Big Data in Reliability and Security: Some Basics

Saurabh Bagchi
School of Electrical and Computer Engineering
Department of Computer Science
Purdue University

WHIN

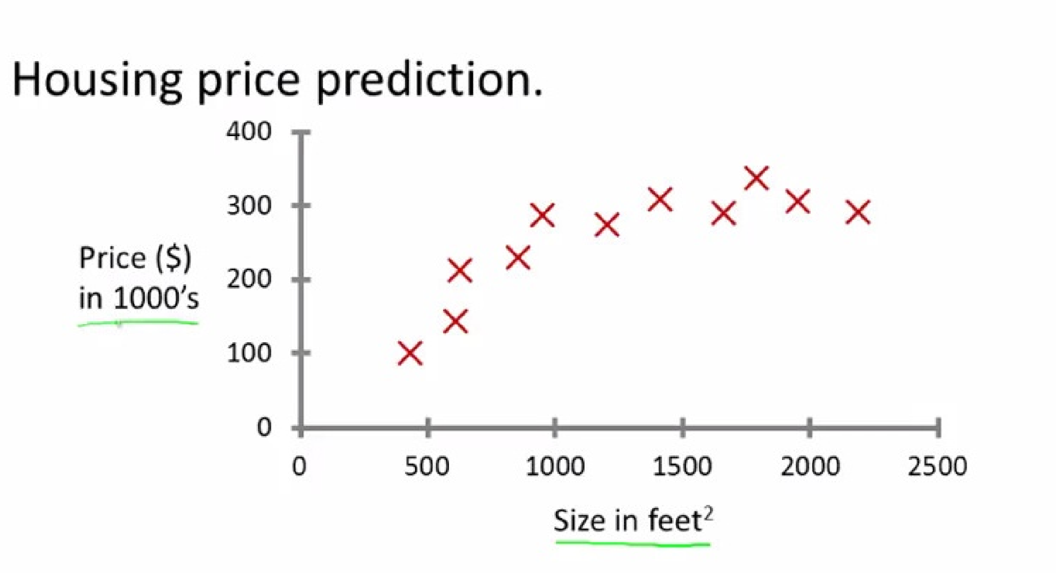
Material at: ***http://bit.ly/whin-big-data-algorithms***

Overview

- We will cover the basics and insight behind some big data algorithms that have been successfully applied to reliability and security
- The coverage will stress the intuition and applicability to real-world problems rather than theoretical purity
- New terms will be formatted as: *Here is an example new term*
- About Me:
  - WHIN Thrust Lead on Sensors and Systems
  - Professor in ECE, CS at Purdue University for 15 years
  - IEEE Computer Society Board of Governors
  - Research advisor to IBM, TCS
  - Startup co-founder in embedded system reliability (2012)
  - MS, PhD from University of Illinois at Urbana-Champaign (1998, 2001)

Hypothetical Problem

- We want to predict house prices (in a particular city)
- We are given the area of a house as an *input feature*
- We start off by collecting a dataset of houses that have sold in the last year



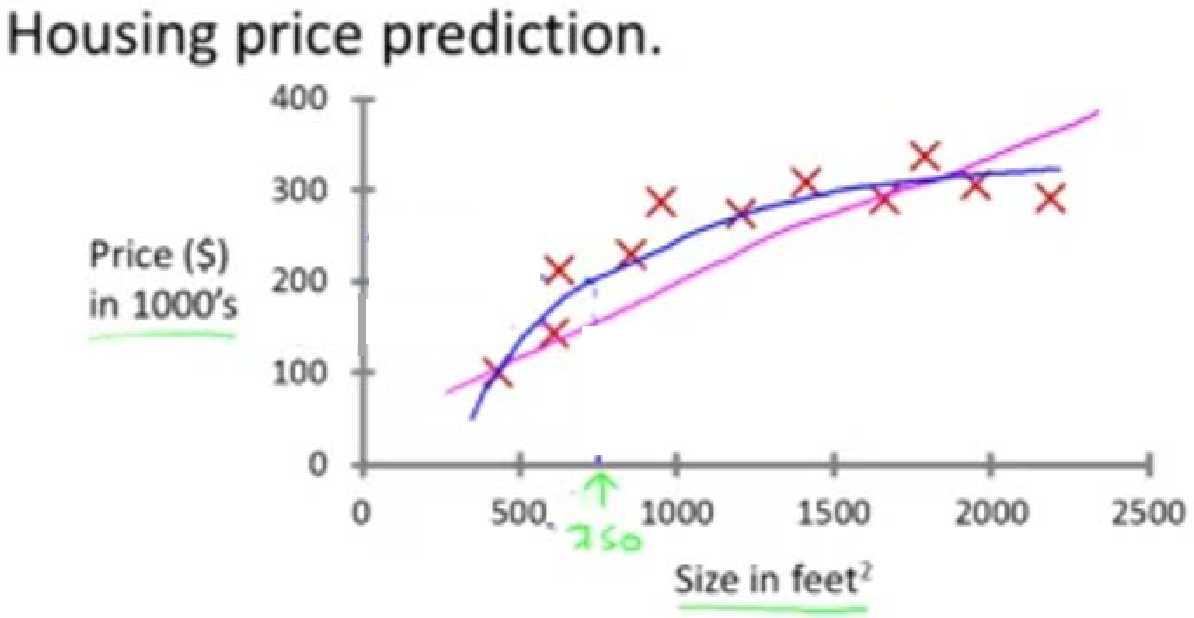
- My friend has a house of 750 sq. ft. and he wants me to predict what the house will sell for.

Housing Price Prediction

- Learning algorithm takes as input a set of <key, value> pairs like: [Living area (sq. ft.), Price (in 1000’s)]
- This is an example of a *Supervised Learning algorithm*
- The term Supervised Learning refers to the fact that we gave the algorithm a data set in which the "right answers" were given
  - Exact values of the keys (or features) and the values
  - Algorithm’s task is to create model out of the data set that is given
  - Algorithm’s task then is to use the model to predict for an unseen item what the value will be

Solution to the Housing Price Problem

- This specifically is a *regression problem*
  - We are trying to predict a continuous valued output
  - We can use various types of regression – linear, polynomial, logistic

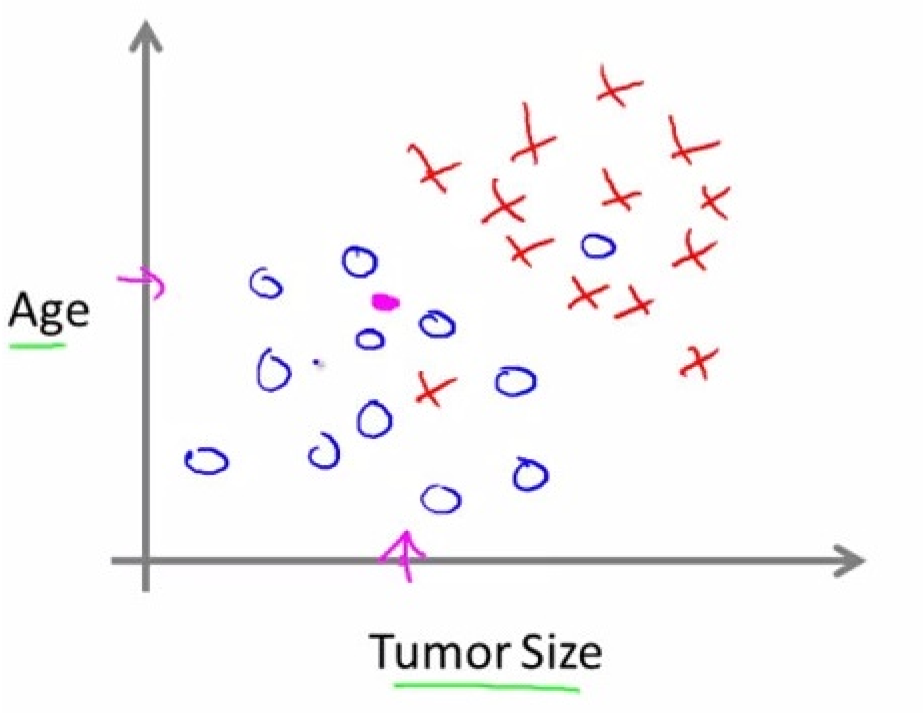


- So what should I tell my friend?
  - If we use linear regression, the price my friend's house sells for is $150K
  - If we use quadratic regression, the price my friend's house sells for is $200K

