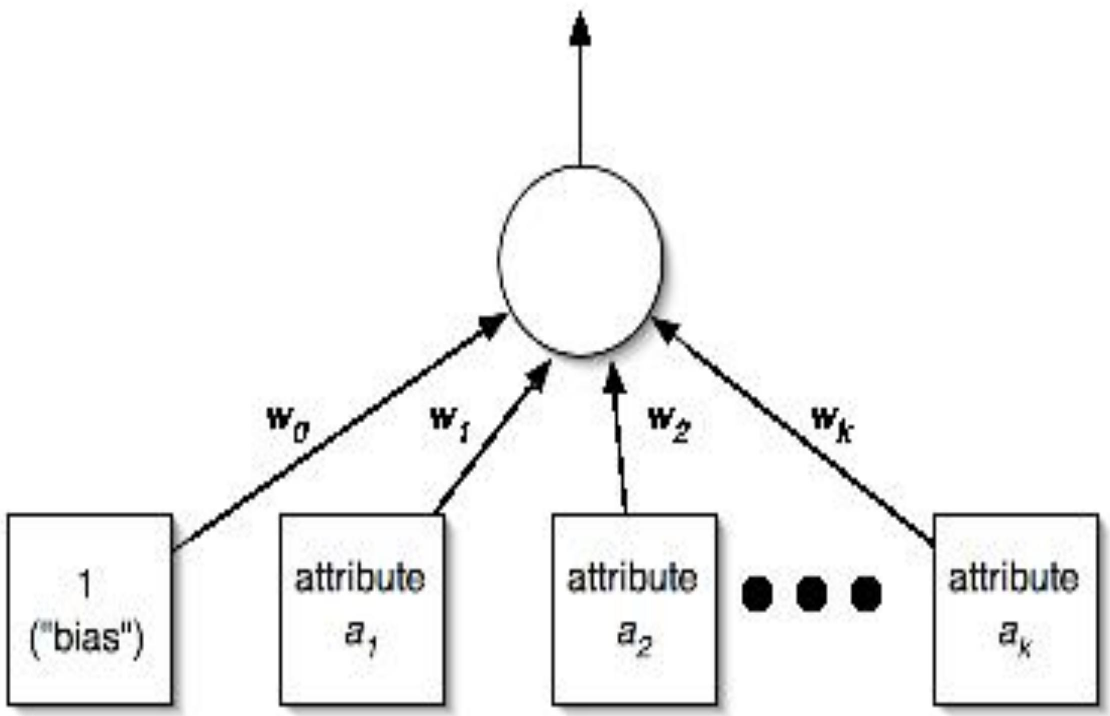
- Prediction

$$prediction = 1.0 \text{ IF activation} >= 0.0 \text{ ELSE } 0.0$$

Perceptron as a neural network

Output layer

Input layer



UNION COLLEGE

A new idea!

- Wrong
- Mathematically:
  - McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5:115–133.

- But we do need to find those weights on the inputs

So finding weights…

- If only we knew a method to do that…
- We can use stochastic gradient descent, where:

weight = weight + learning rate * error * x

UNION COLLEGE

Neural Model

- In this way, this perceptron is VERY closely related to linear regression
- All of these linear models are pretty straightforward
- …and powerful
- I don't fully expect you to grasp them UNTIL you implement them
- But I do want you to grasp certain principles

Principles

- We use training data to estimate some parameters, or learn some relationship between input variable (x) and output (y)
- These relationships can be found, without us knowing if they are meaningful
- We can only find what is in the data
- We can't say anything about what ISN'T in the data