LEDE Algorithms

Richard Dunks

Chase Davis

WEEK 6
CLASS 2

Theses slides are based on the slides by

- Tan, Steinbach and Kumar (textbook authors)
- Eamonn Koegh (UC Riverside)
- Andrew Moore (CMU/Google)

Goals for today

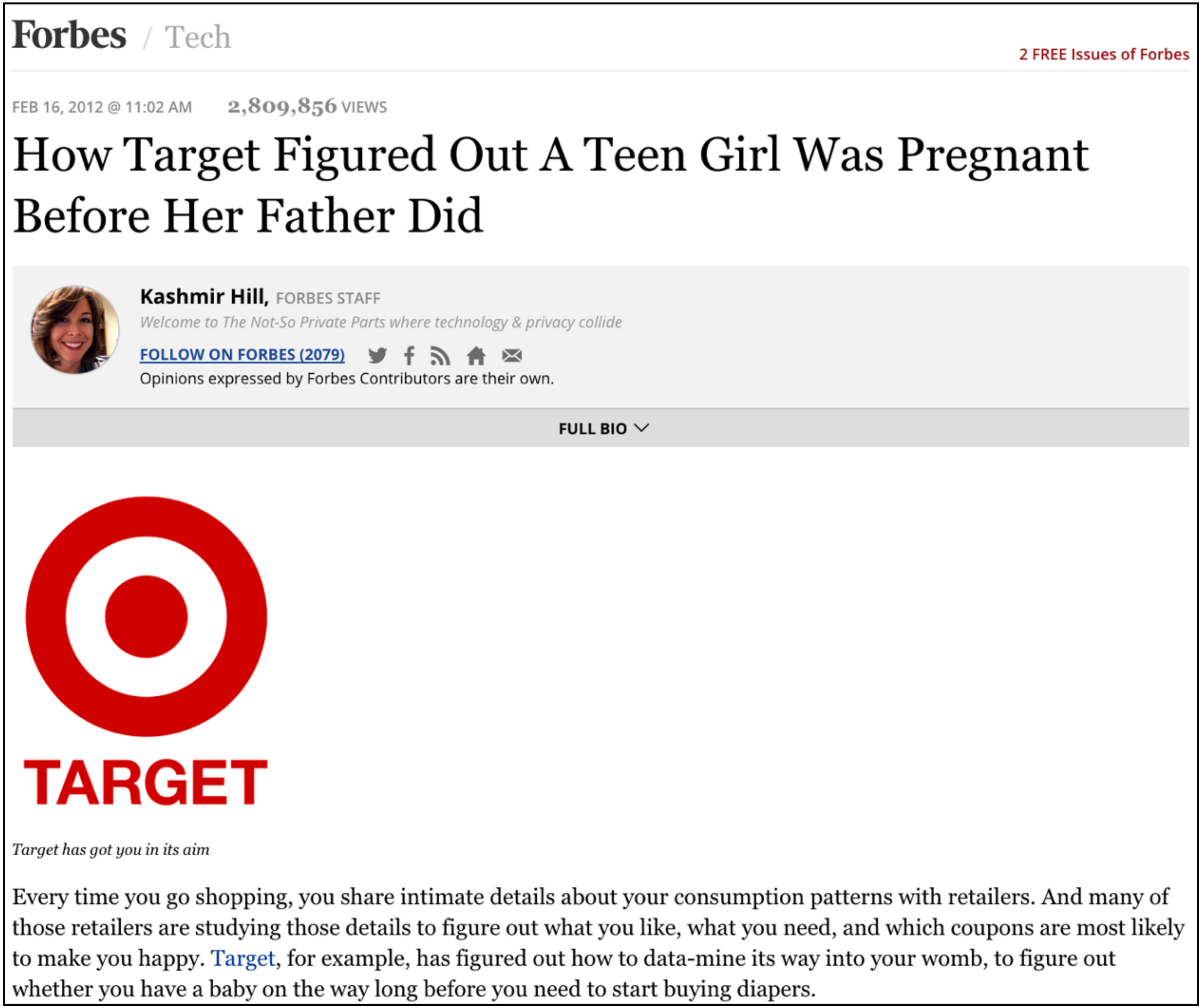
- Discuss Association Rule Mining
- Discuss Neural Networks
- Discuss Random Forests

Basic Process

1. Start with an intuition (maybe even a hypothesis)
2. Gather data
3. Clean data
4. Clean data some more
5. Do some more cleaning
6. Run a model
7. Test the results
8. Change parameters and run another model
9. Clean the data some more and repeat step 8 as necessary

Association Rule Mining

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects
- Data usually stored in transaction databases, relational databases, and other information repositories
- What items occur together?
  - ex -> Bread, milk



Forbes / Tech

2 FREE Issues of Forbes

FEB 16, 2012 @ 11:02 AM 2,809,856 VIEWS

How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did

Kashmir Hill, FORBES STAFF

Welcome to The Not-So Private Parts where technology & privacy collide

FOLLOW ON FORBES (2079)

Opinions expressed by Forbes Contributors are their own.

FULL BIO

Target has got you in its aim

Every time you go shopping, you share intimate details about your consumption patterns with retailers. And many of those retailers are studying those details to figure out what you like, what you need, and which coupons are most likely to make you happy. Target, for example, has figured out how to data-mine its way into your womb, to figure out whether you have a baby on the way long before you need to start buying diapers.

http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/

Association Rule Mining Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., 98% of people who purchase tires and auto accessories also get automotive services done

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

An itemset is simply a set of items

Association Rule Mining

- We are interested in rules that are
  - non-trivial (and possibly unexpected)
  - actionable
  - easily explainable

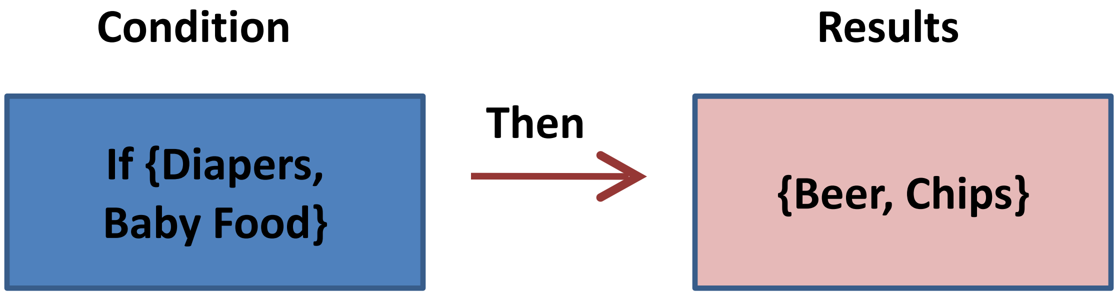
Supermarket Example

- If you have a rule X ➔ Y, you could:
  - Run a sale on X if you want to increase sales of Y
  - Locate the two items near each other
  - Locate the two items far from each other to make the shopper walk through the store
  - Print out a coupon on checkout for Y if shopper bought X but not Y

Association Rules Standard Format

Rule format: (*A set can consist of just a single item*)

If {set of items} → Then {set of items}



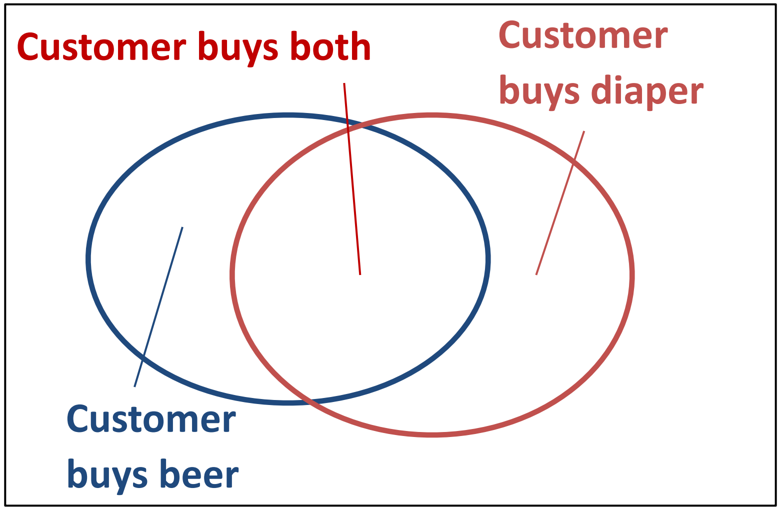
Condition

Results

If {Diapers, Baby Food}

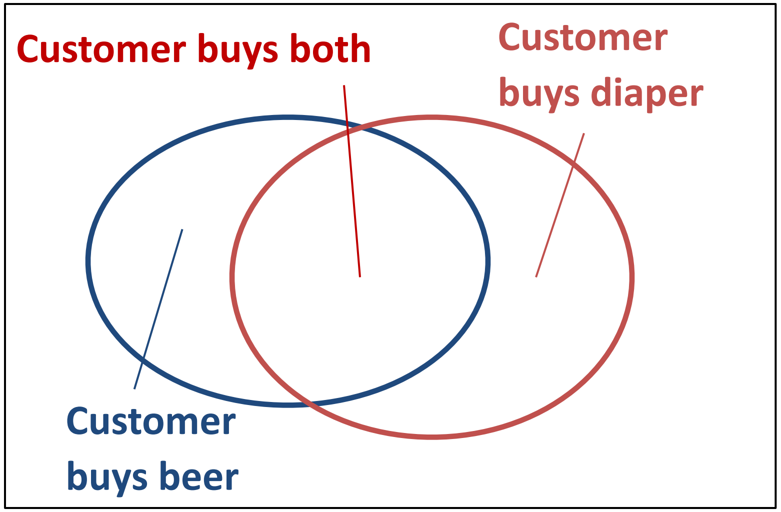
Then

{Beer, Chips}



Condition implies Results

Support and Confidence



- Find all the rules $X \Rightarrow Y$ with minimum confidence and support
  - Support = probability that a transaction contains {X,Y}
    - i.e., ratio of transactions in which X, Y occur together to all transactions in database.
  - Confidence = conditional probability that a transaction having X also contains *Y*
    - i.e., ratio of transactions in which X, Y occur together to those in which X occurs.