PURDUE UNIVERSITY

Another Supervised Learning Problem

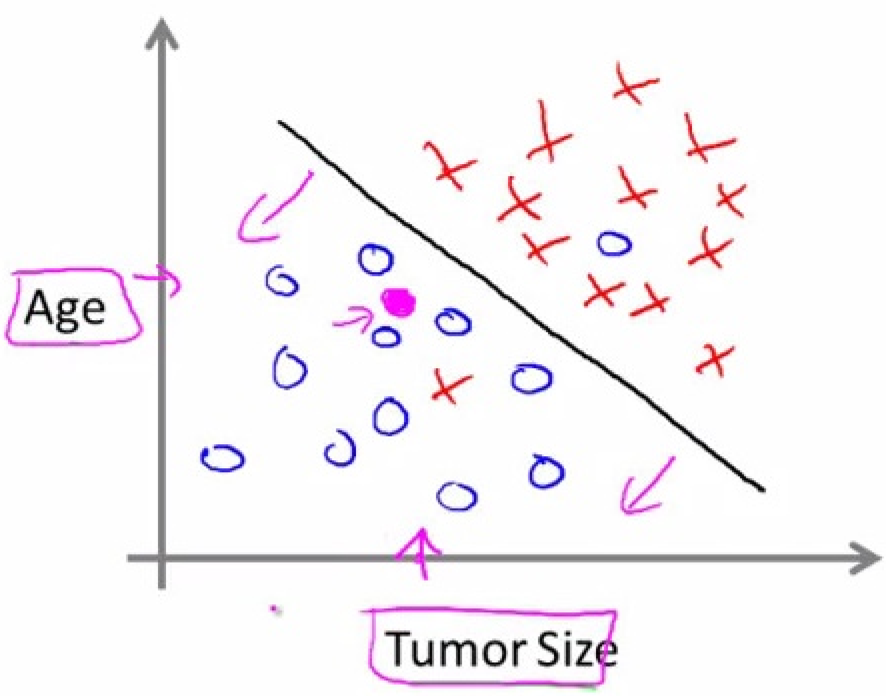
- A set of patients who have developed tumor and are coming in to an oncology clinic to determine if the tumor is benign or malignant
- Features: Age, Tumor Size
- Need to predict: Benign (0) or Malignant (1)



PURDUE
UNIVERSITY

Classification Problem

- This is an example of a *classification problem*
- The term classification refers to the fact, that here, we're trying to predict a discrete value output – zero or one, malignant or benign
- In classification problems, sometimes you can have more than two possible values for the output



- Typically problems have many more features
- Here, other features typically are: clump thickness, uniformity of cell size of tumor, etc.

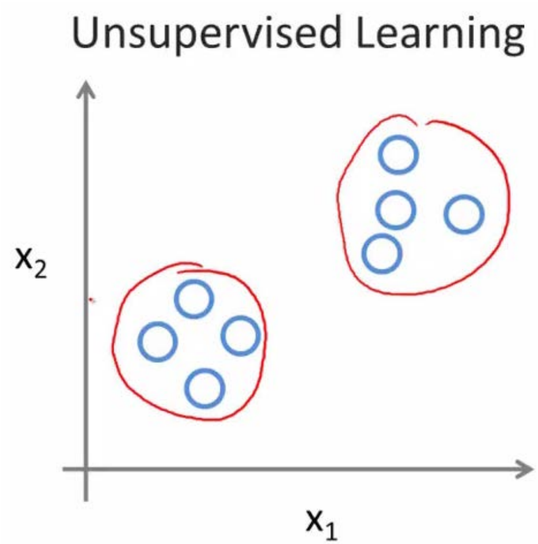
- Problem 1 - **Supervised learning**

Say if for the following problems I should use regression or classification.

1. I have a set of cricket bats for sale through auction. These are characterized by wood quality, weight, brand name. I want to predict how much a given bat will sell for.
2. I have a corpus of emails that have been received at our central IT organization. These are characterized by: domain of sender, length of email, whether it has images or not, whether it has attachments or not. I want to predict if the email is spam or not and then quarantine the spam messages.
3. I have for each student in my class her performance in the homeworks (a score from 0-100). Plus I have each student's attendance record. I want to predict what the score of the student will be in the final exam (on a range from 0-100).

Unsupervised Learning

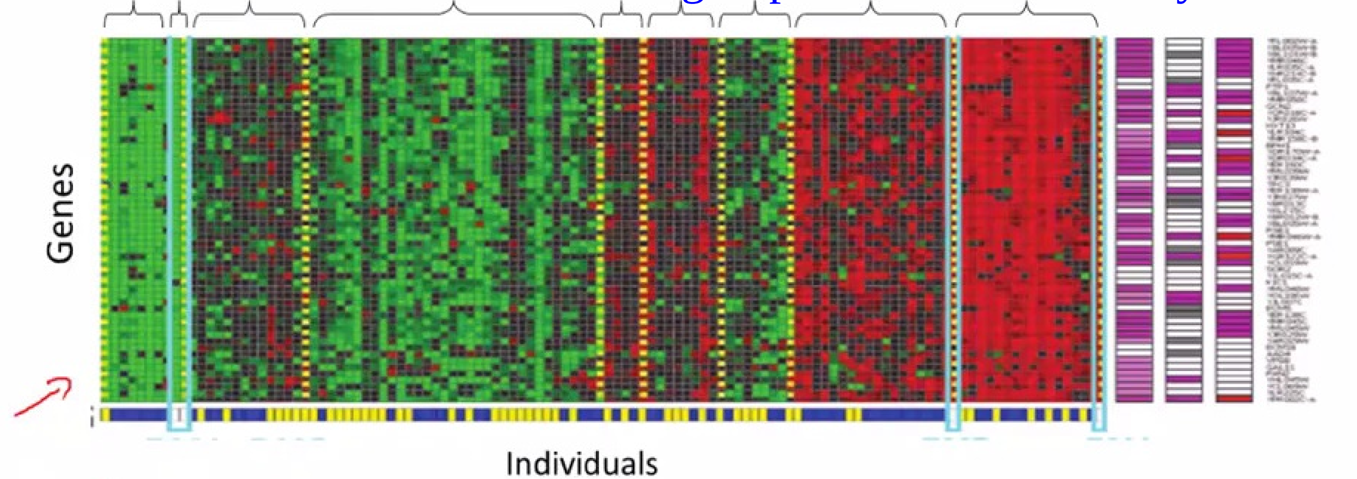
- Unsupervised learning allows us to approach problems with little to no ground truth
  - Contrast with supervised learning where each data point was “labeled” as a positive or a negative example (or one of multiple classes)
- We can derive structure from data where we don't necessarily know the effect of the variables
  - We can derive this structure by *clustering* the data based on relationships among the variables in the data



