ML Bootcamp Day 1, AM: Intro to ML

Chris Jermaine

Rice University

Plan For The Next Two Days

- This morning: Intro to ML
  - What is ML?
  - History
  - ML success and failure
  - Types of learning
  - Loss and fitting
  - Features and parameters
- This afternoon: Basic Methods
  - Regression
  - SVMs and kernels
  - Decision trees
  - Regularization
  - Clustering and dimensionality reduction

Plan For The Next Two Days

- Tomorrow morning: Deep learning
  - Multi-layer perceptrons
  - VAEs
  - Backprop
  - SGD
  - CNNs and RNNs
- Tomorrow afternoon: Advanced deep learning and RL
  - VAEs
  - Generative adversarial networks
  - Deep RL
  - TensorFlow and PyTorch

Day 1 AM, Chapter 1: A History of ML

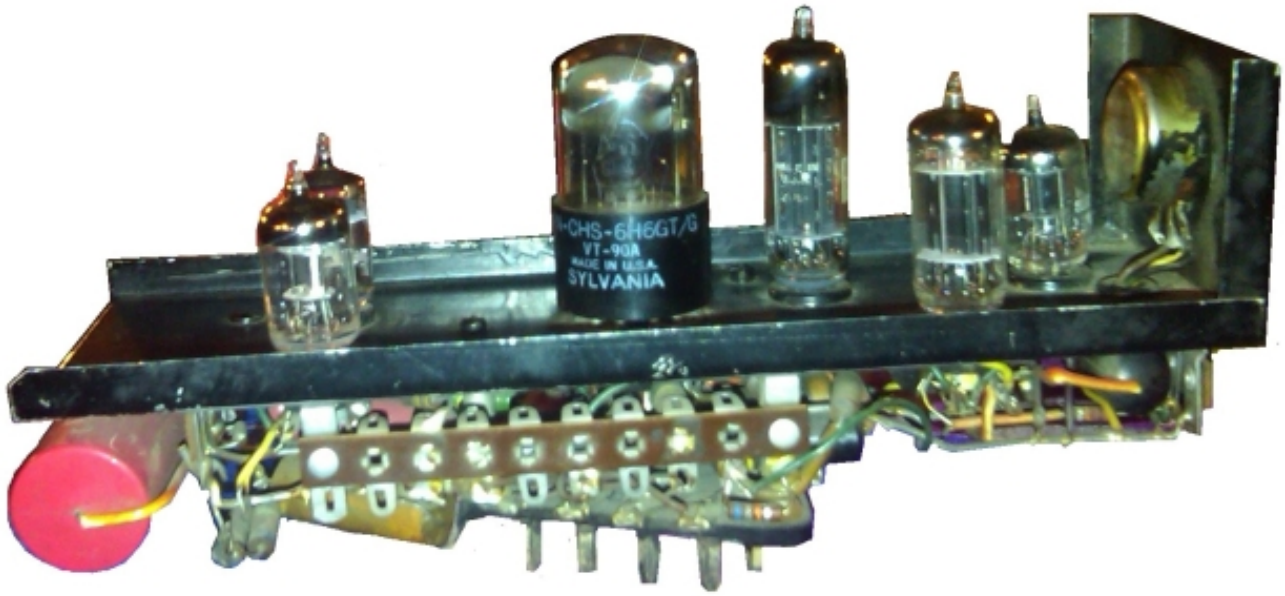
First, What is ML?

- A subset of "AI" (Artificial Intelligence)
  - A more precise definition in a minute!
- AI is the task of programming an "intelligent agent"
  - Able to solve novel problems
  - Has emergent behavior
  - Can do things not specifically programmed to do

AI: Early-Early Years

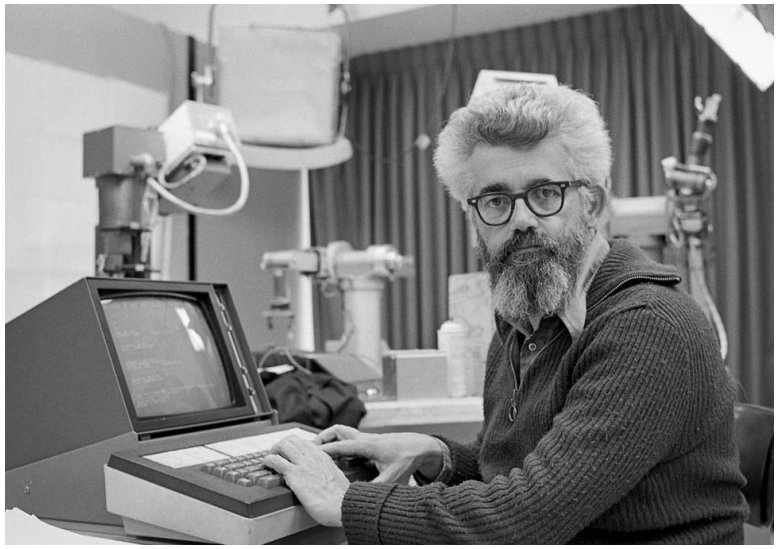
- AI goes back a long time, to Leibniz, Hobbes, Descartes
- Leibniz "Calculus ratiocinator"
  - 17th century
  - Precursor to mathematical logic
  - Imagines an inference engine able to reason

AI: Early-Early Years



- Artificial Neural Networks
  - First computational model proposed in 1943
  - Inspired by the brain
  - Warren McCulloch and Walter Pitts: ANs can perform logic
  - Contemporary with ENIAC (UPenn, 1946)
  - Minsky built SNARC (Stochastic Neural Analog Reinforcement Calculator) in 1951
  - Early ANNs: compute model, not AI