In general confidence of a rule C => R can be computed as the support of the whole itemset divided by the support of LHS:

Confidence (C => R) = Support(C ∪ R) / Support(C)

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Definition: Association Rule

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

- Association Rule
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Mining Association Rules

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

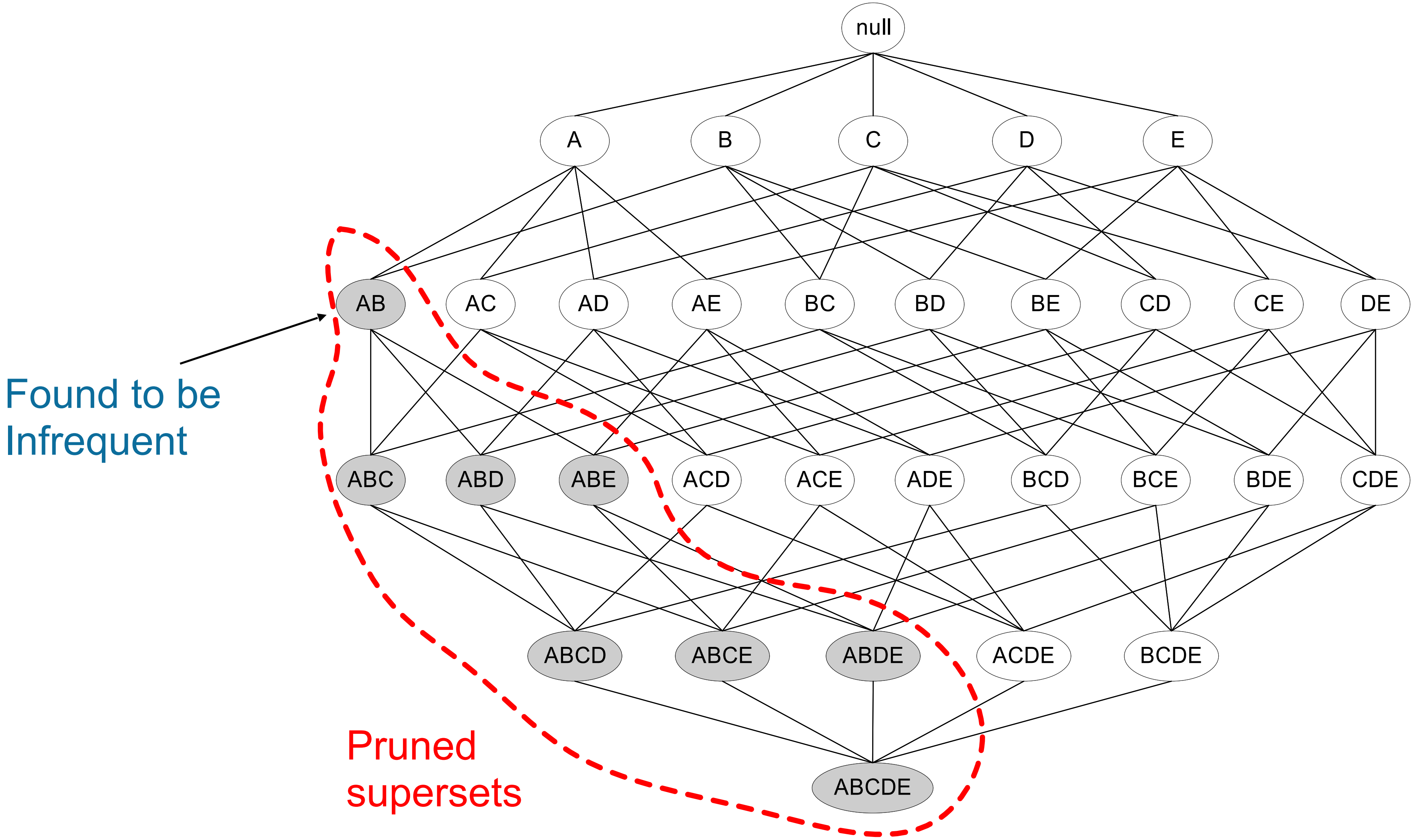
Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Apriori Algorithm

- The best known algorithm
- Two steps:
  - Find all itemsets that have minimum support (frequent itemsets, also called large itemsets)
  - Use frequent itemsets to generate rules
- Downward closure property requires that a subset of a frequent itemset must also be a frequent itemset

Illustrating Apriori Principle



Found to be Infrequent

Pruned supersets

Apriori Algorithm Example

- Transaction data →

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

- Assume:

  minsup = 30%

  minconf = 80%

- An example frequent *itemset*:

  {Chicken, Clothes, Milk} [sup = 3/7]

- Association rules from the itemset:

  Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

  … …

  Clothes, Chicken → Milk, [sup = 3/7, conf = 3/3]

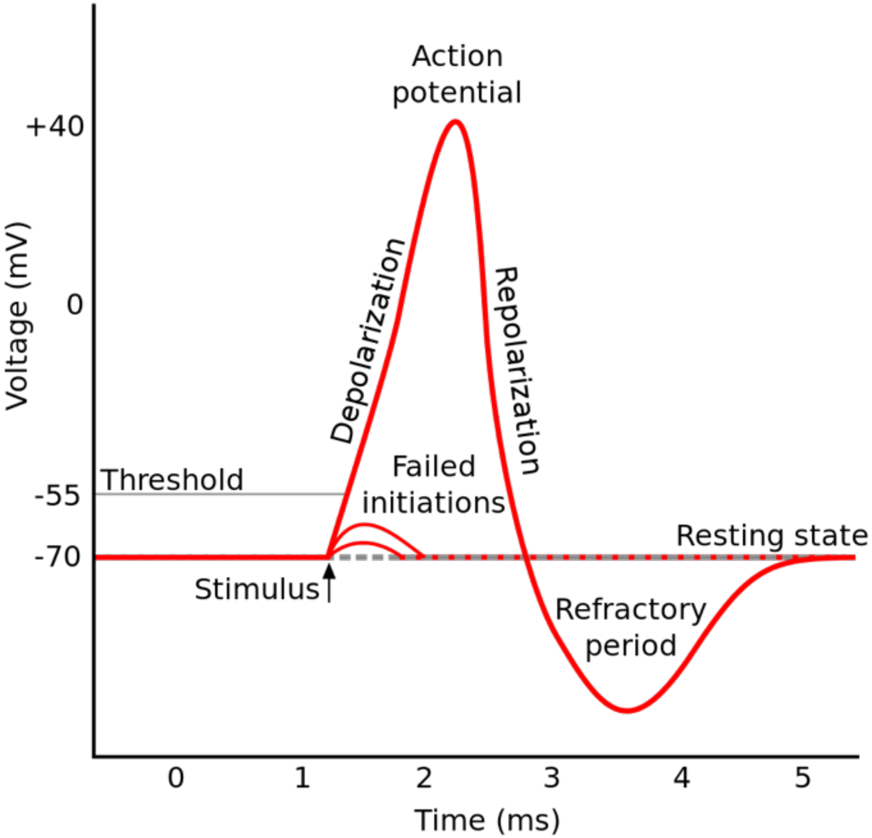
10 MIN BREAK

YOU'RE TRYING TO PREDICT THE BEHAVIOR OF <COMPLICATED SYSTEM>? JUST MODEL IT AS A <SIMPLE OBJECT>, AND THEN ADD SOME SECONDARY TERMS TO ACCOUNT FOR <COMPLICATIONS I JUST THOUGHT OF>.
EASY, RIGHT?
SO, WHY DOES <YOUR FIELD> NEED A WHOLE JOURNAL, ANYWAY?
LIBERAL-ARTS MAJORS MAY BE ANNOYING SOMETIMES, BUT THERE'S NOTHING MORE OBNOXIOUS THAN A PHYSICIST FIRST ENCOUNTERING A NEW SUBJECT.

http://www.xkcd.com/793/

Learning in the Human Brain

- A neuron is connected to other neurons via its input and output links
- Each incoming neuron has an activation value and each connection has a weight associated with it
- The neuron sums the incoming weighted values and this value is input to an activation function
- The output of the activation function is the output from the neuron



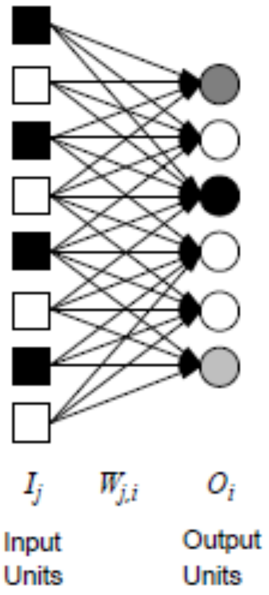
Source: https://commons.wikimedia.org/wiki/File:Action_potential.svg

Artificial Neural Network

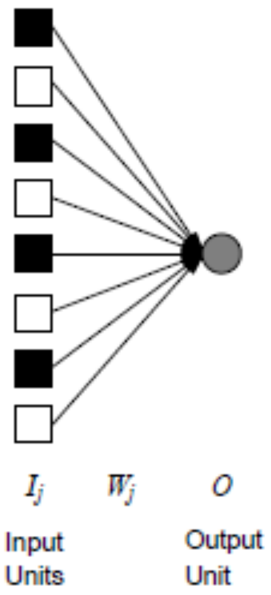
- Network of simple units
- Real-valued inputs & outputs
  - Many neuron-like threshold switching units
  - Many weighted interconnections among units
  - Highly parallel, distributed process
  - Emphasis on tuning weights automatically

Perceptron

- Initial algorithm for learning simple neural networks (single layer) developed in the 1950's
- Feed-forward neural network (data only travels in one direction)



Perceptron Network



Single Perceptron

Perceptron Training

- Assume supervised training examples giving the desired output for a unit given a set of known input activations
- Goal: learn the weight vector (synaptic weights) that causes the perceptron to produce the correct +/- 1 values
- Perceptron uses iterative update algorithm to learn a correct set of weights
  - Perceptron training rule
  - Delta rule