- Illustration: Two features
- Naturally forms two clusters
- Post analysis can assign meanings to the two clusters

Sample Problems: Canonical, Gene Expression

- Some canonical applications:
  - Cluster all incoming mails into 3 clusters – Important, Not important, Spam
  - Cluster all the news stories into a set of pre-defined clusters – World, India, Technology, Sports, etc.
  - Cluster users on a social network into groups that interact closely and often



[Source: Su-In Lee, Dana Pe'er, Aimee Dudley, George Church, Daphne Koller]

- Cluster people according to the expression profile of their genes
- Subsequently a clinician can come in and say which cluster has proclivity for which kind of disease

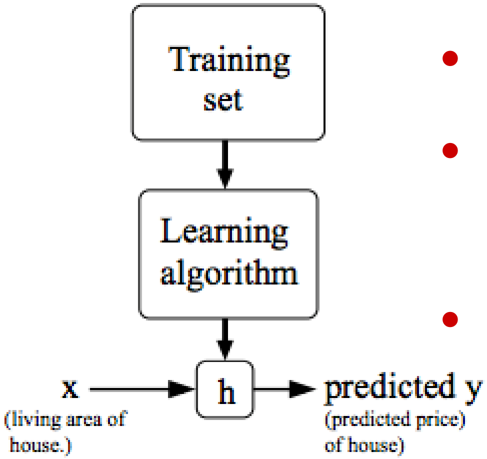
PURDUE UNIVERSITY

- Problem 2 – **Supervised vs unsupervised learning**

Say if for the following problems I should use supervised learning or unsupervised learning.

1. Given a set of emails, each with its originating domain and number of routing hops it has come through to reach my Inbox, I want to learn a filter that tells me if the sender is from IIT Kharagpur or outside.
2. Given a set of social network posts, I want to categorize them by the trending topics of the day.
3. Given a set of purchasing data for movie sound tracks (age, income, gender, and which genre movie the user bought), I want to create user segments to target marketing ads to.
4. Given a dataset of patients who are normal versus obese based on their lifestyle choices (number of hours of exercise, amount of calories consumed, categorization of parents and spouse), I want to classify a new patient with the likelihood for obesity.

Different Datasets: Training, Validation, Test

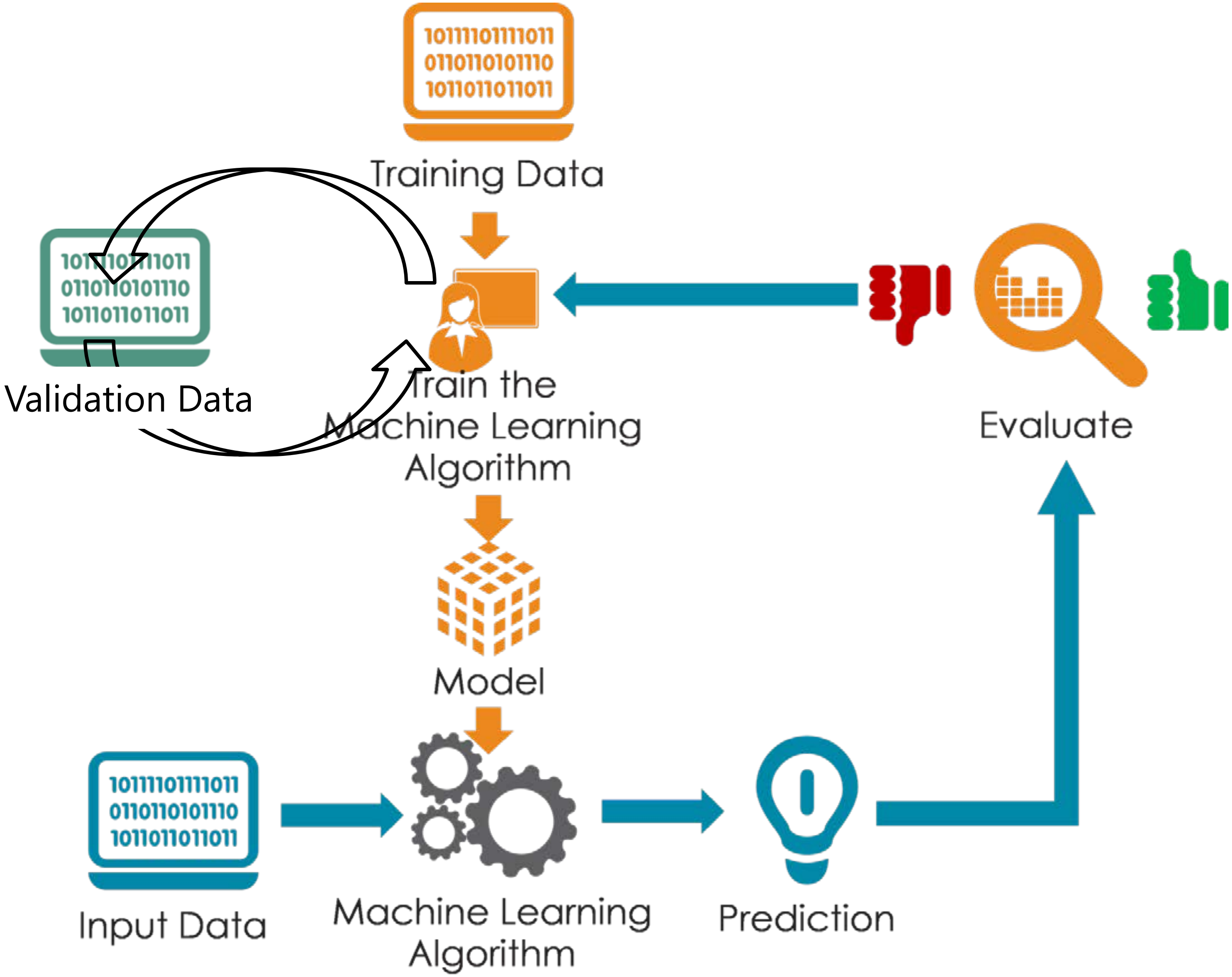


- Input features: $x^{(i)}$ (living area in this example)
- Target or output variable that we want to predict: $y^{(i)}$
- Training set: m pairs of $(x^{(i)}, y^{(i)})$

- Commonly used jargon in ML literature
- ***Train set***: Using to train and optimize the parameters of the model
- ***Validation set***: Choose hyperparameters and prevent overfitting
- ***Test set***: Data that will be given to our algorithm in production and which our algorithm will be evaluated on

PURDUE UNIVERSITY

Different Datasets: Training, Validation, Test



What's a Good Model?