AI: Early Years

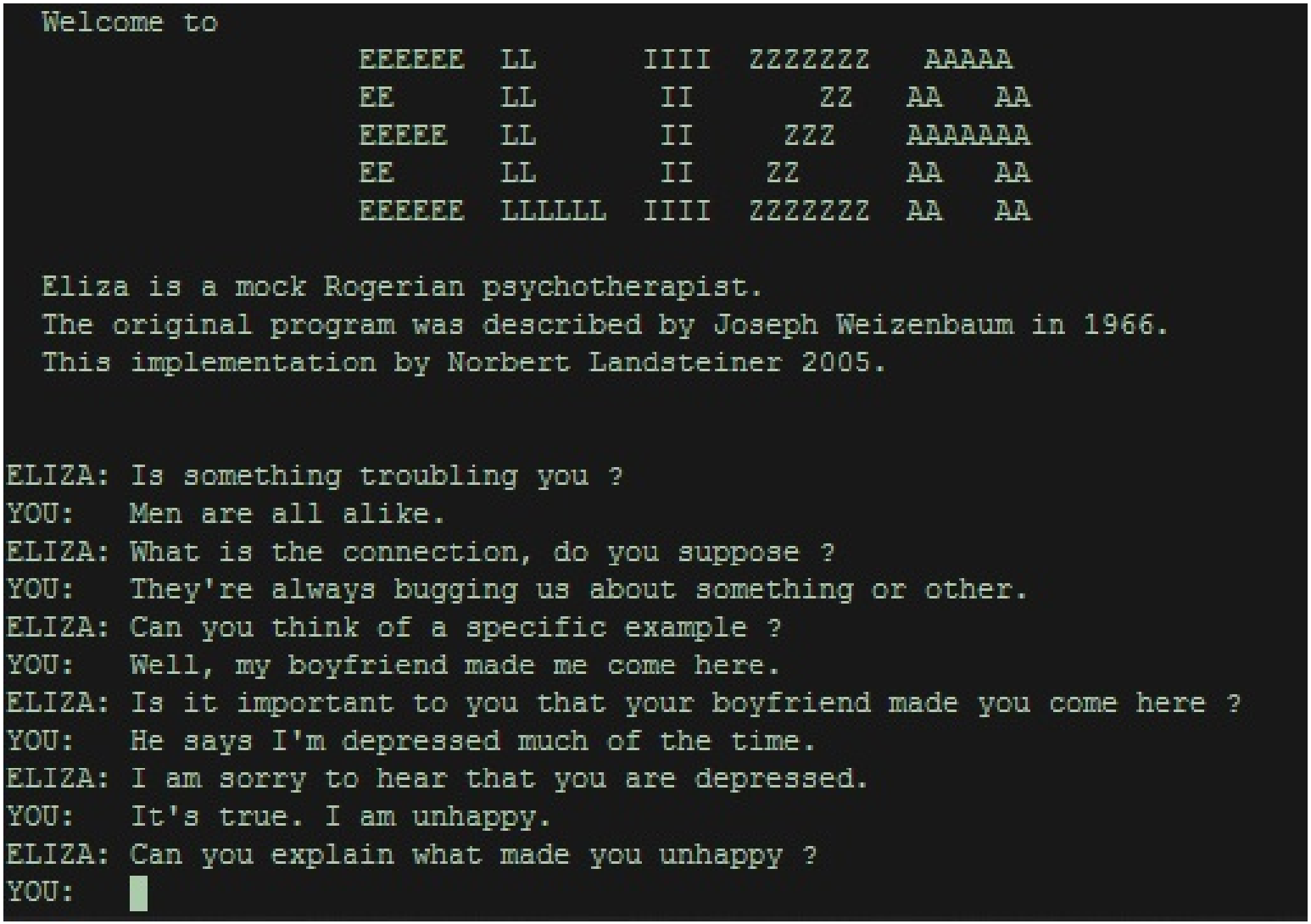


- AI: term invented by John McCarthy in 1956
  - John McCarthy: Stanford computer scientist who invented Lisp
  - Turing award winner in 1971
  - "Dartmouth Summer Research Project on Artificial Intelligence"
  - Summer-long brainstorming session

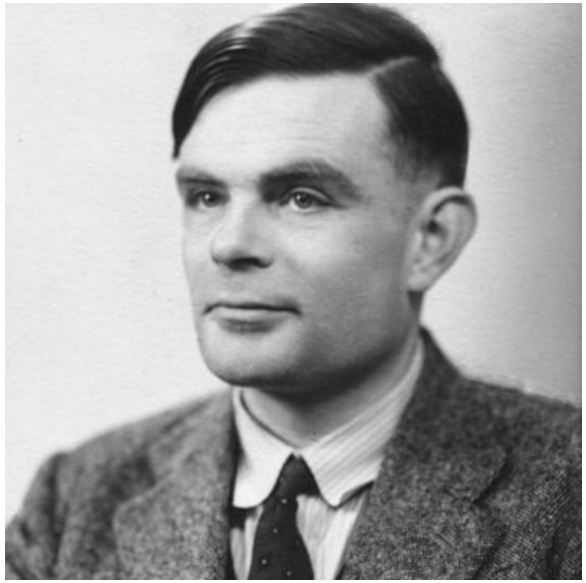
AI: Early Years

- Early successes came quickly
  - AI as search: goals, subgoals, backtracking
  - Newell and Simon "General Problem Solver"
  - Arthur Lee Samuel: Samuel Checkers-playing Program (at IBM, around 1960)
  - Eliza: 1964-1966 MIT AI Lab

AI: Early Years



The Turing Test



- Eliza: first serious attempt to pass the "Turing Test"
  - Based on Alan Turing's 1950 paper "Computing Machinery and Intelligence"
  - Turing Test: three "players"
  - (1) Interrogator, (2) Computer, (3) Human
  - Interrogator passes written questions to both
  - Tries to determine which one is human
- Eliza was astonishing to people in the 1960's
  - Not close to passing the Turing Test!

The Hype

- Herbert Simon in 1965:
  - "Machines will be capable, within twenty years, of doing any work a man can do."
- Marvin Minsky in 1967:
  - "Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."
- Marvin Minsky in 1970:
  - "In from three to eight years we will have a machine with the general intelligence of an average human being."

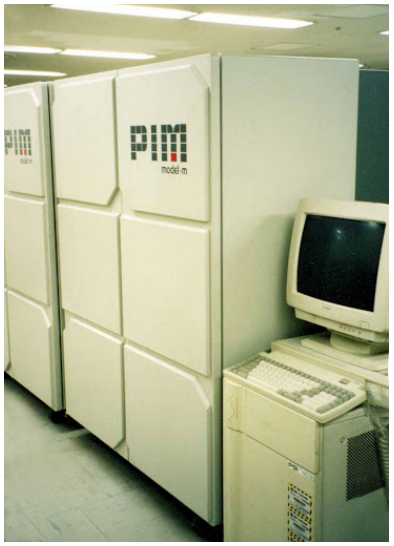
But All Was Not Well

Winter Is Coming

Some ominous signs...

- Automatic Language Processing Advisory Committee, 1964-66
  - Seven scientists, established by US government
  - Looked to study the state of computational linguistics
  - Critical of machine translation work to date
- Minsky and Papert: 1969 book "Perceptrons"
  - Showed some limitations of common neural networks
  - Seemed to kill off neural networks research for a long time
- "Artificial Intelligence: A General Survey" by James Lighthill, 1973
  - "Formed the basis for the decision by the British government to end support for AI research in all but two universities"

Winter Is Here



- Speech Understanding Research program (DARPA) at CMU, other locations
  - Harpy: recognized more than 1000 words
  - But had severe limitations
  - Led to massive cutback in DARPA AI funding by 1974

- 1990–Death of the Fifth-Generation project
  - In the 80's, Japanese invested $400 million (1990 dollars)
  - Not much to show for it

The Rise of ML

- AI traditionally relied on smart programmers
  - A program may be able to "reason"
  - Could search for a better solution (ex: minimax)
  - Could chain together rules

$$A \to B, B \to C \implies A \to C$$

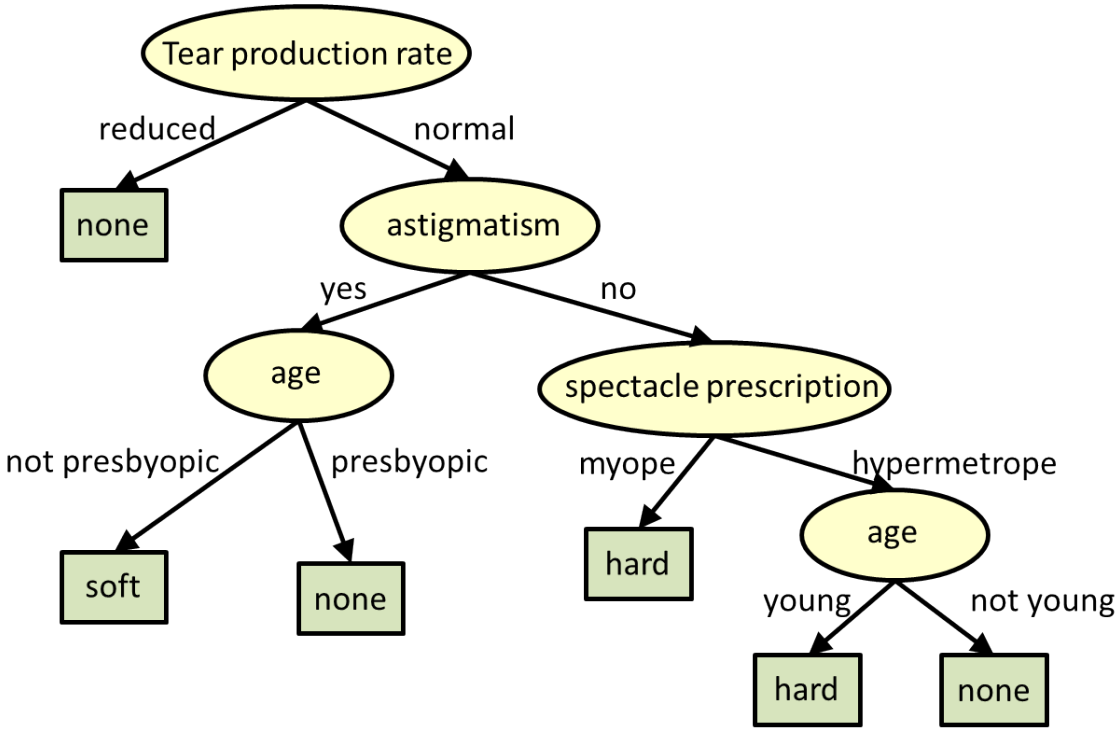
  - But didn't really learn

- That ended with the advent of ML
  - AI: an intelligent agent; has emergent behavior
  - ML: an AI that learns from retrospective data or experience
  - ML is often statistical in nature