Perceptron Training Rule

- Update weights by: $w_i \leftarrow w_i + \Delta w_i$

$$\Delta w_i = \eta (t - o) w_i$$

where

- η is the learning rate
  - a small value (e.g., 0.1)
  - sometimes is made to decay as the number of weight-tuning operations increases
- t – target output for the current training example
- o – linear unit output for the current training example

Perceptron Training Rule

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs
- Can prove it will converge
  - if training data is linearly separable
  - and η is sufficiently small

Perceptron Learning Algorithm

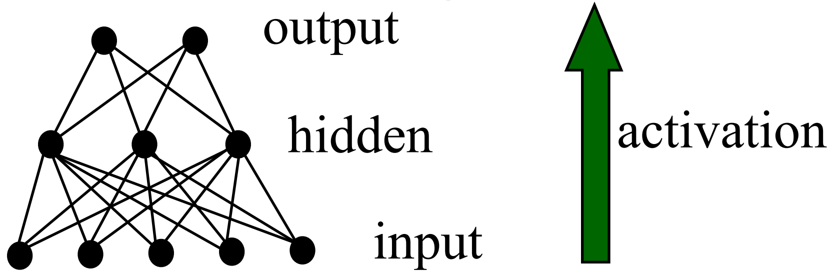
- Iteratively update weights until convergence

```
Initialize weights to random values
Until outputs of all training examples are correct
    For each training pair, E, do:
        Compute current output $o_j$ for E given its inputs
        Compare current output to target value, $t_j$ , for E
        Update synaptic weights and threshold using learning rule
```

- Each execution of the outer loop is typically called an epoch

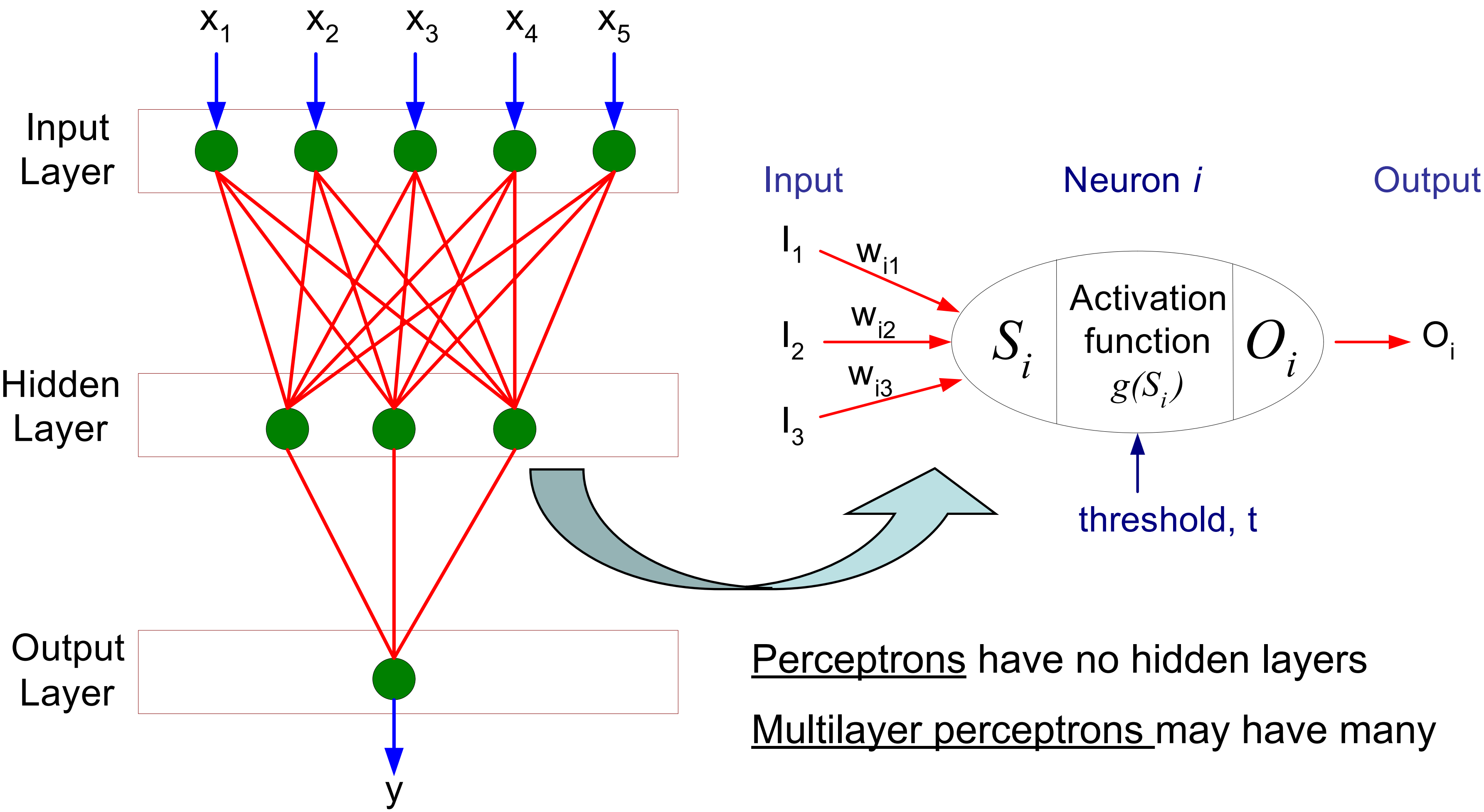
Multi-Layer Networks

- Multi-layer networks can represent arbitrary functions, but an effective learning algorithm for such networks was thought to be difficult
- A typical multi-layer network consists of an input, hidden and output layer, each fully connected to the next, with activation feeding forward.



- The weights determine the function computed. Given an arbitrary number of hidden units, any boolean function can be computed with a single hidden layer

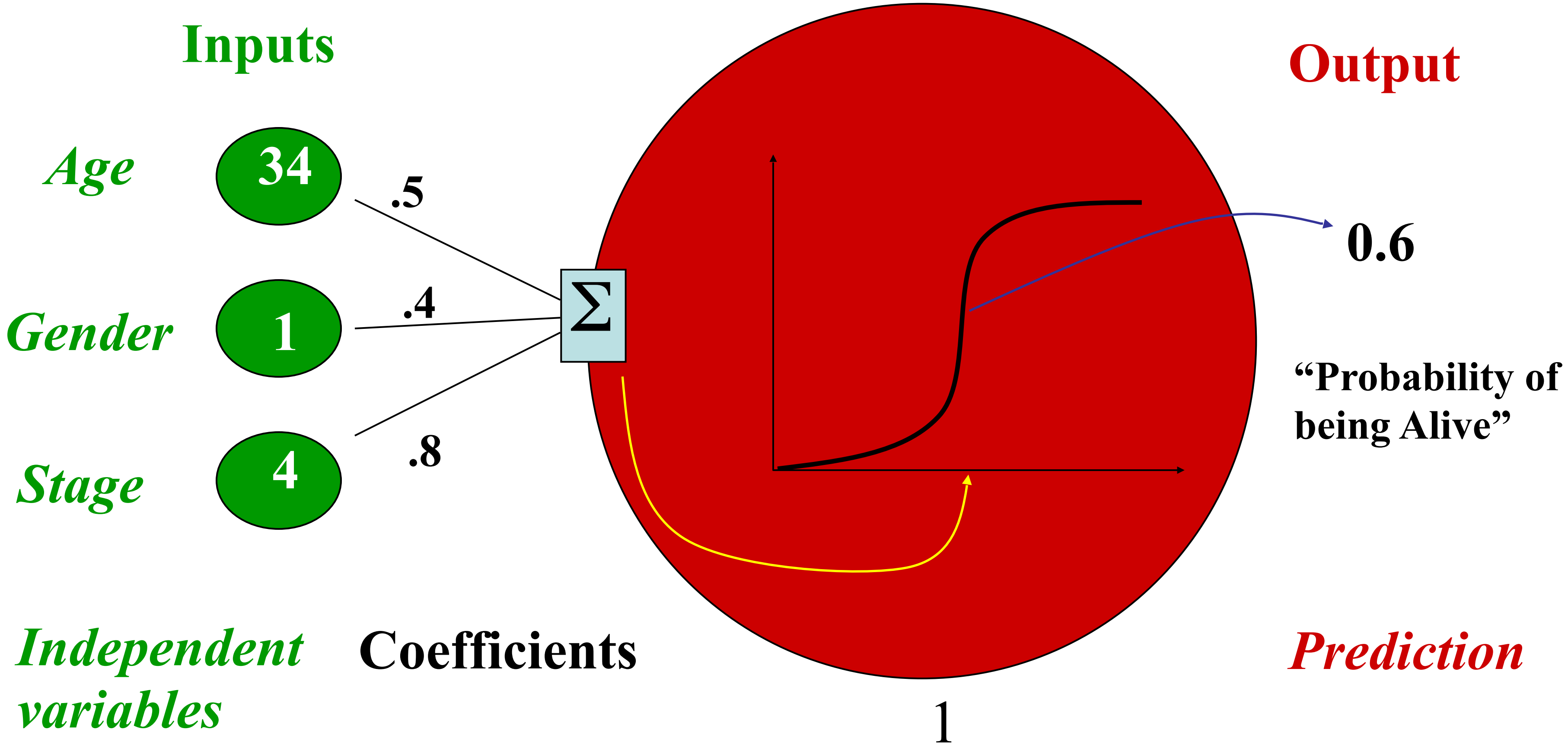
General Structure of an ANN



Perceptrons have no hidden layers

Multilayer perceptrons may have many

Logistic function



Inputs

Age 34

Gender 1

Stage 4

Output

0.6

“Probability of being Alive”