Training Set

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

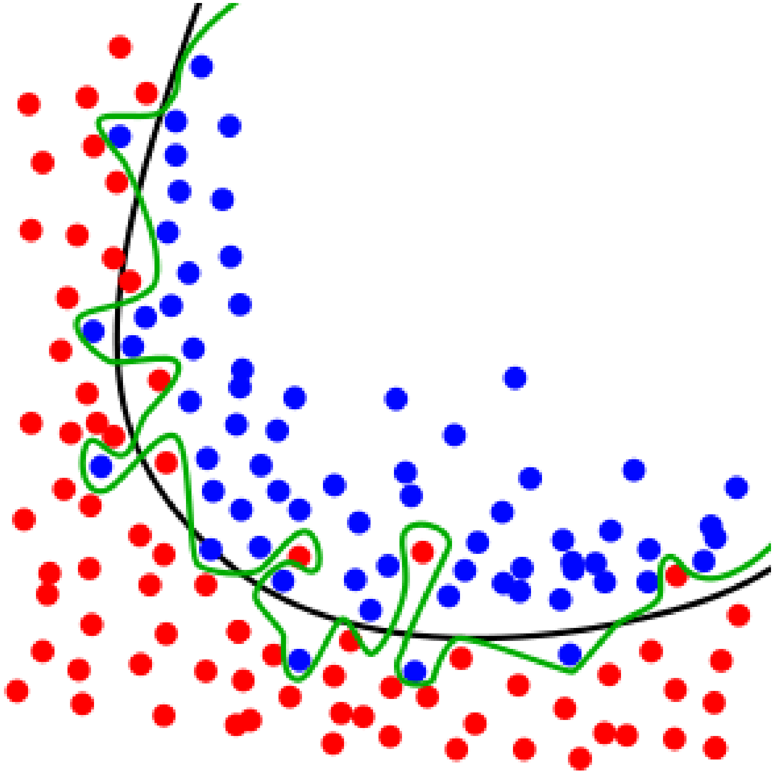
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

- Want to make predictions using above linear model
- Model fitting $\Rightarrow$ Figure out "correct" values of θ_0 and θ_1 $\Rightarrow$ $h(x)$ is close to the y values in the training set
- Cost function that we minimize

$$J(\theta_0, \theta_1) = \underset{\theta_0,\ \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)}\right)^2$$

PURDUE
UNIVERSITY

What's a Good Model?

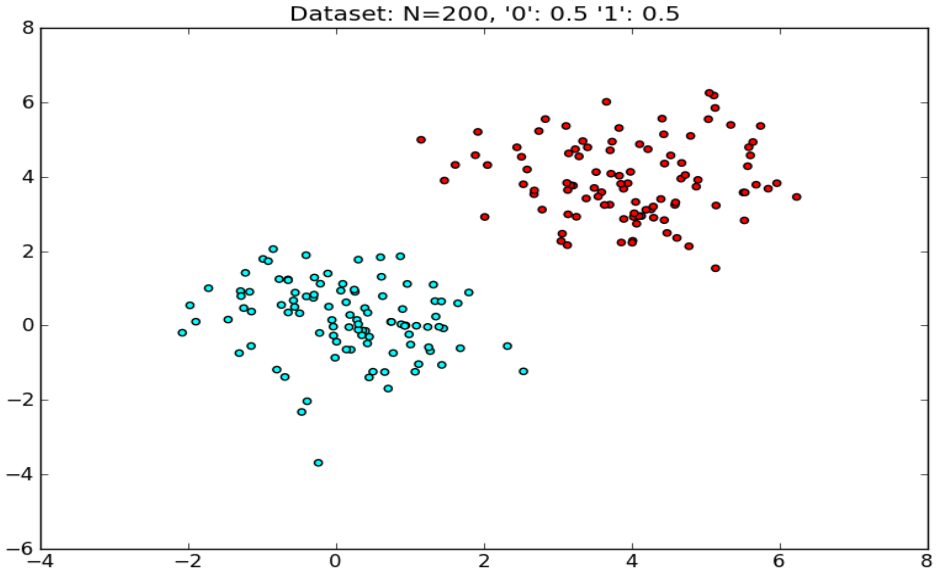


- We want to separate the two different colored points
- Which is the better model?
  - Black curve
  - Green curve
- Which has a lower cost function based only on the training data?

PURDUE
UNIVERSITY

Linear SVM and Binary Classification

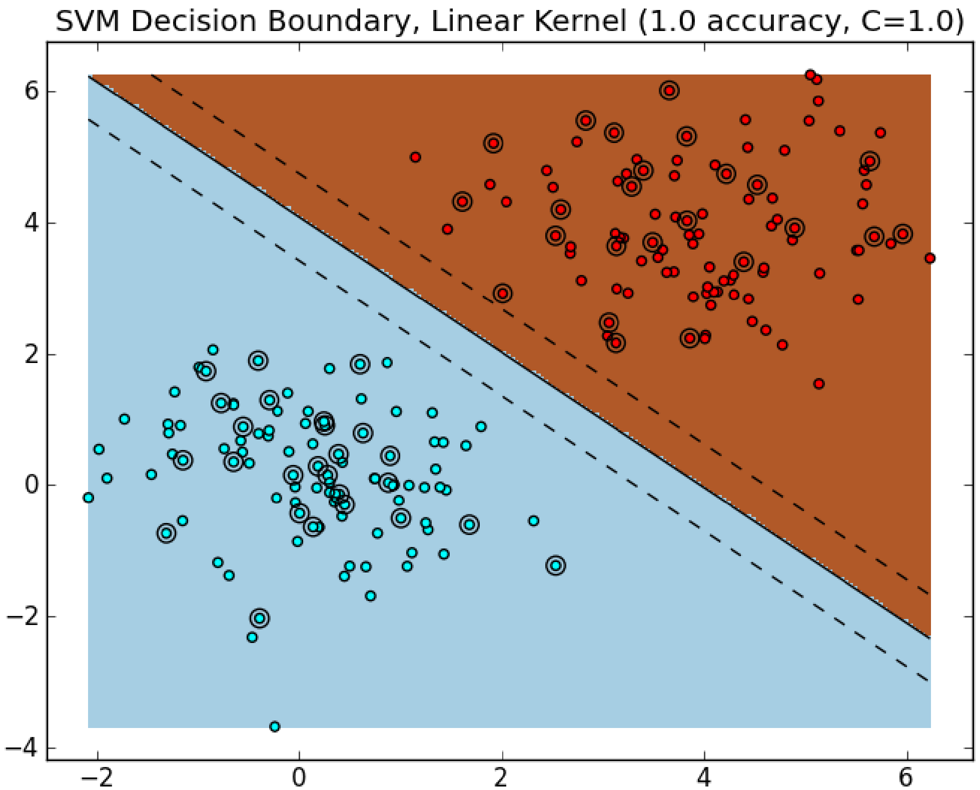
- Suppose that we have a two-class dataset *D*, and we wish to train a classifier to predict the class labels of future data points
  - This is known as the "binary classification" problem
- Examples:
  - Medicine: Given a patient's vital data, does the patient have a cold?
  - Computer Vision: Does this image contain a person?
- A popular and yet simple classifier is the Support Vector Machine (SVM)



- The data is easily linearly separable $\Rightarrow$ SVM is able to find a margin that perfectly separates the training data
- This also generalizes very well to the test set
- The hyperplane $\vec{w}$ (a line in R^2) separates the space into two halves