ML: Important Developments

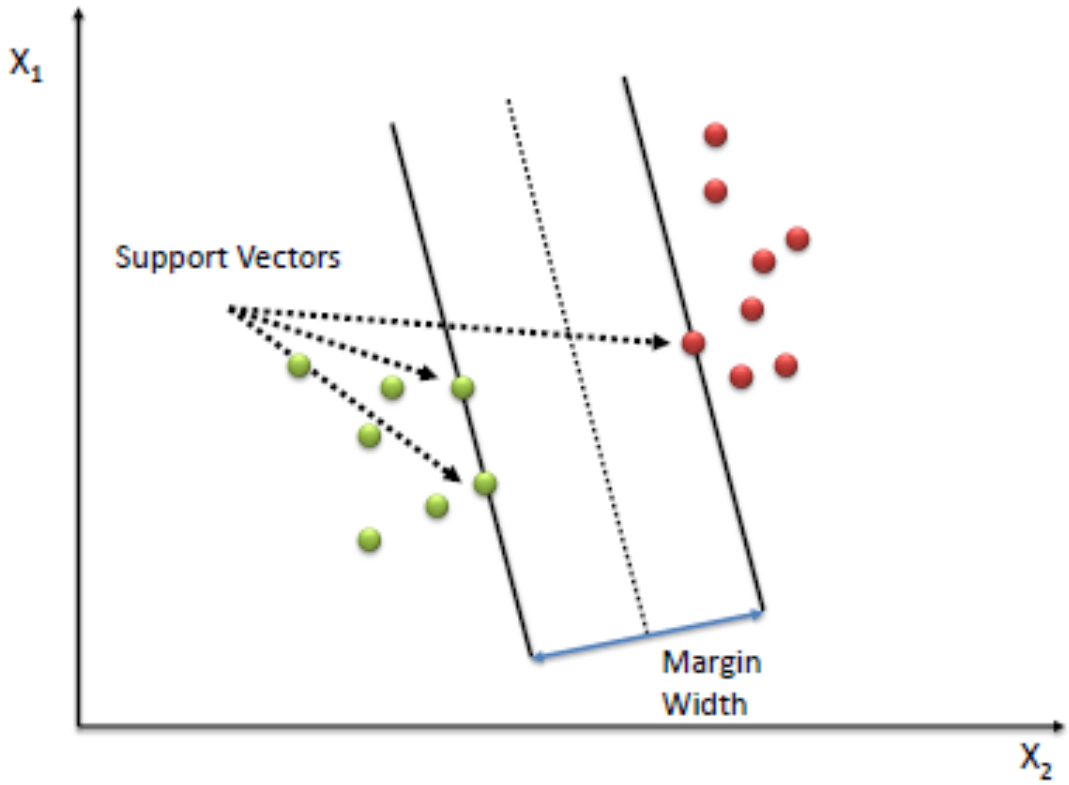
- 1986: Backpropagation
  - Rumelhart, Hinton, Williams
  - Use BP to train a NN with hidden layer(s) via gradient descent
  - Forerunner of all modern "deep learning"
- 1989: Reinforcement Learning (Q-learning)
  - Developed by Watkins
  - Paradigm where ML learns via experience
  - Not pre-collected data
- 1993: Association rule mining
  - 1993 paper of Agrawal et al., IBM
  - Example: $\{\texttt{bread}, \texttt{butter}\} \implies \{\texttt{milk}\}$
  - Kicked off field of "data mining"

ML: Important Developments



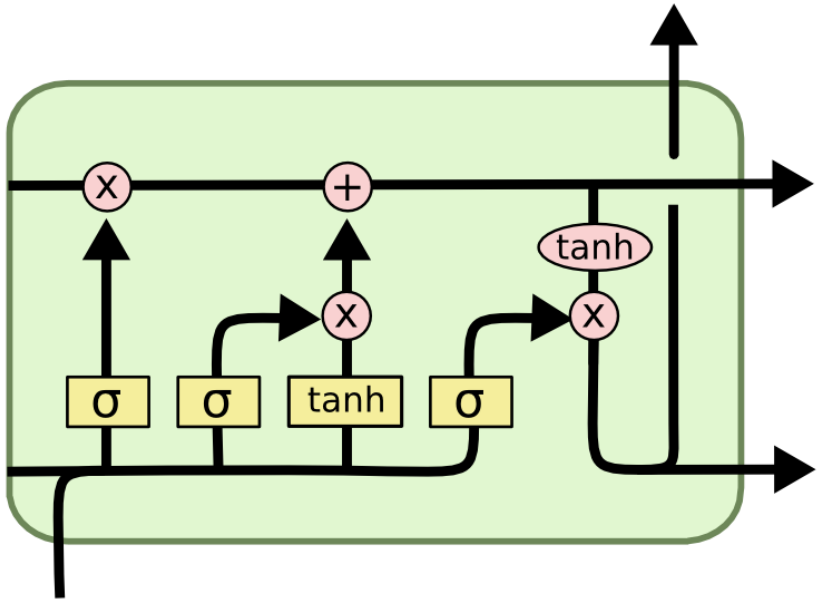
- 1995: Random Forests
  - Ensemble decision tree method
  - Still among the most accurate classifiers

ML: Important Developments



- 1995: Invention of Support Vector Machines
  - Cortes and Vapnik
  - "Max margin" method
  - Works well for large number features, little data

ML: Important Developments

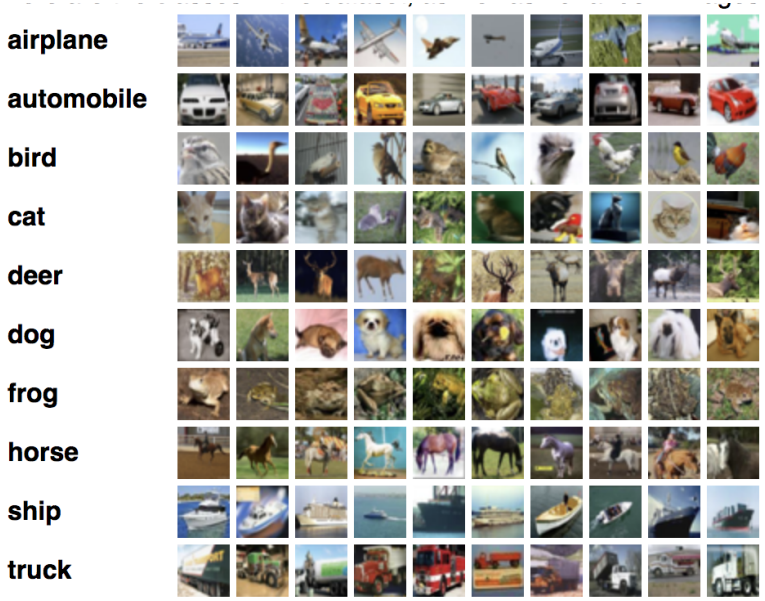


- 1997: Invention of LSTM
  - Type of neural network
  - Can "remember" state over sequential data
  - Avoids "vanishing gradients" problem