Independent variables

Coefficients

Prediction

$$\sigma = \frac{1}{1 + e^{-\sum_{i=0}^{n} w_i x_i}}$$

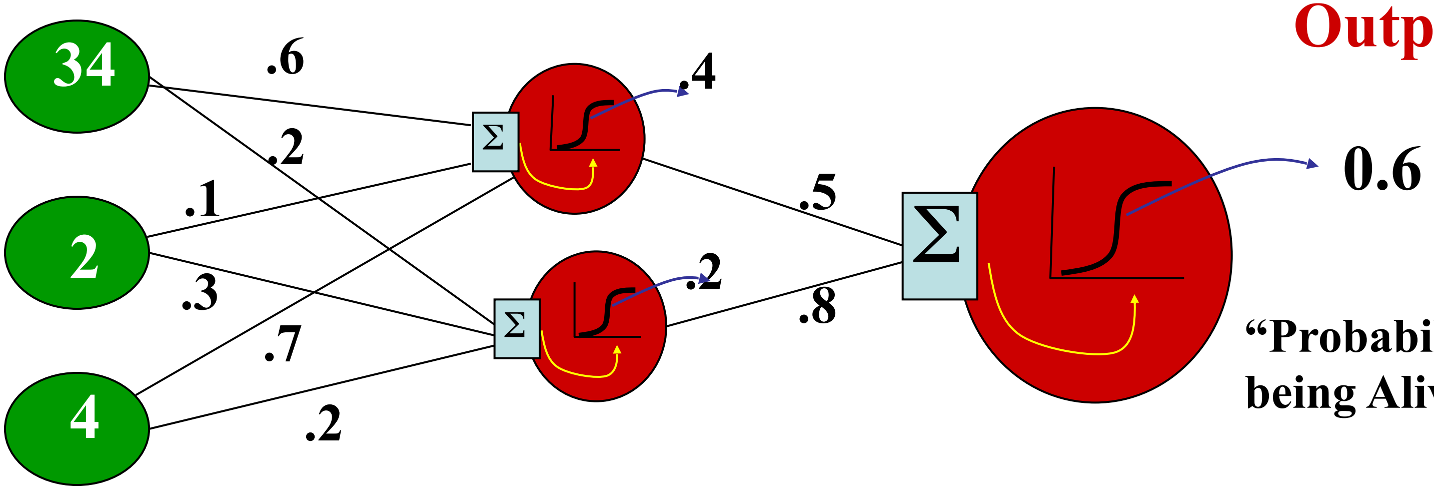
Neural Network Model

Inputs

Age

Gender

Stage



Output

"Probability of being Alive"

Independent variables

Weights

Hidden Layer

Weights

Dependent variable

Prediction

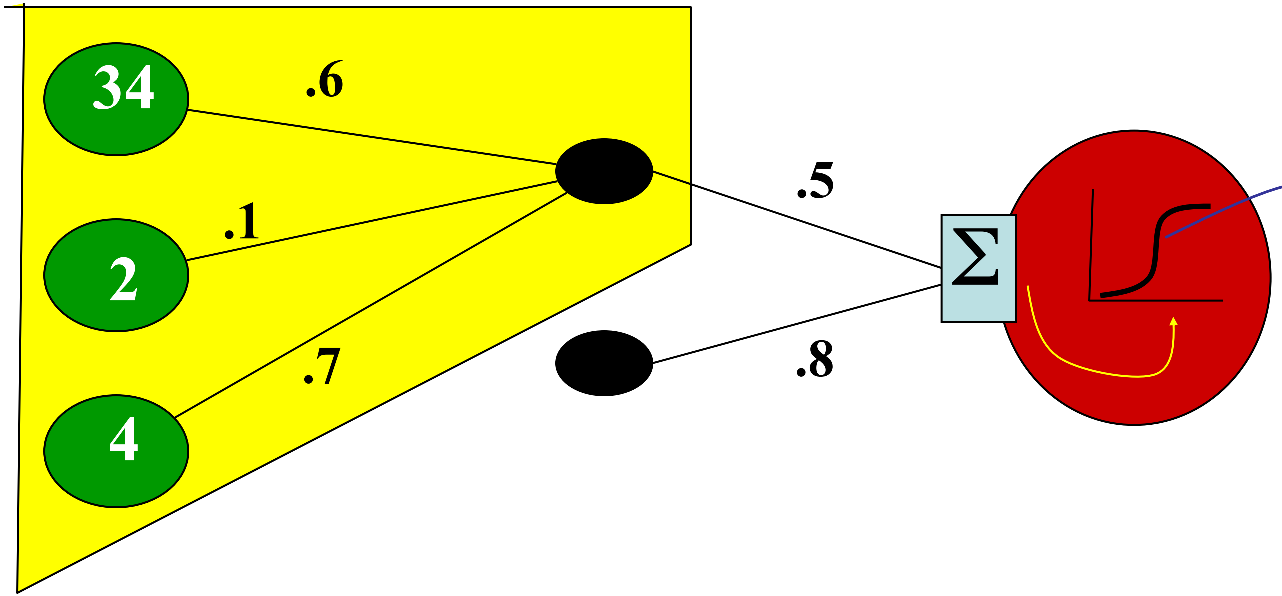
Getting an answer from a NN

Inputs

Age

Gender

Stage



Output

0.6

“Probability of being Alive”

Independent variables

Weights

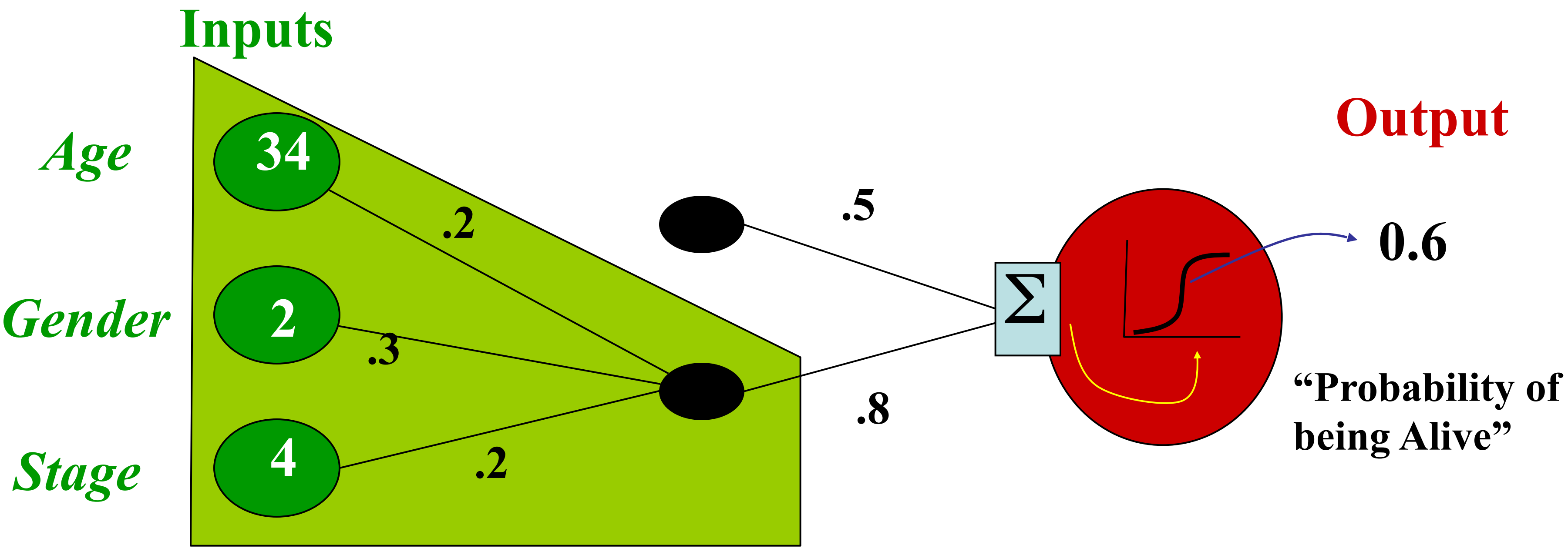
Hidden Layer

Weights

Dependent variable

Prediction

Getting an answer from a NN



Independent variables

Weights

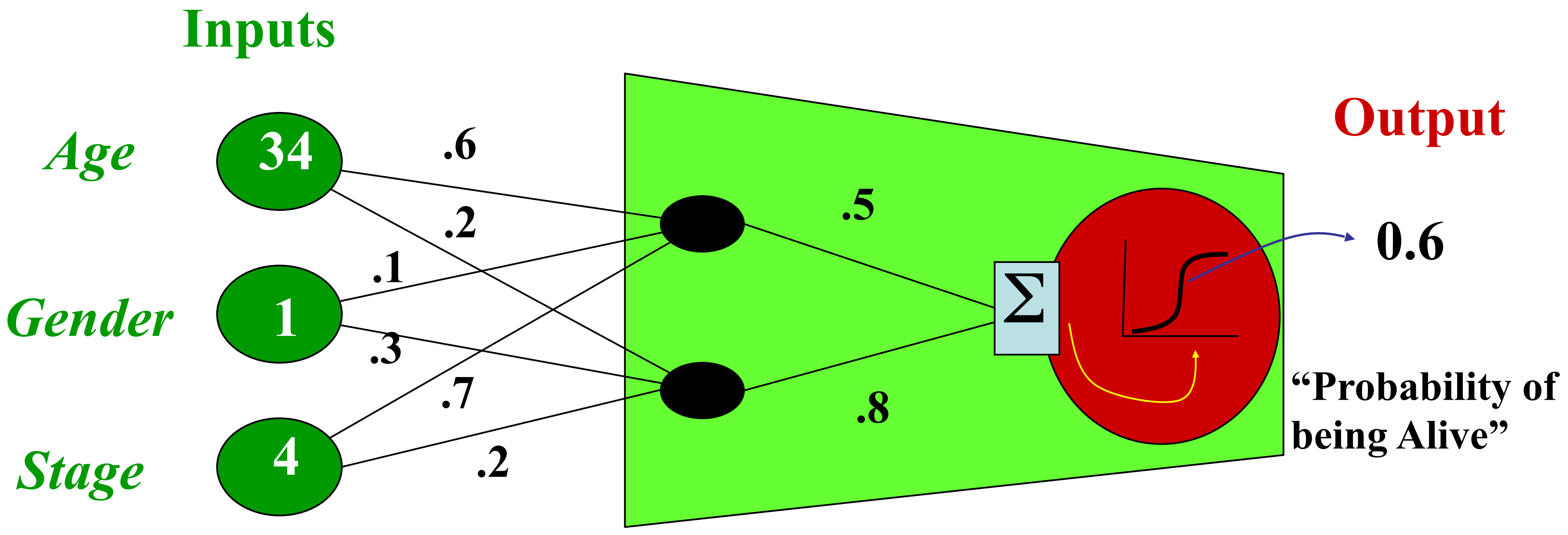
Hidden Layer

Weights

Dependent variable

Prediction

Getting an answer from a NN



Inputs

Age

Gender

Stage

Output

“Probability of being Alive”