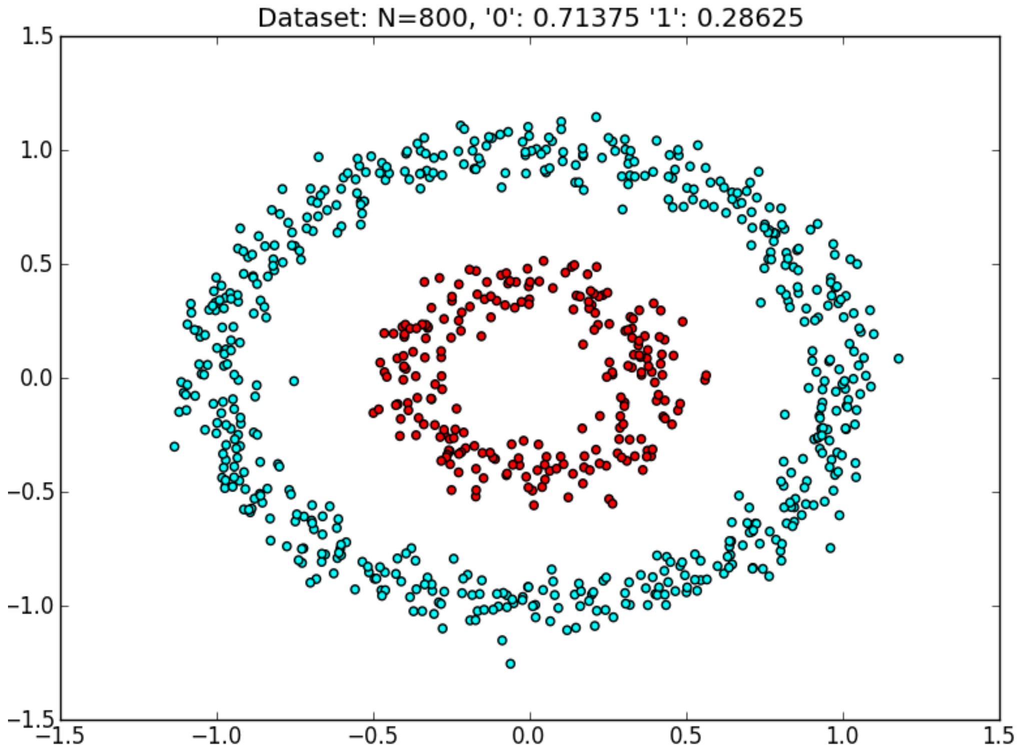
PURDUE
UNIVERSITY

Linear SVM Decision Boundary



- Accuracy = 100%
- Linear SVM is same as "SVM with linear kernel"
- The SVM is trained on 75% of the dataset, and evaluated on the remaining 25%; Circled data points are from the test set.

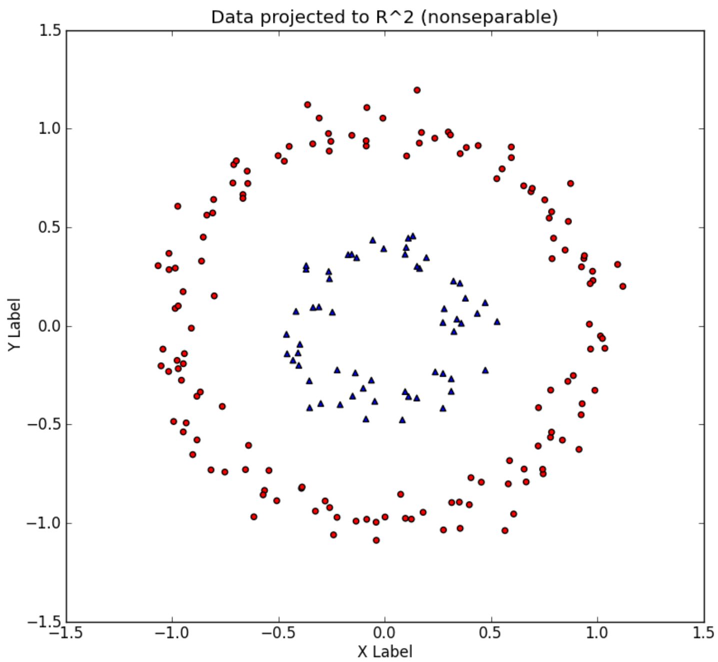
Linearly Non-Separable Data Set

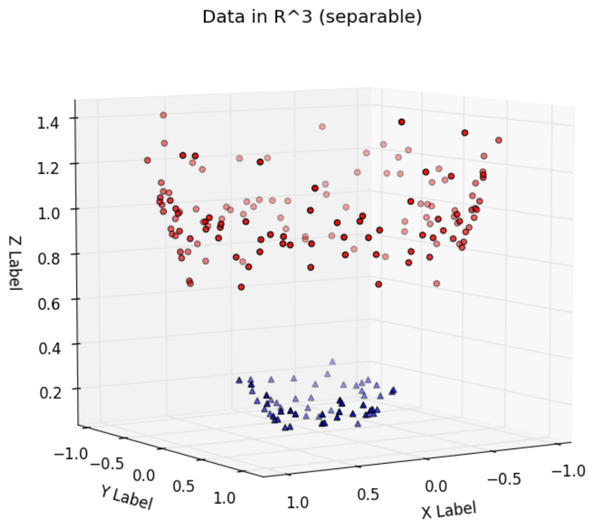


- A two-class dataset that is not linearly separable
- The outer ring (cyan) is class '0', inner ring (red) is class '1'
- No line in R^2 can reasonably separate the two classes - thus, we expect that a linear SVM will perform poorly on this dataset
  - Accuracy of a random classifier = 0.45
  - Accuracy of the best trained linear SVM = 0.445
- Primary problem is the constraint that the decision boundary be linear in the original feature space (R^2)
- Could we generalize SVM to discover decision boundaries with arbitrary shape?
- We are stuck with an SVM that, for an *N*-dimensional dataset, finds an *(N-1)*-dimensional separating hyperplane
- Key idea: Increase *N* by mapping the original data set