ML: Important Developments

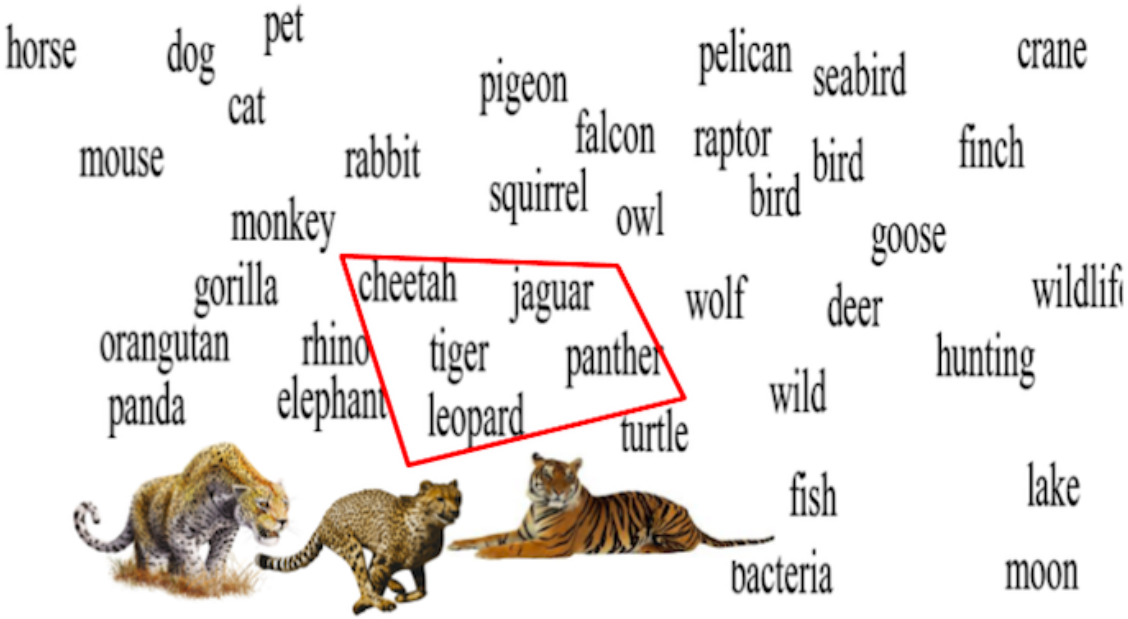
- 2006: Netflix Prize
  - Spurred research on recommender systems
  - Netflix provided 100M movie ratings
  - Took form of `<user, movie, date of grade, grade>`
  - Learn to predict `grade` on new data
  - Beat Netflix algorithm by 10%? Win $1M!
  - Prize collected in 2009

ML: Important Developments



- 2009: ImageNet
  - Fei-Fei Li from Stanford University
  - Data set that spurred vision research
  - 14 million images, 20,000+ categories
  - Best deep NNs are now better than people!

ML: Important Developments



- 2013: Word2Vec
  - Tomas Mikolov led a Google team
  - Reads huge amount of text, learns word embedding
  - Makes it easy to use other ML methods on text data

Where Are We Today?

- There have been some amazing recent developments
- Lots of recent buzz: GPT-3
  - 125-billion parameter language model
  - Developed by OpenAI
  - GPT = "Generative Pre-trained Transformer"
  - Transformers are new class of big NN
  - Learned on most of English available electronically
  - 6 million Wikipedia docs: is less than 1% of training data!
- Reported results are amazing
  - Compare with Eliza!

GPT-3 Example

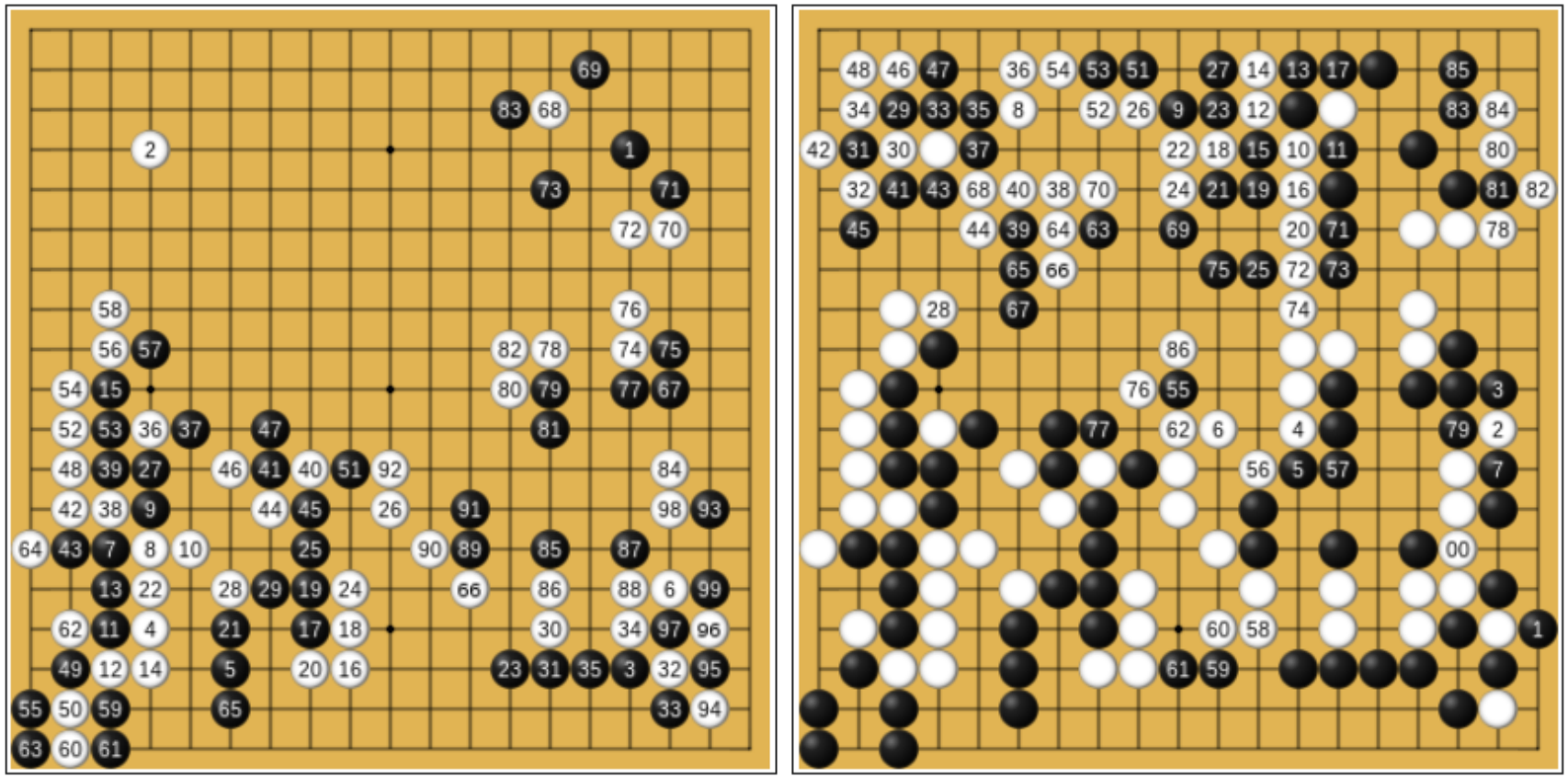
Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Not Just Image Classification: Image Generation

- GANS: Generative Adversarial Networks
  - Ian Goodfellow and his colleagues in 2014
  - Two neural networks learn together:
  - One to generate data
  - One to recognize real from fake data
  - Generator gets so good it can fool humans
  - (This leads to "deep fakes", unfortunately)