Independent variables

Weights

Hidden Layer

Weights

Dependent variable

Prediction

Comments on Training Algorithm

- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely
- However, in practice, does converge to low error for many large networks on real data

Comments on Training Algorithms

- Many epochs (thousands) may be required, hours or days of training for large networks
- To avoid local-minima problems, run several trials starting with different random weights (random restarts)
  - Take results of trial with lowest training set error
  - Build a committee of results from multiple trials (possibly weighting votes by training set accuracy)

Hidden Unit Representations

- Trained hidden units can be seen as newly constructed features that make the target concept linearly separable in the transformed space: key features of ANNs
- On many real domains, hidden units can be interpreted as representing meaningful features such as vowel detectors or edge detectors, etc.
- However, the hidden layer can also become a distributed representation of the input in which each individual unit is not easily interpretable as a meaningful feature

Neural Networks

- Can we simulate the human learning process? Two schools of thought
  1. modeling biological learning process
  2. obtain highly effective algorithms, independent of whether these algorithms mirror biological processes

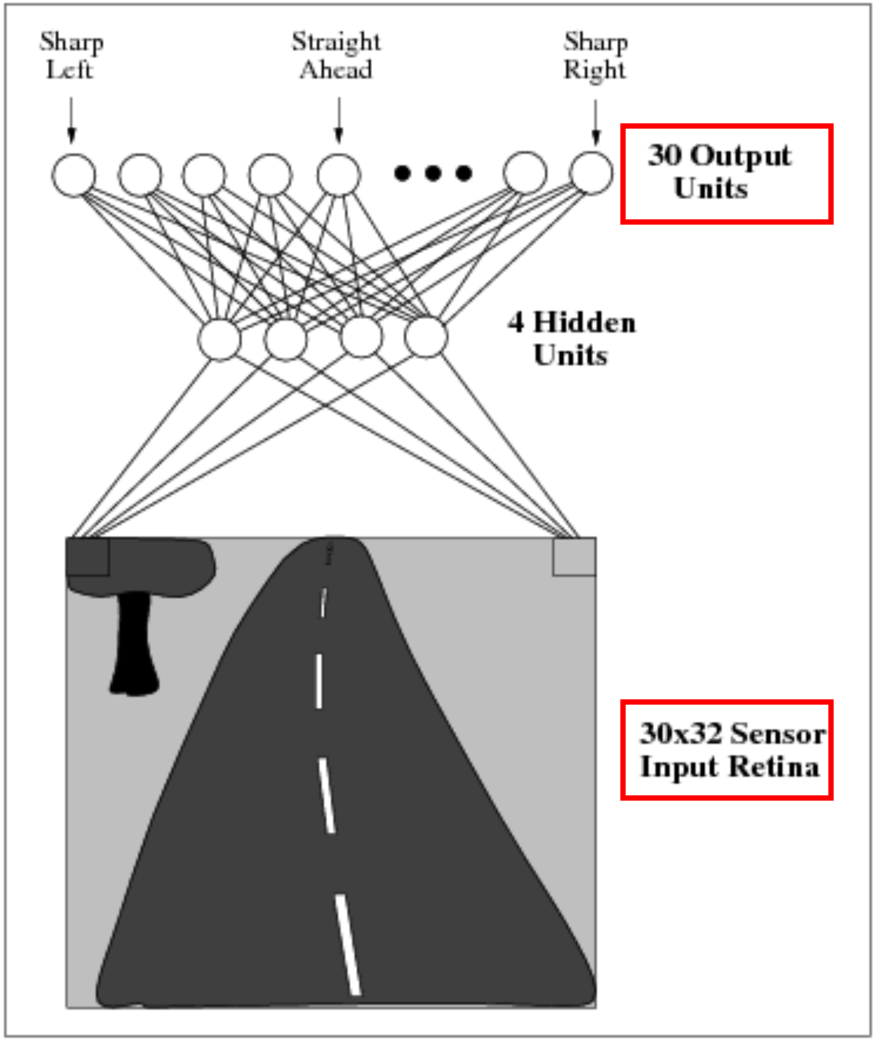
Successful Applications

- Text to Speech (NetTalk)
- Fraud detection
- Financial Applications
  - HNC (eventually bought by Fair Isaac)
- Chemical Plant Control
  - Pavillion Technologies
- Automated Vehicles
- Game Playing
  - Neurogammon
- Handwriting recognition

Recurrent Neural Networks

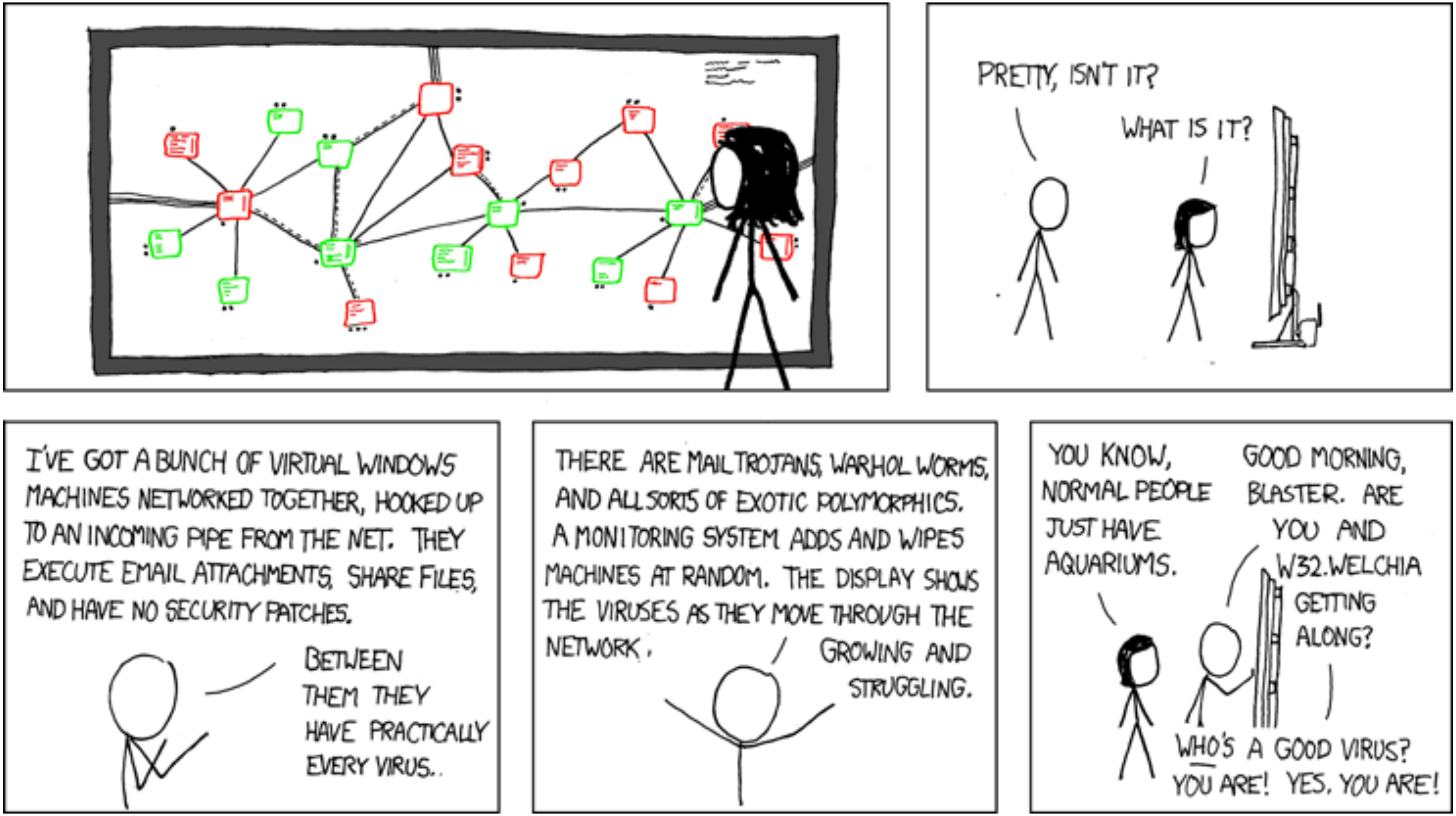
- A class of artificial neural networks with connections between units that create a directed cycle
- Creates an internal state for the network
- Network exhibits dynamic temporal behavior (changing with time)
- Have an internal memory to process inputs