PURDUE UNIVERSITY

Idea: Separable in Higher Dimension



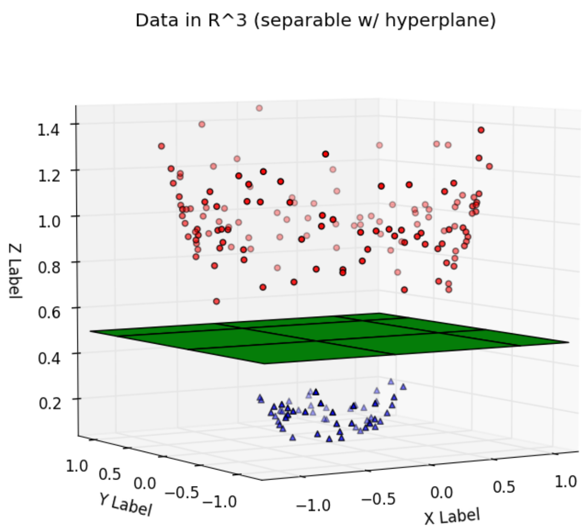


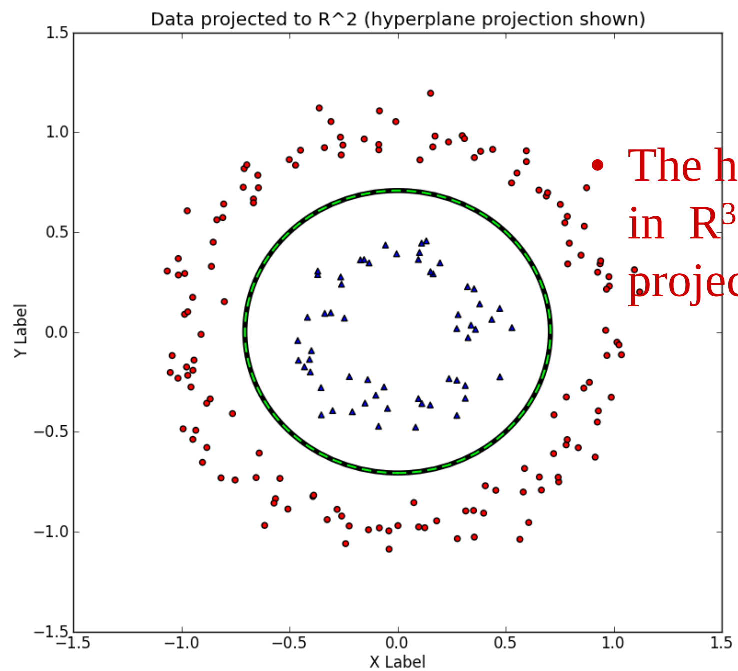
- Imagine that left dataset is merely a 2-D version of the 'true' dataset that lives in R^3 (right figure)
- The R^3 dataset is easily linearly separable by a hyperplane $\Rightarrow$ We can train a linear SVM classifier that successfully finds a good decision boundary
- Challenge is to find a transformation $T: R^2 \rightarrow R^3$, such that the transformed dataset is linearly separable
  - Here $[x_1, x_2] \rightarrow [x_1^2, x_2^2, x_1^2+x_2^2]$

PURDUE
UNIVERSITY

Non-linear SVM

- Pipeline to apply non-linear SVM:
  1. Transform the training set X to X' using ϕ
  2. Train a linear SVM on to get classifier f_{SVM} .
  3. At test time, a new example x will first be transformed to $x' = \phi(x)$
  4. The output class label is then determined by $f_{SVM}(x')$



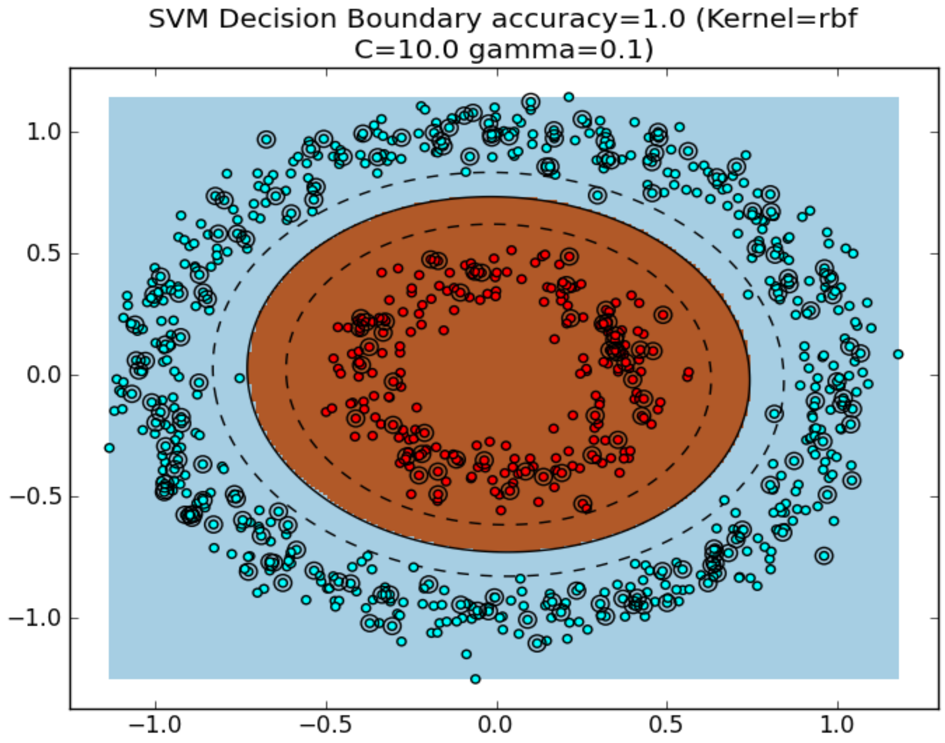


- The hyperplane learned in R^3 is nonlinear when projected back to R^2.

Non-linear SVM: Kernel Trick

- No need to explicitly transform the data points to the higher dimensional space
- Instead use a kernel to implicitly transform datasets to a higher-dimensional one using no extra memory, and with a minimal effect on computation time
- We can efficiently learn nonlinear decision boundaries for SVMs simply by replacing all dot products in the SVM computation with $K(\vec{x}_i, \vec{x}_j)$
  - Intuition: A kernel K effectively computes dot products in a higher-dimensional space R^M while remaining in R^N
- Most off-the-shelf classifiers allow the user to specify one of three popular kernels: the *polynomial* (2 parameters), *radial basis function* (1 parameter), and *sigmoid* (1 parameter) kernel

Application to our Canonical Example



- Decision boundary with a RBF kernel
  - Similar boundaries with the polynomial and Sigmoid kernels
- Accuracy is very high

PURDUE
UNIVERSITY

- Problem 3 - **Support Vector Machine**

Kernels allow us to make complex, non-linear classifiers using Support Vector Machines.

Given x, compute new feature depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$.

To do this, we find the "similarity" of x and some landmark $l^{(i)}$:

$$f_i = similarity(x, l^{(i)}) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2}\right)$$

What kernel is this?

1. Sigmoid
2. Radial basis function
3. Polynomial

What is the range of values of this similarity function?

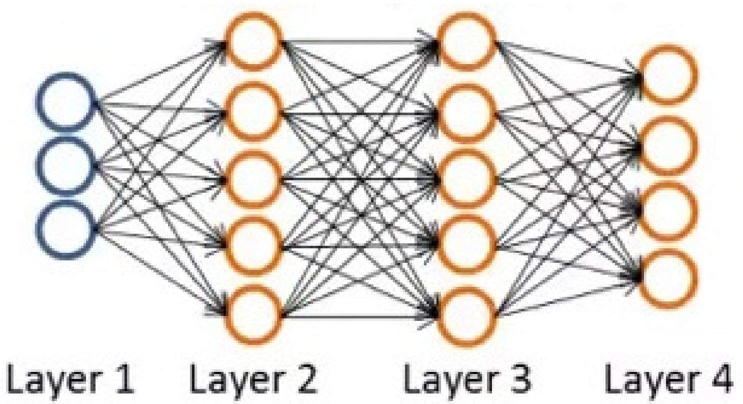
- Problem 4 - **Good Model**

Let's say I have a training data set of size *M*, validation data set of size *N*, and test data set of size *P*. I want to learn a model by minimizing some cost function. The cost function will have a sum of how many terms? Here I am talking of creating one iteration of the model, which I will then refine by using the validation data set.