Example: GANS

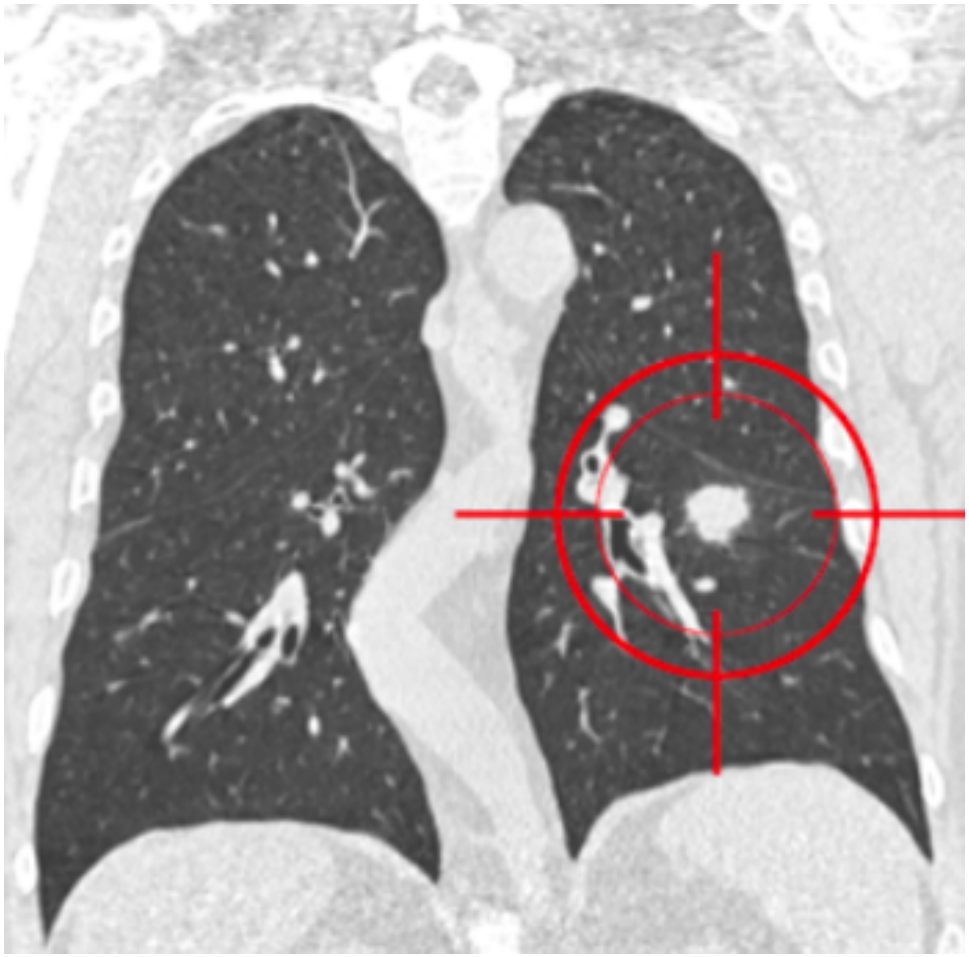
Game Playing



Few Games Remain Where Humans Can Beat Machines

- Most modern ML game playing relies on reinforcement learning
  - ML learns to solve a "Markov Decision Process"
  - A game where the ML learns to traverse a graph
  - Actions lead to stochastic rewards/transitions
  - "Deep RL" allows a deep NN to learn to maximize reward
- RL-based ML algorithms have done amazing things
  - AlphaGo: Go-playing ML algorithm
  - March 2016: beat Lee Sodol (9-dan pro)
  - 2017: beat Ke Kie, world number 1
  - AlphaZero: trained only via self-play (no data)
  - Beat AlphaGo 100-0

Image Classification



- Not just useful for toy database (ImageNet)
  - ML algorithms can out-perform human experts at important tasks

Reading Low-Dose CT Scans

- "End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography"
  - Trained deep learning model on 42,290 CTs from 14,851 patients
  - 578 developed biopsy-confirmed cancer in one year
  - Tested on 6,716 cases (86 cancer positives)
  - Deep learning model: ROC 94.4%
  - Six radiologists: all performed worse

What Is ML Bad At? Causation

- Key weakness: ML algorithms do correlation not causation
  - Is grass necessary to identify the presence of a lion?
  - Even a child would discount the grass...
  - But ML algorithm may classify a lawn as a lion!
- One way to avoid this: learn in the "real-world" (ex: RL)
  - But it can be tough to facilitate this

What Is ML Bad At? Fairness

- ML algorithms are only as good as their data
  - We want to use ML to help with hiring decisions
  - Our company has always done a good job promoting men
  - Most examples of high performers are men
  - Gender is specifically excluded as a criteria
  - ML algorithm can spot proxies for gender
  - May still be biased towards hiring men

Day 1 AM, Chapter 2: Supervised Learning

ML: How Does It Work?

Foundation of modern ML: supervised learning

- One of the most fundamental problems in data science
  - Given a bunch of (x, y) pairs
  - Goal: learn how to predict value of y from x
  - Classically, x is a feature vector
  - Example: x is $\langle \texttt{age}, \texttt{income}, \texttt{gender} \rangle$
  - Called "supervised" because have examples of correct labeling

Supervised Learning Examples

Any task where we want to mimic existing labeling:

- Given an image, tell if has a cat or not
- Given a text EMR, label "breast cancer" or not
- Given a document (email) in a court case, figure which subjects relevant to
- Given information about a patient surgery, predict death
- Given head trauma patient info, predict ICP crisis
- Given an set of surgical vital signs, label "good surgery" or not
- Many others!

Two Most Common Types of SL

- Classification and regression
- Classification:
  - Outcome to predict is in $\{+1, -1\}$ ("yes" or "no")
  - Ex: Given a text EMR, label "breast cancer" or not
- Regression:
  - Outcome to predict is a real number
  - Ex: Given an ad, predict number of clickthrus per hour

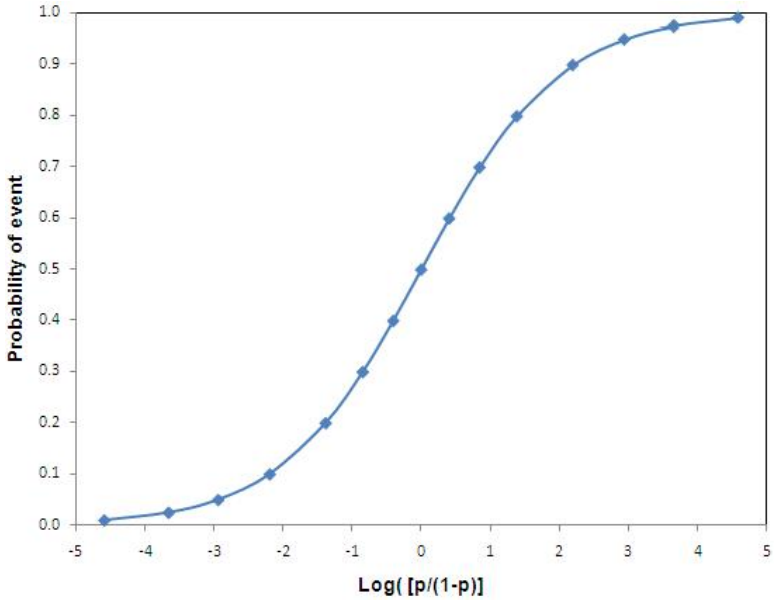
Workhorse of Modern SL

- Linear regression
  - From x, predict y as:

$$f(x|r) = \sum_j x_j r_j$$

  - $\langle r_1, r_2, \ldots, r_m \rangle$ are called regression coefficients
  - Note: this gives a real-valued output
- This is just one way to do SL
- You'll see many others:
  - kNN, support vector machines, random forests, etc.