ALVINN drives 70 mph on highways



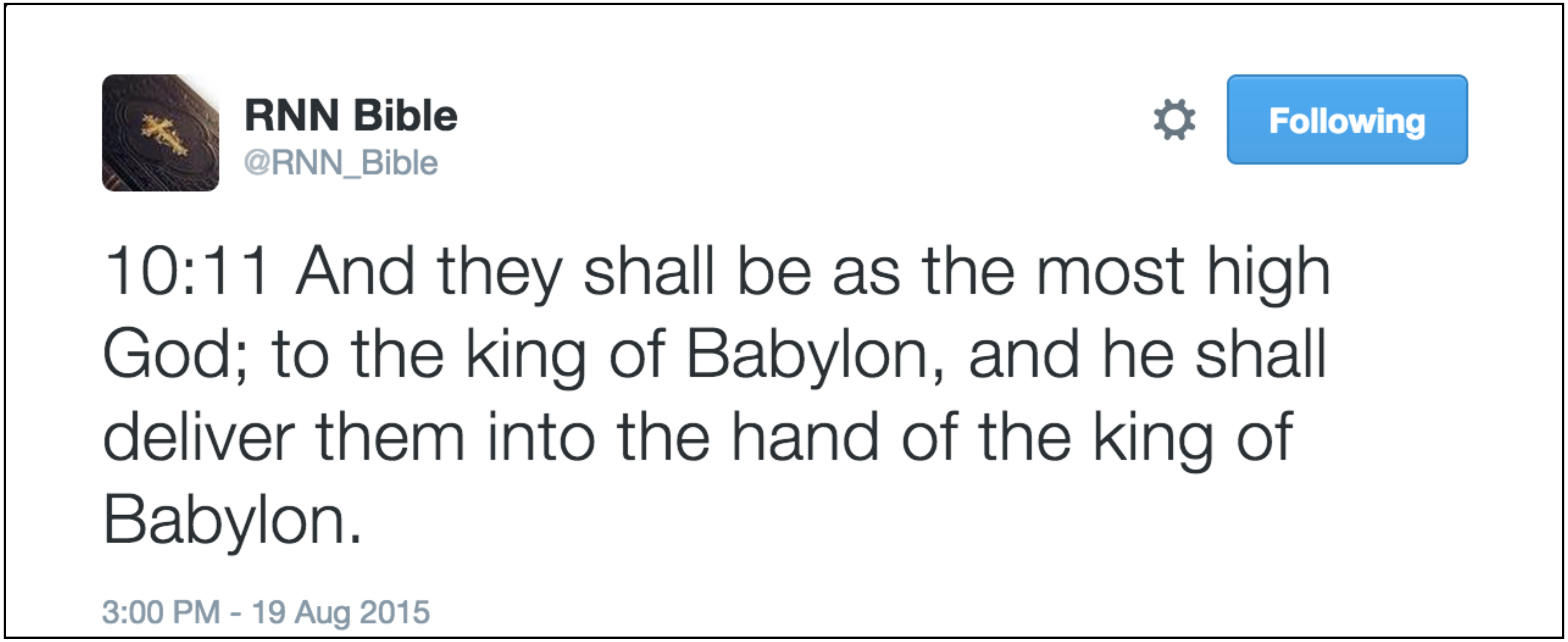
See Alvinn video
Alvinn Video

10 MIN BREAK


PRETTY, ISN'T IT?
WHAT IS IT?
I'VE GOT A BUNCH OF VIRTUAL WINDOWS MACHINES NETWORKED TOGETHER, HOOKED UP TO AN INCOMING PIPE FROM THE NET. THEY EXECUTE EMAIL ATTACHMENTS, SHARE FILES, AND HAVE NO SECURITY PATCHES.
BETWEEN THEM THEY HAVE PRACTICALLY EVERY VIRUS.
THERE ARE MAIL TROJANS, WARHOL WORMS, AND ALL SORTS OF EXOTIC POLYMORPHICS. A MONITORING SYSTEM ADDS AND WIPES MACHINES AT RANDOM. THE DISPLAY SHOWS THE VIRUSES AS THEY MOVE THROUGH THE NETWORK,
GROWING AND STRUGGLING.
YOU KNOW, NORMAL PEOPLE JUST HAVE AQUARIUMS.
GOOD MORNING, BLASTER. ARE YOU AND W32.WELCHIA GETTING ALONG?
WHO'S A GOOD VIRUS? YOU ARE! YES, YOU ARE!

https://xkcd.com/350/

Recurrent Neural Networks

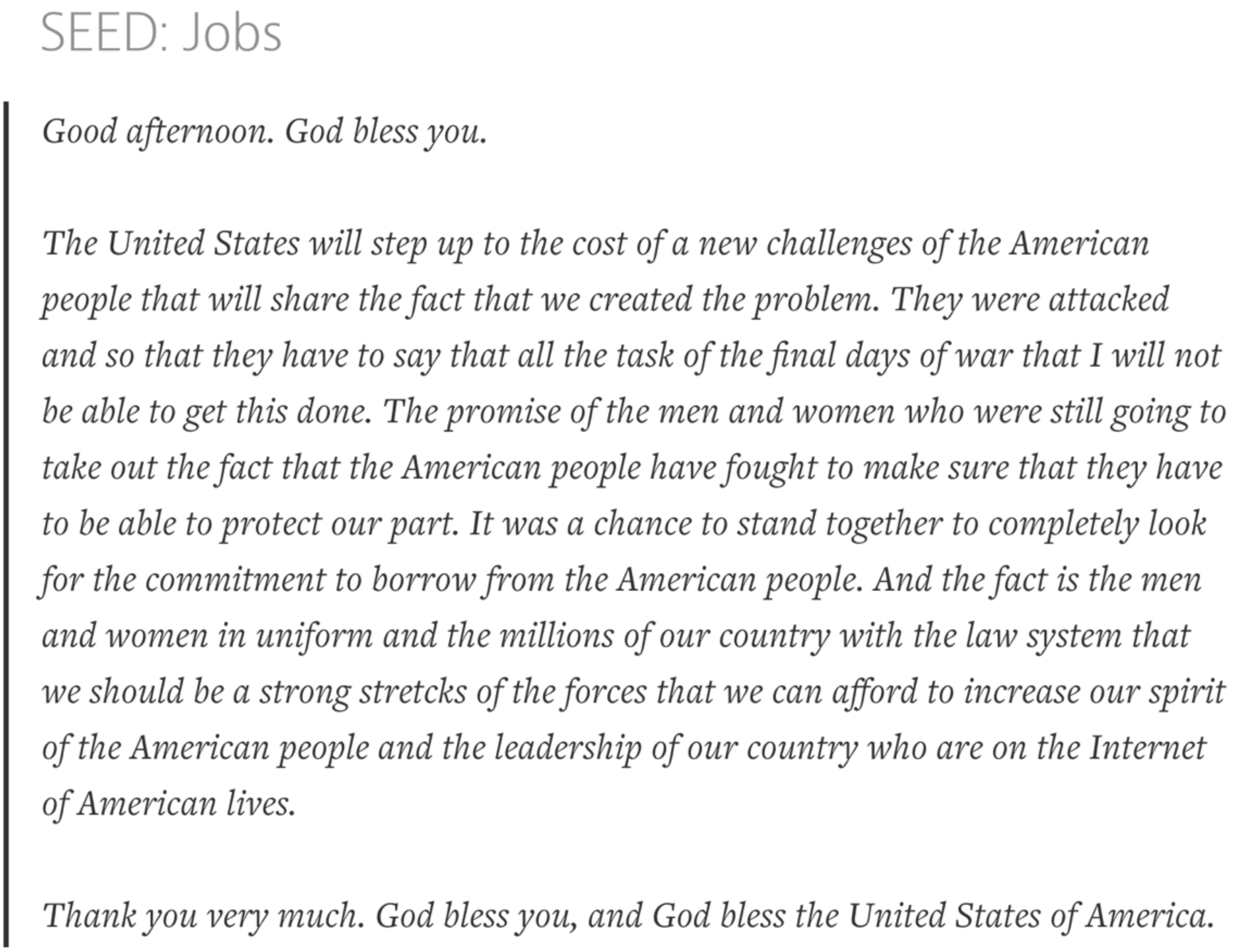


RNN Bible
@RNN_Bible

10:11 And they shall be as the most high God; to the king of Babylon, and he shall deliver them into the hand of the king of Babylon.

3:00 PM - 19 Aug 2015

Obama RNN



SEED: Jobs

> *Good afternoon. God bless you.*
>
> *The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretcks of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.*
>
> *Thank you very much. God bless you, and God bless the United States of America.*

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Code

You can run your own Obama-RNN by following these instructions:

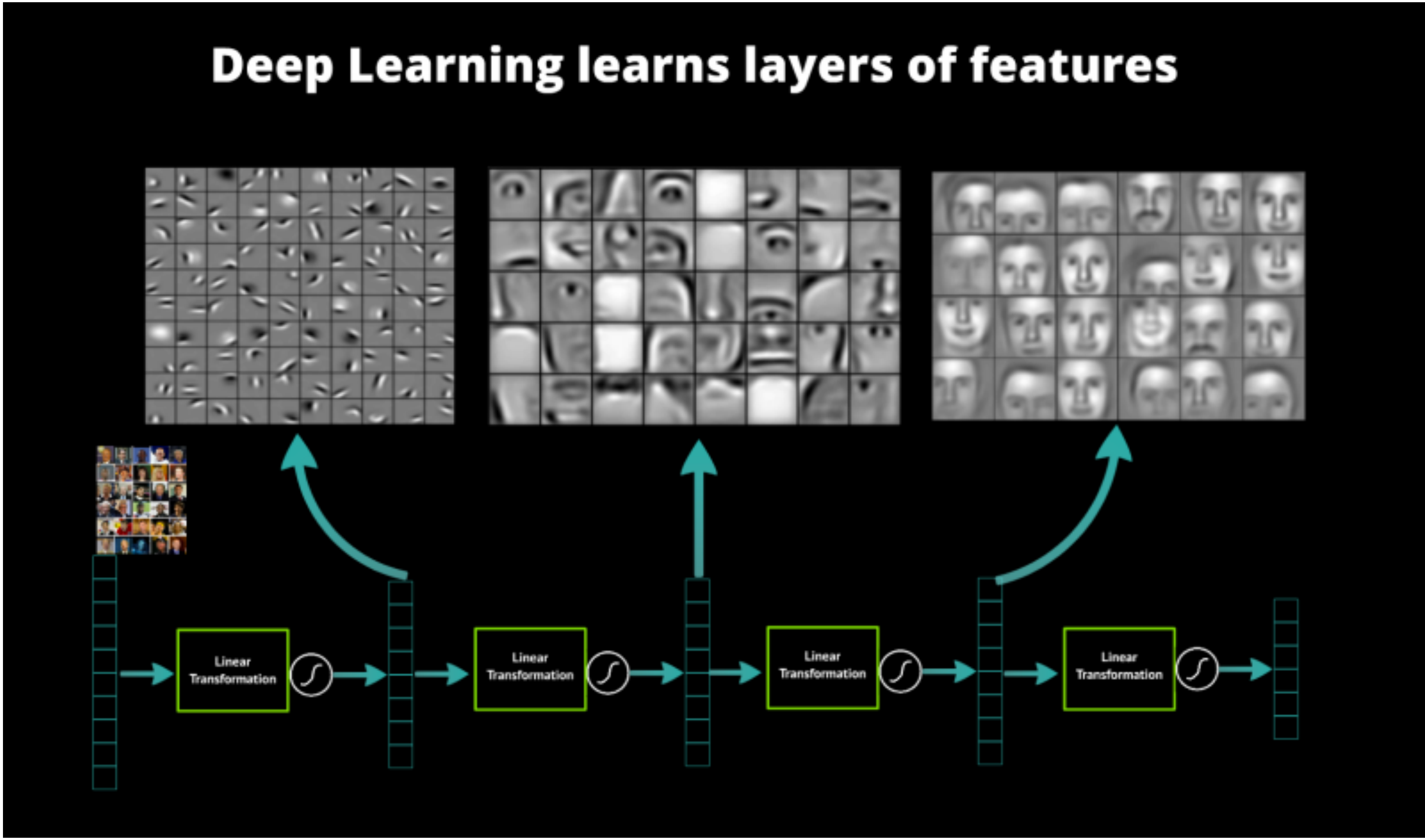
samim23/obama-rnn

obama-rnn - Obama Political Speech generator Recurrent Neural Networks

github.com

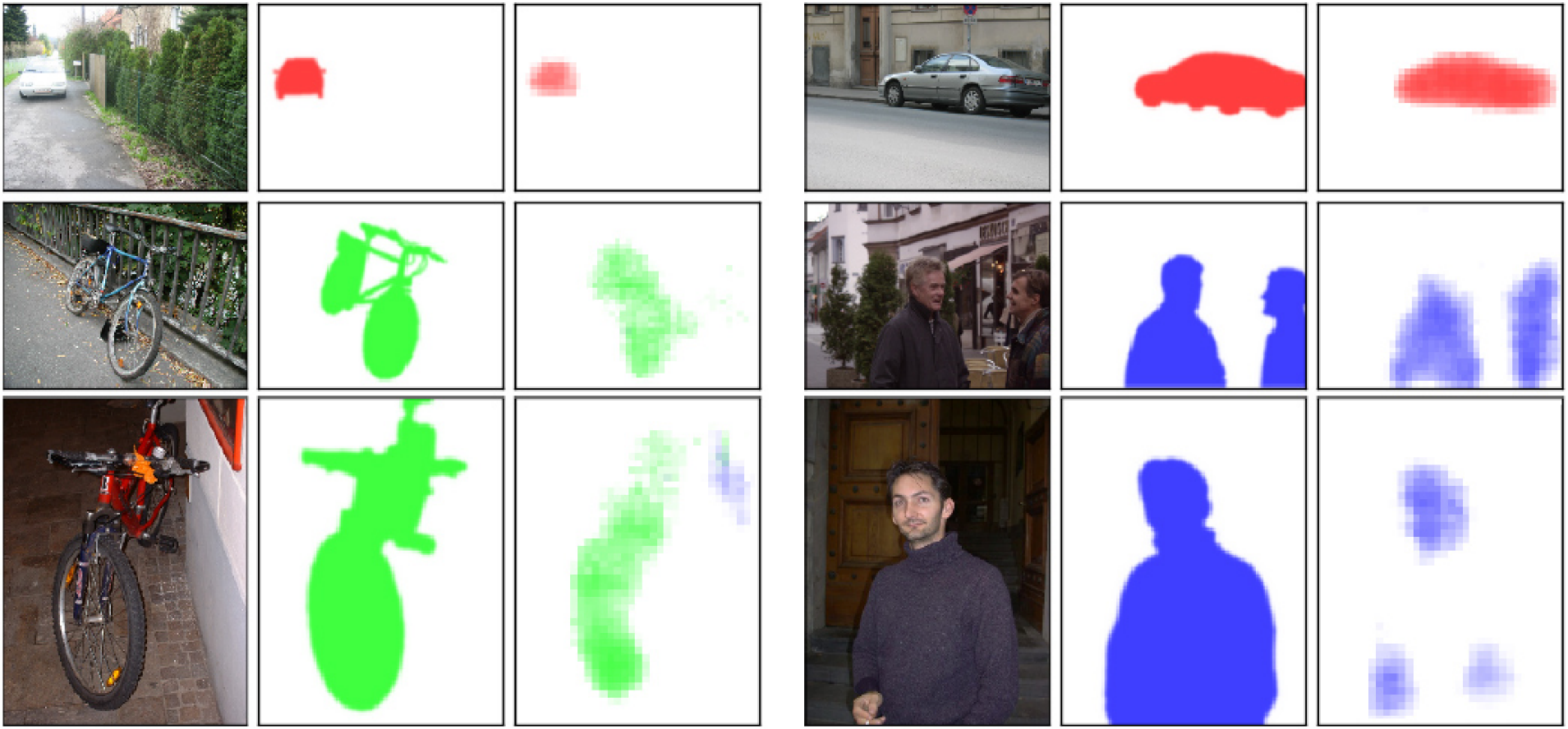
Get in touch here: https://twitter.com/samim | http://samim.io/

Deep Learning



http://www.datarobot.com/blog/a-primer-on-deep-learning/

Deep Learning



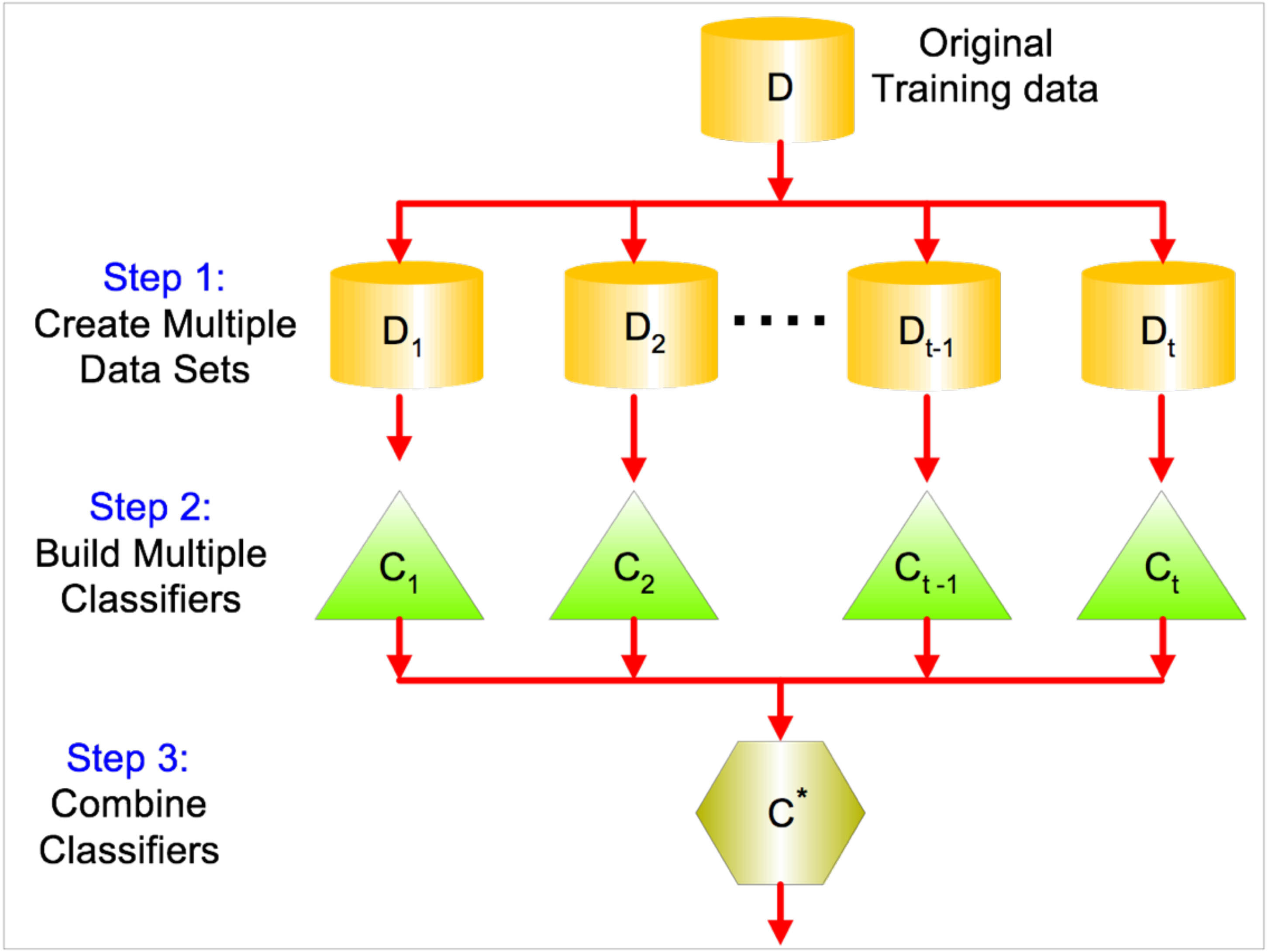
http://www.ais.uni-bonn.de/deep_learning/

Bringing the Machine Learning Posse

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers
- In Olympic Ice-Skating you have multiple judges? Why?