1. *M*
2. *P*
3. *M*+*N*

Neural Network

- A powerful modeling formulation that is being used everywhere these days
  – Some of these uses are wildly inappropriate
  – But a neural network holds the promise of being able to model arbitrarily complex functions



- A 4 layer network
  – Today's leading edge NNs have 100+ layers
- Use for classification
  – 4 output layers means 4-way classification
  – 1 output layer for binary classification

Canonical Use Case of Neural Network

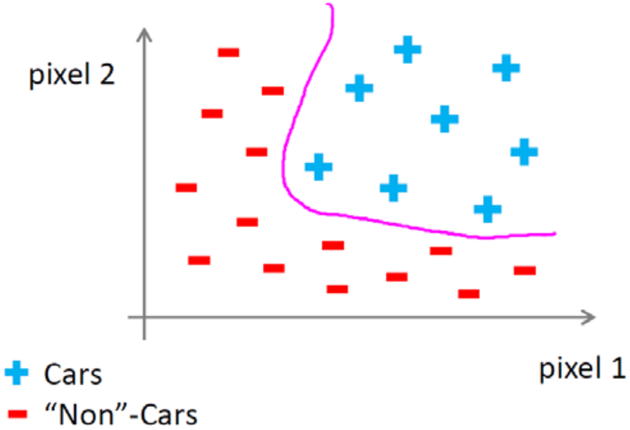
You see this

Camera sees this

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

50 × 50 pixel image = 2,500 pixels

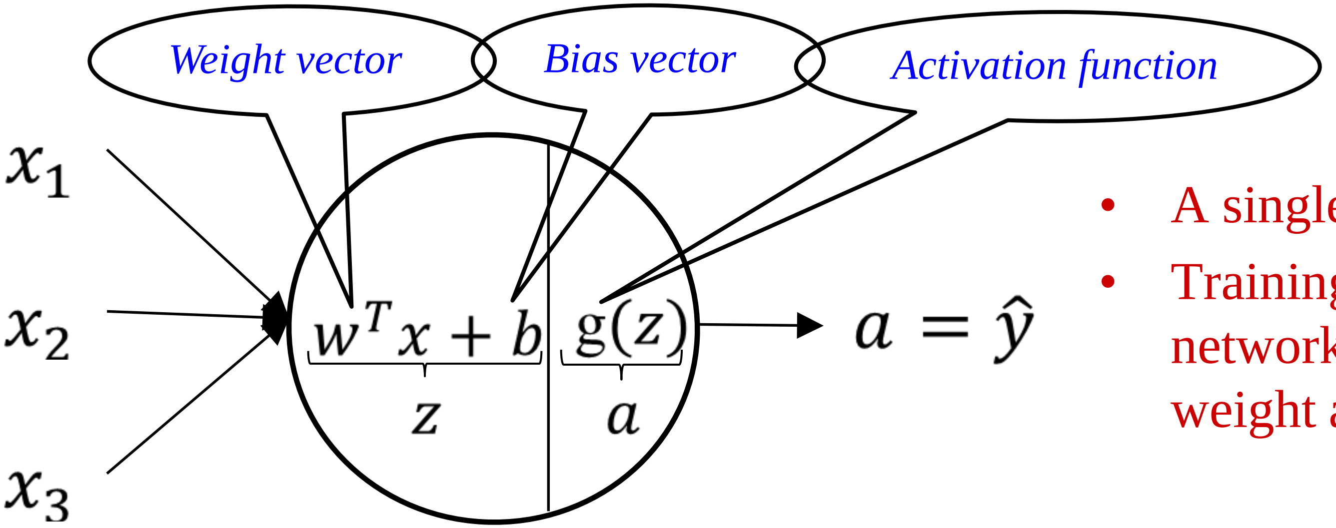
Let’s say we use the intensity value of each pixel (0-255)



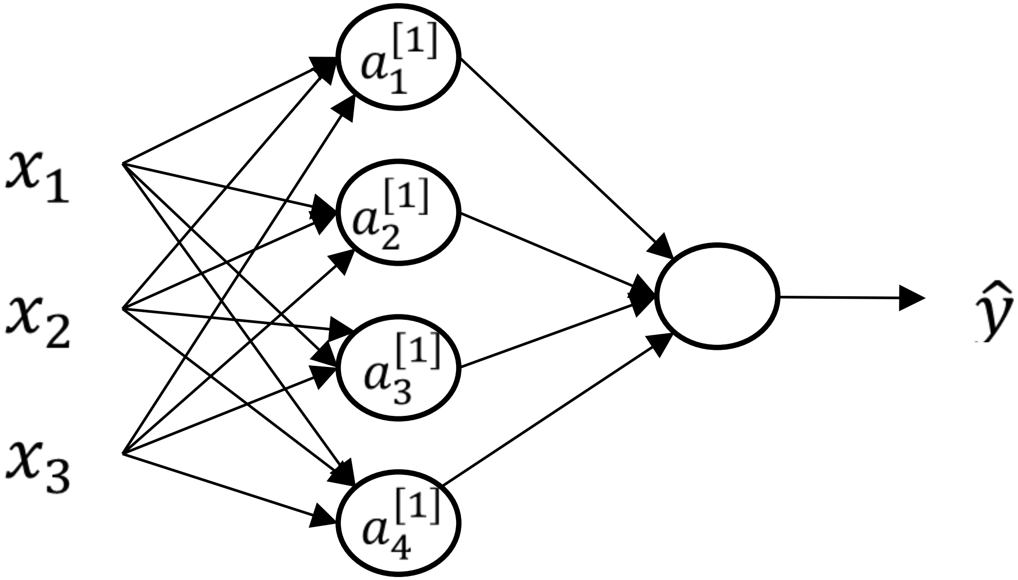
- Create a complex decision boundary to say if the image is of a car or not a car

PURDUE
UNIVERSITY

Neural Network: Basic Block



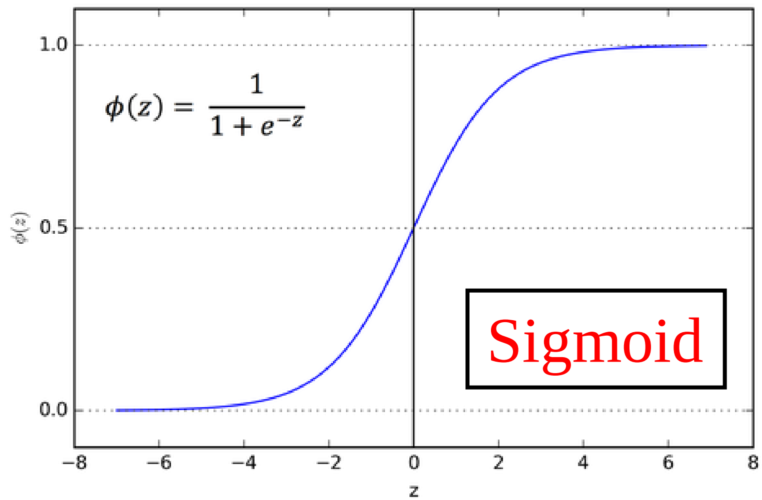
- A single neuron
- Training a neural network creates the weight and bias vectors

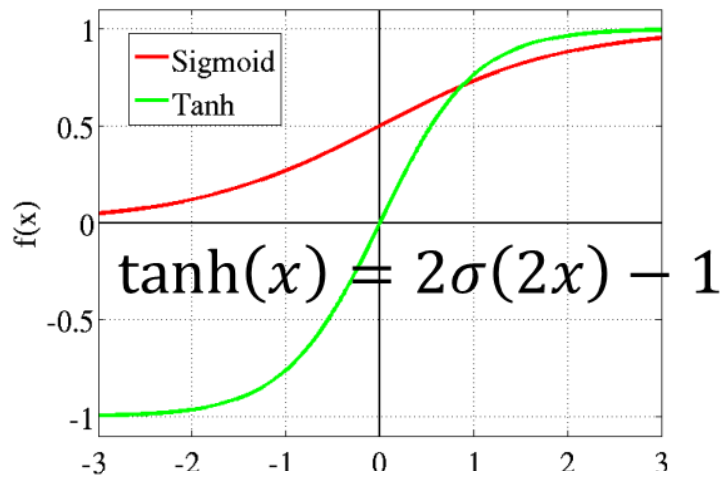


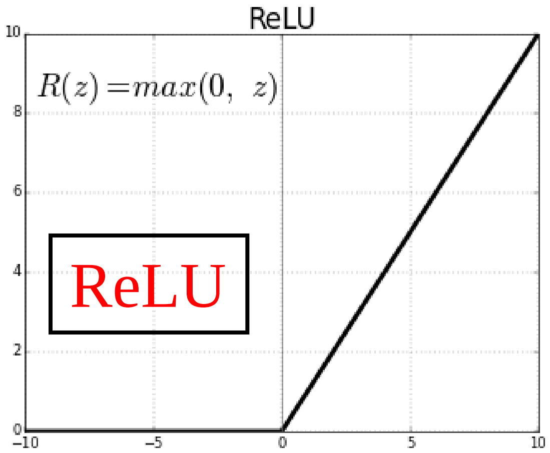
- 3 layer NN $\Rightarrow$ 2 weight vectors and 2 bias vectors