Extending LR to Classification



- Output of linear regression input into a logistic function
  - From x, predict y as:

$$f(x|r) = (1 + e^{-\sum_j x_j r_j})^{-1}$$

  - Maps feature vector to a probability of true

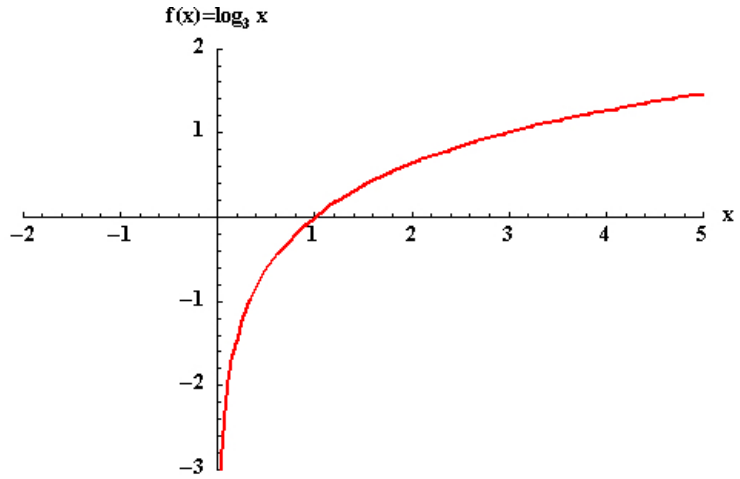
Most Modern SL is Optimization-Based

- Deep learning in particular follows this paradigm...
- The model $f(x|\Theta)$ we are trying to "learn" has a parameter set Θ
  - Θ is list of regression coefs r_1, r_2, r_3, etc. in LR
- Given a data set, we definite a loss function
- Simplest example: loss is "squared error"
  - Data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$

$$L(\Theta) = \sum_i \left(f(x_i|\Theta) - y_i\right)^2$$

- Goal: choose Θ so as to minimize the value of the loss function

Classical Loss for Classification: Cross Entropy



- Assume $f(x, y|\Theta)$ outputs probability of y, given feature vector x
  - Then cross-entropy loss at data point (x, y) defined as:

$$L(\Theta) = -\log f(x, y|\Theta)$$

  - For a data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$:

$$L(\Theta) = -\sum_i \log f(x_i, y_i|\Theta)$$

- Intuitively makes sense
  - You give prob of 1 to the "real" outcome, loss is zero
  - You give prob of 0 to the "real" outcome, loss is ∞

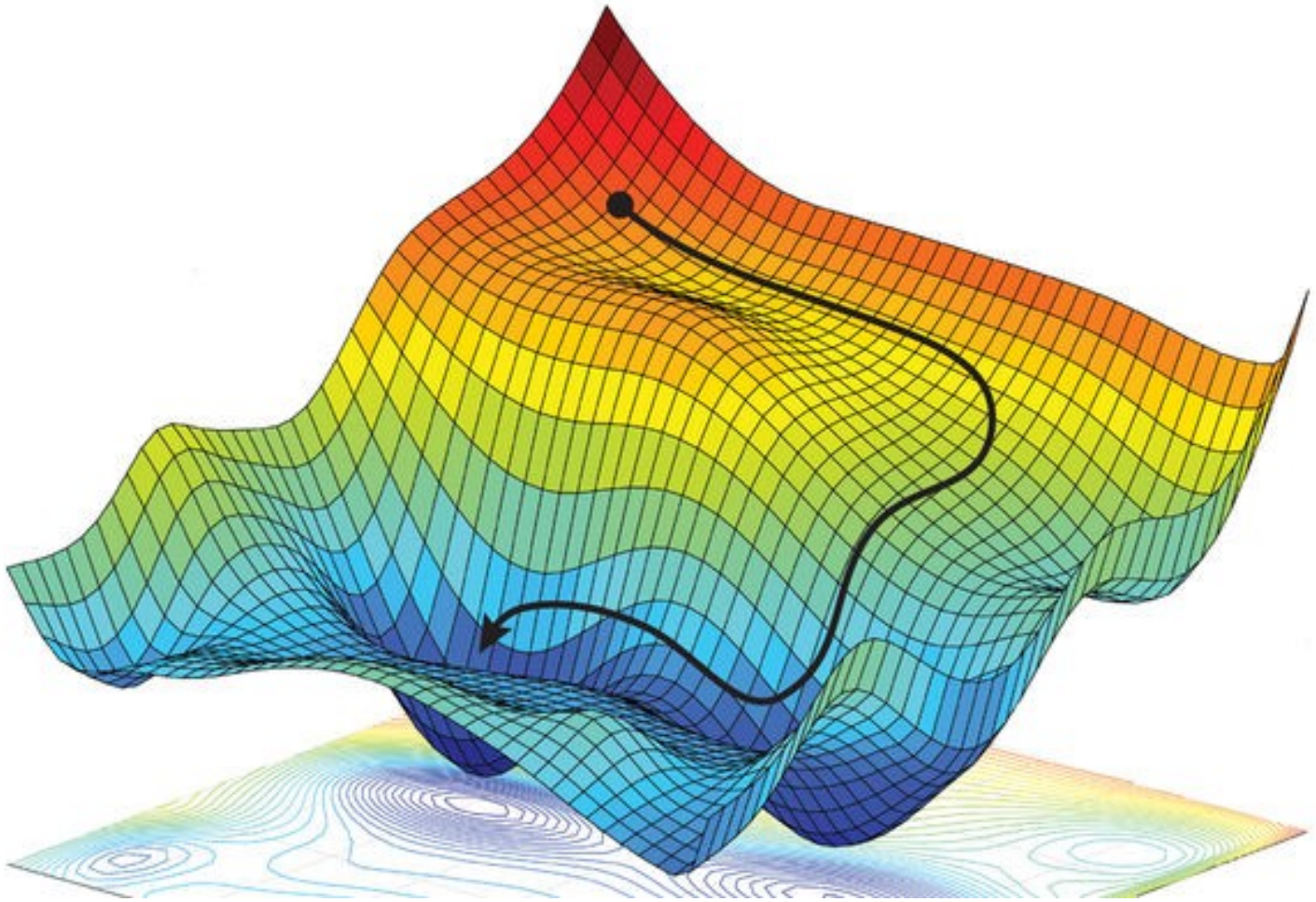
How to Minimize the Loss?

- What are the desired properties?
- To be useful for data science, opt framework should be
  - Easily applied to many types of opt problems
  - Scalable (easily built in MapReduce, for example)
  - Fast (quick convergence)

Most Widely Used Opt Framework Is...

- For deep learning in particular...
  - Gradient descent!
- What's the idea?
  - GD is an iterative algorithm
  - Goal: choose Θ^* to min $L(\Theta)$
  - Tries to incrementally improve current solution
  - At step i, Θ_i is current guess for Θ^*