General Idea



Methods for generating Multiple Classifiers

- Manipulate the training data
  - Sample the data differently each time
  - Examples: Bagging and Boosting
- Manipulate the input features
  - Sample the features differently each time
    - Makes especially good sense if there is redundancy
  - Example: Random Forest
- Manipulate the learning algorithm
  - Vary some parameter of the learning algorithm
    - E.g., amount of pruning, ANN network topology, etc.
    - Use different learning algorithms

Background

- Classifier performance can be impacted by:
  - Bias: assumptions made to help with generalization
    - "Simpler is better" is a bias
  - Variance: a learning method will give different results based on small changes (e.g., in training data).
    - When I run experiments and use random sampling with repeated runs, I get different results each time.
  - Noise: measurements may have errors or the class may be inherently probabilistic

How Ensembles Help

- Averaging the results over multiple runs will reduce the variance
  - I observe this when I use 10 runs with random sampling and see that my learning curves are much smoother
- Ensemble methods especially helpful for unstable classifier algorithms
  - Decision trees are unstable since small changes in the training data can greatly impact the structure of the learned decision tree

How Ensembles Help

- If you combine different classifier methods into an ensemble, then you are using methods with different biases
  - You are more likely to use a classifier with a bias that is a good match for the problem
  - You may even be able to identify the best methods and weight them more

Examples of Ensemble Methods

- How to generate an ensemble of classifiers
  - Bagging
  - Boosting
- Can combine classifiers built separately by
  - Simple voting (majority rule)
  - Voting and factoring in the reliability of each classifier (each classifier outputs a confidence score)

Bagging

- Sampling with replacement
- Build classifier on each bootstrap sample
- Each sample has probability (1 – 1/n)n of being selected (about 63% for large n)
    - Some values will be picked more than once
- Combine the resulting classifiers, such as by majority voting
- Greatly reduces the variance when compared to a single base classifier

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

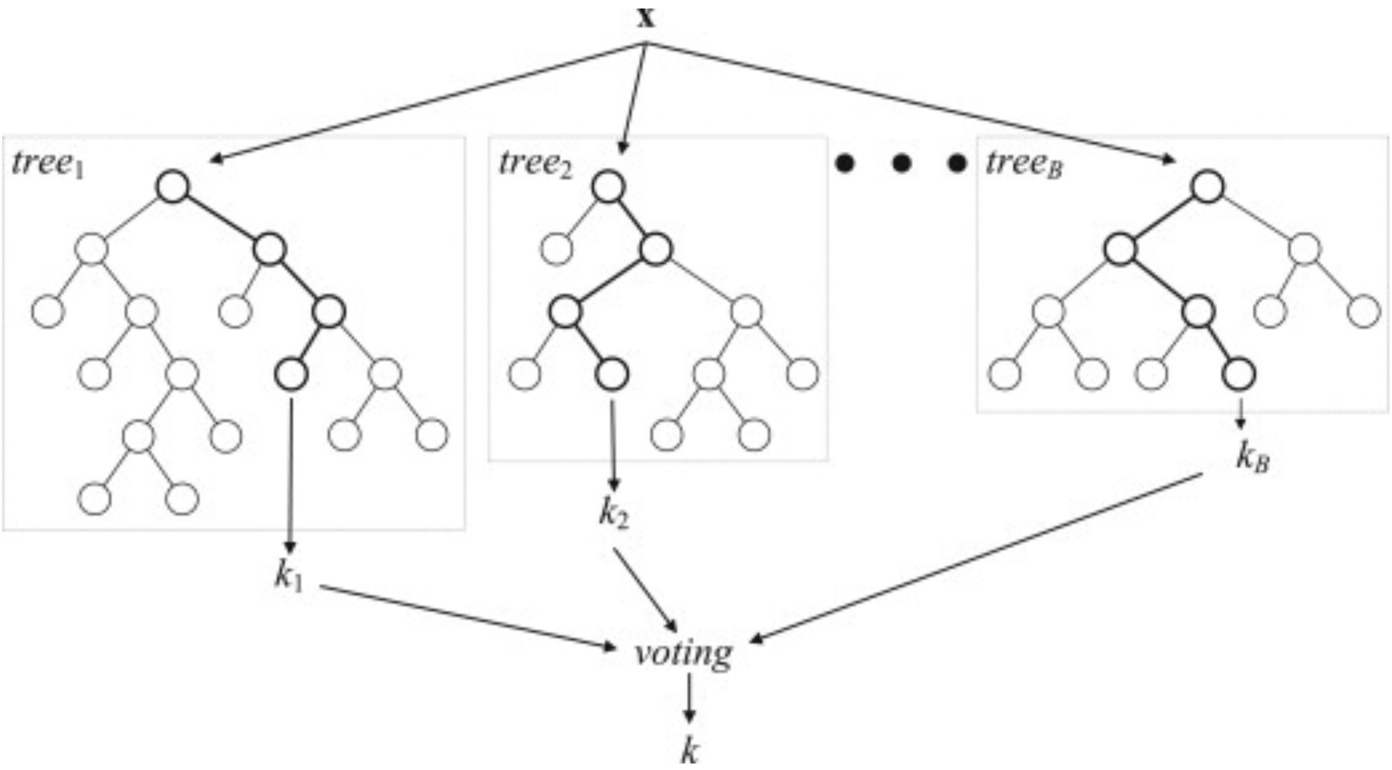
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Example 4 is hard to classify
Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Random Forest

- An ensemble method that creates multiple decision trees
- Training set for each model is created using boosting and bagging from sample data
- Prediction takes into account the output of each model
  - Most frequent (mode) is used for classification
  - Mean of result is used for regression

Random Forest Prediction



http://blogs.sas.com/content/analitika/2012/04/05/az-elvarazsolt-veletlen-erdo/

Advantages of Random Forest

- Corrects tendency of decision trees to overfit
- Reduces model variance and bias
- Requires less tuning
- Works as a general purpose machine learning algorithm that’s reasonably accurate for most tasks
- Using results from all the models, it’s possible to rank feature importance

Disadvantages of Random Forests

- Difficult to explain result (low interpretability)
- The name implies results are arrived at by random

Resources

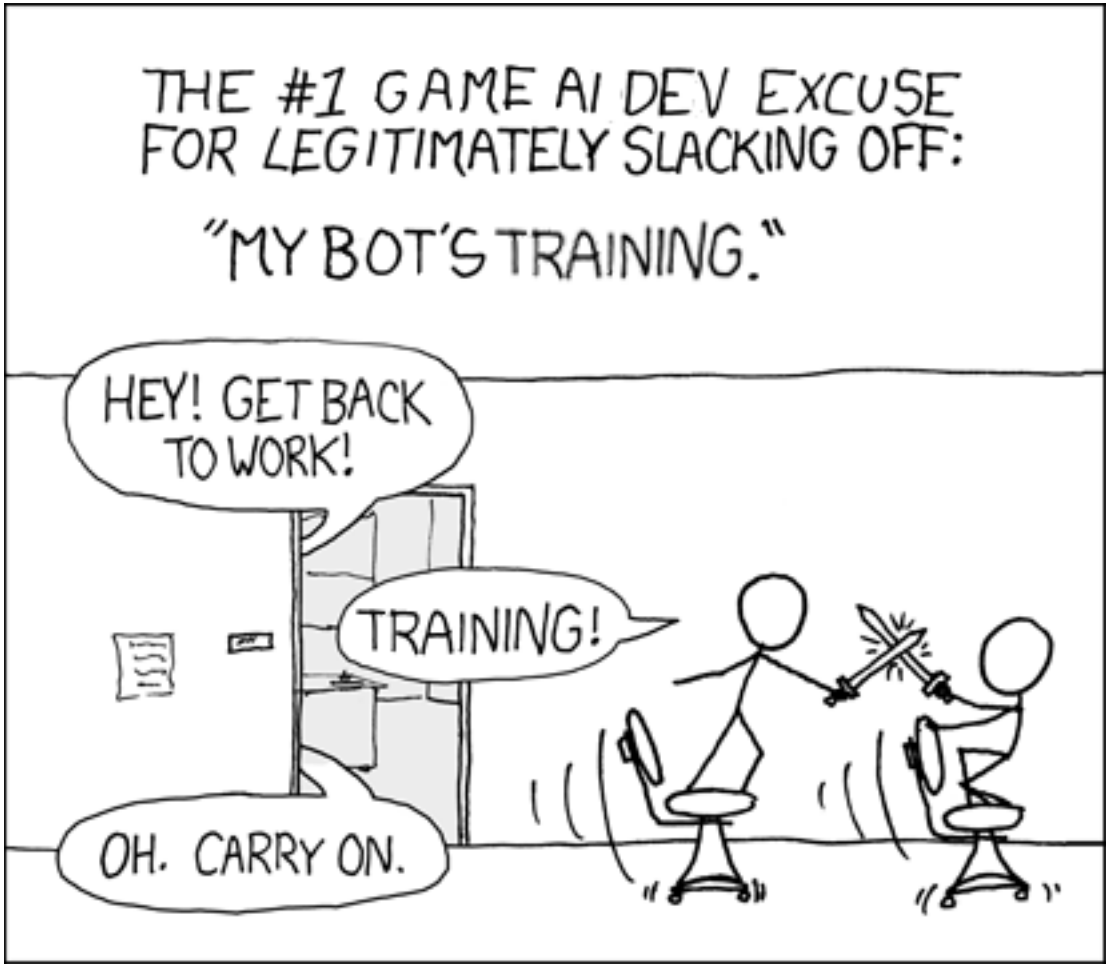
Ensemble methods in Scikit-Learn
http://scikit-learn.org/dev/modules/ensemble.html

yHat Random Forests in Python
http://blog.yhathq.com/posts/random-forests-in-python.html

Association Rule Mining in Python using Orange
http://orange.biolab.si/docs/latest/reference/rst/Orange.associate.html

Association Rule Mining in Python for text
http://magpiehall.com/learning-association-rules-with-python/

WRAP-UP



http://pekalicious.com/blog/training/

Review

- Linear Regression
- Decision Trees
- Logistic Regression
- Naive Bayes
- k-Nearest Neighbor
- k-Means Clustering

Review

- Feature Engineering
  - Tokenization
  - Vectorization
  - TF-IDF