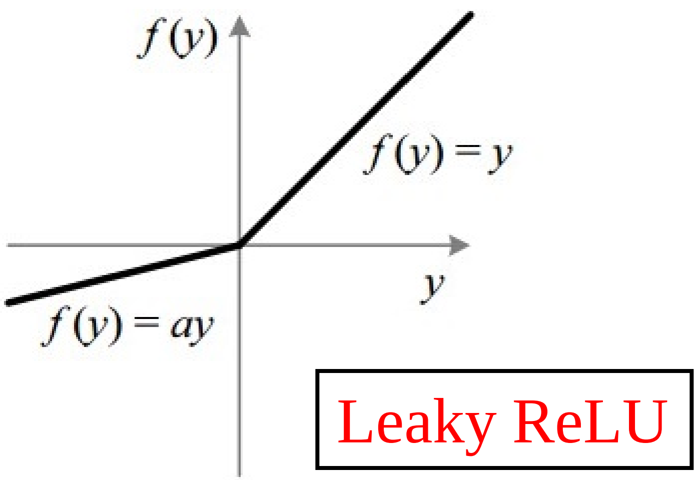
Neural Network: Activation Function

- Many different activation functions (*g* in previous slide) are used









PURDUE
UNIVERSITY

Use of NN in Computer Systems

- Deep Neural Networks (DNN) were an early generation of NNs
  - Needs a lot of data to train
  - Needs a lot of time to train
- Newer NNs such as Convolutional Neural Networks (CNNs) have fewer parameters to train
  - Especially successful in image processing domain
  - Have to choose appropriate filters for high accuracy
- NNs that can carry historical state in making decisions are being used – example: Recurrent Neural Networks (RNN)
  - Especially successful in language modeling
  - Expensive even in the inference stage, especially its most popular variant - Long Short-Term Memory (LSTM) network

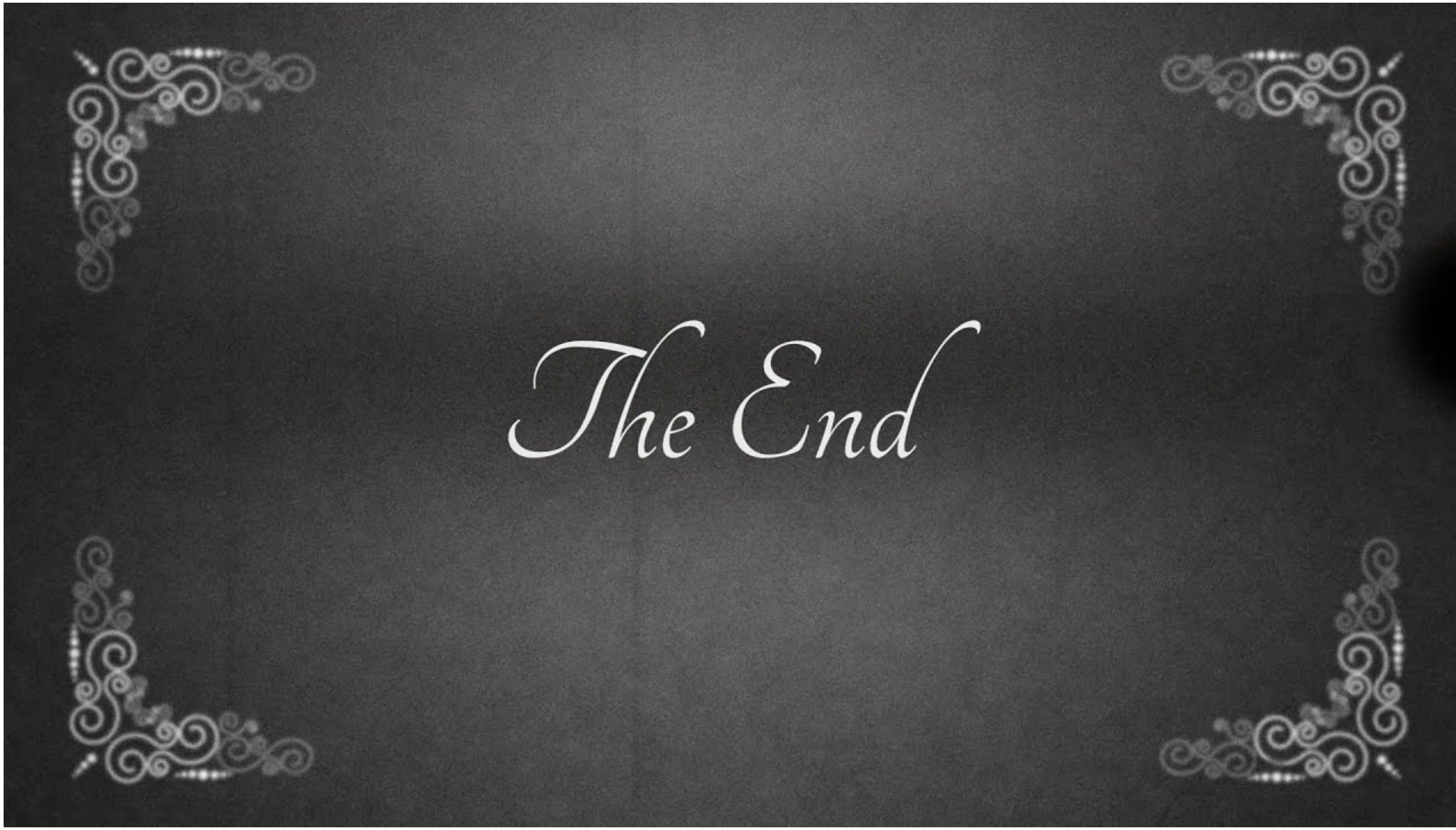
- Problem 5 – **Neural Networks**

Which of the following are generally true about neural networks? (Check all that apply.)

1. Decreasing training set size generally does not hurt an algorithm’s performance, and it may help significantly.
2. Increasing training set size generally does not hurt an algorithm’s performance, and it may help significantly.
3. Increasing the size of a neural network generally does not hurt an algorithm’s performance, and it may help significantly.
4. Decreasing the size of a neural network generally does not hurt an algorithm’s performance, and it may help significantly.

Recap

- Supervised versus unsupervised learning
- Training, validation, and test data sets
- What makes for a good model?
- Linear and non-linear SVM
- Neural network – basic block, activation function, multiple layers



Material at: ***http://bit.ly/whin-big-data-algorithms***