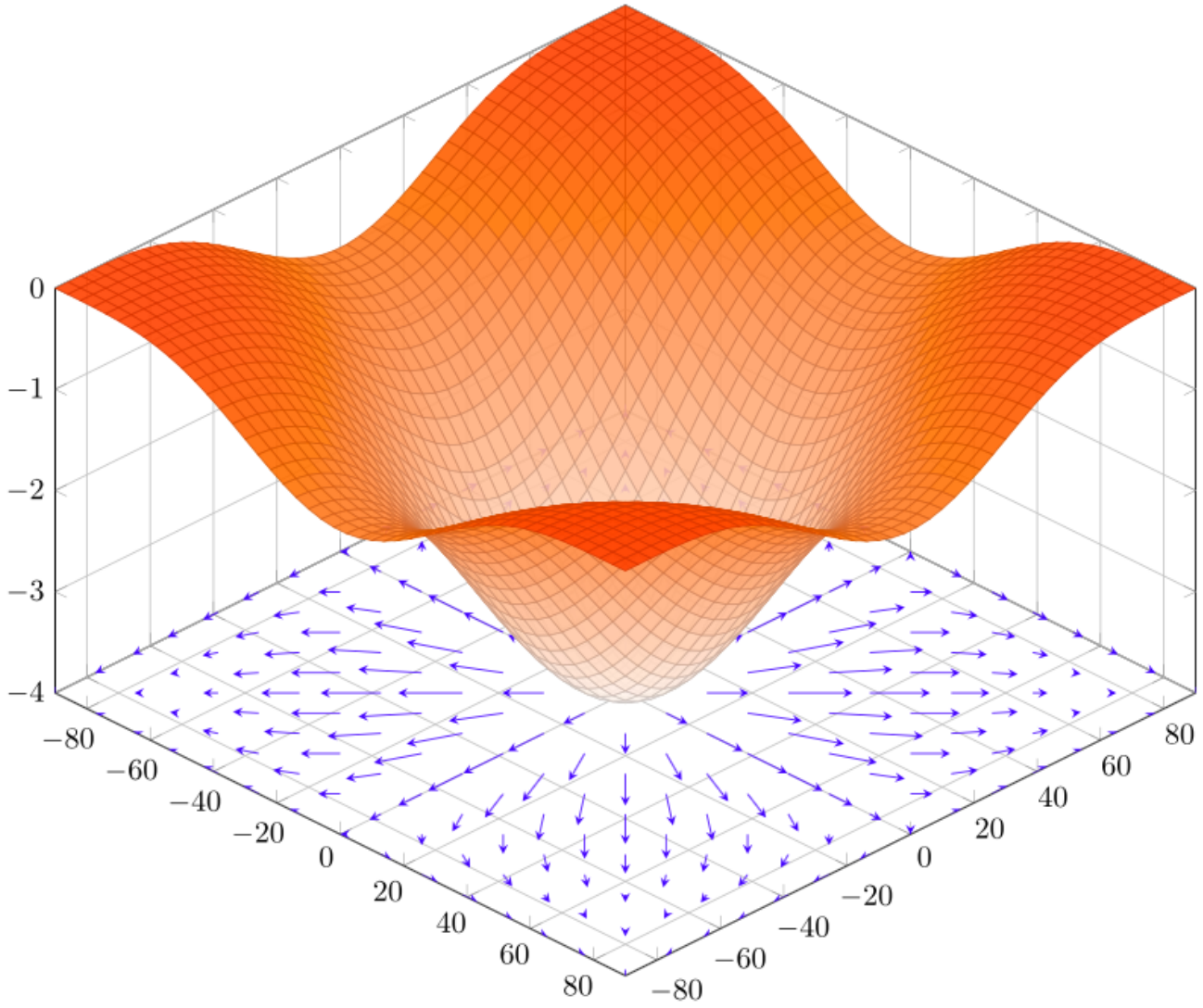
AKA: Method of Steepest Descent



What's a Gradient?

- Gradient is the multi-dimensional analog to a derivative
  - If $L(.)$ accepts a vector
  - ∇L is a vector-valued function
  - That is, accepts a vector Θ
  - Returns a vector...
  - whose ith entry is ith partial derivative evaluated at Θ
  - Points in direction of steepest ascent

Ex: Gradient of a 2-D Function



Gradient Descent

Basic algorithm:

$\Theta_1 \leftarrow$ non-stupid guess for Θ^*;
$i \leftarrow 1$;
repeat {
  $\Theta_{i+1} \leftarrow \Theta_i - \lambda \nabla L(\Theta_i)$;
  $i \leftarrow i + 1$;
} while $(|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon)$

Gradient Descent

Basic algorithm:

$\Theta_1 \leftarrow$ non-stupid guess for Θ^*;
$i \leftarrow 1$;
repeat {
  $\Theta_{i+1} \leftarrow \Theta_i - \lambda \nabla L(\Theta_i)$;
  $i \leftarrow i + 1$;
} while $(|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon)$

- Here λ is the "learning rate"
  - Controls speed of convergence
- And $\nabla L(\Theta_i)$ is the gradient of L evaled at Θ_i

Stopping Condition

- Here we use

$$\texttt{while } (|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon)$$

- We keep going until the loss stops improving
  - ▷ That is, until we have "converged"
  - ▷ $|L(\Theta_i) - L(\Theta_{i-1})|$ is the difference in the loss across last two iterations

- Drawback: requires loss computation... can be expensive

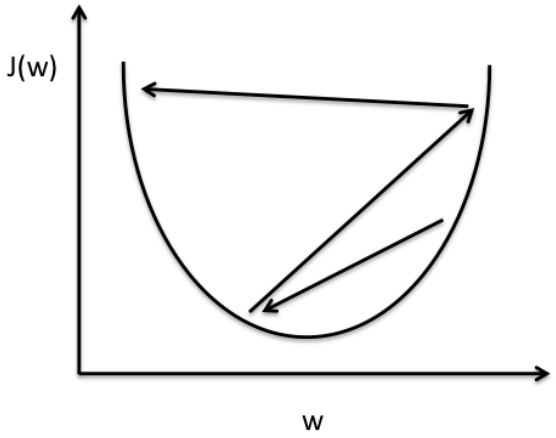
The Learning Rate

Reconsider the algorithm:

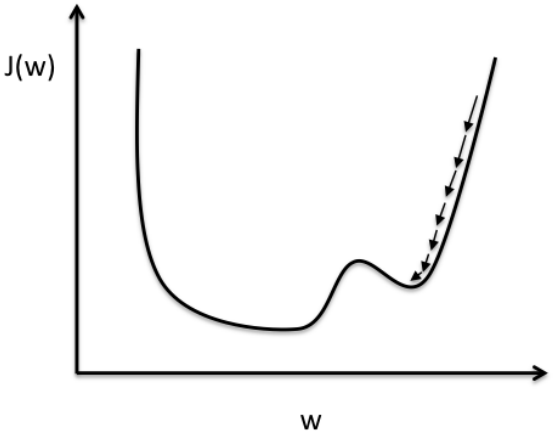
$\Theta_1 \leftarrow$ non-stupid guess for Θ^*;
$i \leftarrow 1$;
repeat {
  $\Theta_{i+1} \leftarrow \Theta_i - \lambda \nabla L(\Theta_i)$;
  $i \leftarrow i + 1$;
} while ($|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon$)

- How to choose λ?
  - Multiplier on the gradient $\nabla L(\Theta_i)$
  - So controls the distance traveled at each step

Effect of Learning Rate



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

- Choice of learning rate super important
  - Too small: many, many passes thru the data to converge
  - Too large: oscillate into oblivion

Line Search

Best option (in terms of results) but most expensive:

- Solve another mini-optimization problem at each iteration
- That is, choose λ so as to minimize $L(\Theta_{i+1})$
- At lest now, it's a 1-dimensional opt problem!
- Called a "line search"

Line Search

- Sort of like a binary search
- But try to find a min, not a specific value
  - Always have two bounds l and h on λ
  - At each iteration, choose two l', h' within $[l, h]$
  - Breaks line segment between l and h three ways (two ends and a middle)
  - Evaluate loss at l', h'
  - Cut off the worse of the two ends

Line Search

$l \leftarrow 0$;
$h \leftarrow 999999$;
while $(h - l > \epsilon)$ do {
  $h' \leftarrow l + \frac{1}{c}(h - l)$;
  $l' \leftarrow h - \frac{1}{c}(h - l)$;
  $\text{loss}_h \leftarrow L(\Theta_i - h' \nabla L(\Theta_i))$;
  $\text{loss}_l \leftarrow L(\Theta_i - l' \nabla L(\Theta_i))$;
  if (loss_h < loss_l) {
    $l \leftarrow l'$;
  else
    $h \leftarrow h'$;
  }
}

Day 1 AM, Chapter 3: Measuring Accuracy

Measuring Classification Accuracy

- OK, so now we've built a model
- How do we know if it's good? We need a metric
- Simplest: % correct
  - ▷ Pros and cons?

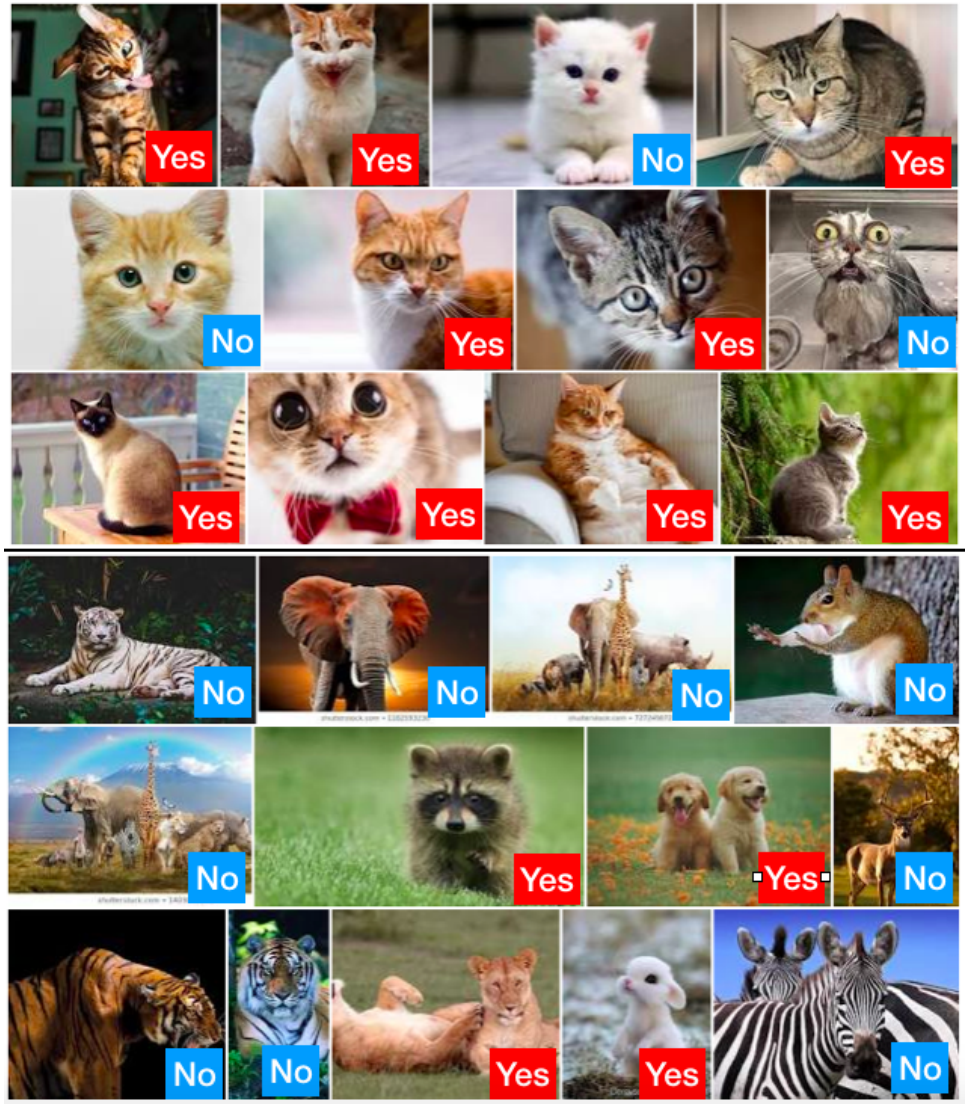
Measuring Classification Accuracy

- Simplest: % correct
  - Pros and cons?
  - Pro: single number
  - Pro: easy to understand
  - Con: Terrible with unbalanced classes (99% are "no"? Get 99% accuracy: say "no" all of the time)

Measuring Classification Accuracy

- Can do better
- False positive and false negative rates
  - More common in ML
  - False negative: (num we say are false that are actually true / num that are true)
  - False positive: (num we say are true that are actually false / num that are false)
- Almost equivalent: Recall and precision
  - More common in information ret.
  - Recall: (num we say are true that are actually true / num that are true)
  - Precision: (num we say are true that are actually true / num we say are true)
- Pro: nice with imbalanced classes
- Con: single number important to order models from best to worst

Example: Classifying House Cats



False Positive and False Negative

- In our example:
  - False negative: (num we say are false that are actually true / num that are true) $= 3/12$
  - False positive: (num we say are true that are actually false / num that are false) $= 4/13$

Recall and Precision

- In our example:
  - Recall or true positive rate: (num we say are true that are actually true / num that are true) $= 9/12$
  - Precision: (num we say are true that are actually true / num we say are true) $= 9/13$

Measuring Classification Accuracy

- Common metric: F_1 (say, "Eff-One")
  - Puts recall and precision into single number

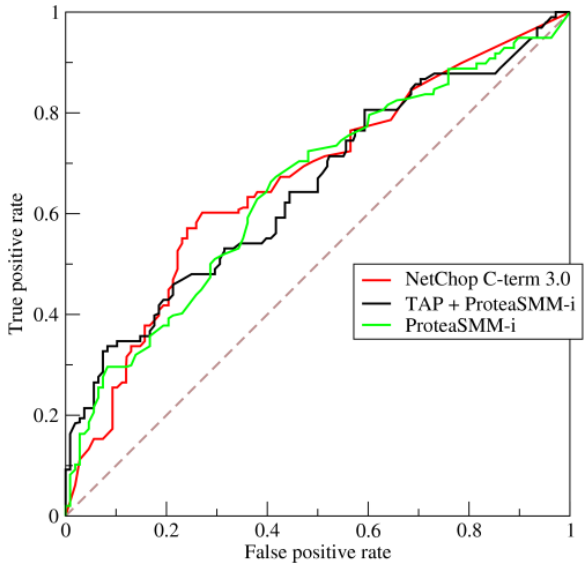
$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - Pros and cons?
  - Pro: single number, more informative than accuracy
  - Con: Doesn't consider false positive/recall trade-off

ROC Plot

- Note: can usually increase recall by changing internal param in learned clarifier
  - Ex: Simple linear classifier: if $\sum_j x_{i,j} r_j > c$, say "yes"
  - Increase c: fewer false positives, lower recall
- ROC = "Receiver operating characteristic"
  - Measures effect of increasing recall on false positive rate

ROC Plot

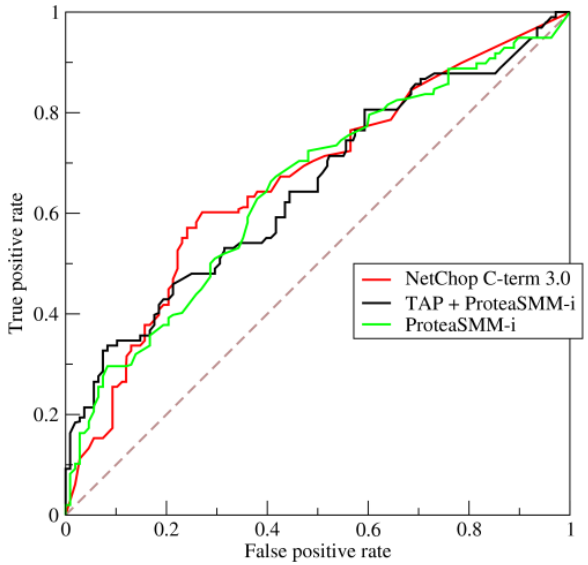


- ROC plots false positive
  - num we say are true that are actually false / num that are false
- Vs true positive rate or recall
  - num we say are true that are actually true / num that are true
- Random classifier will be on diagonal line

Why Random Classifier on Diagonal?

- If we randomly choose 20%, in reality 60% are true...
  - FP: num we say are true that are actually false / num that are false
  - FP is $(20\% \times 40\% \times n)/(40\% \times n) = 20\%$
  - TP: num we say are true that are actually true / num that are true
  - TP (recall) is $(20\% \times 60\% \times n)/(60\% \times n) = 20\%$
  - Replace 20% with any faction f, will be at point (f, f)

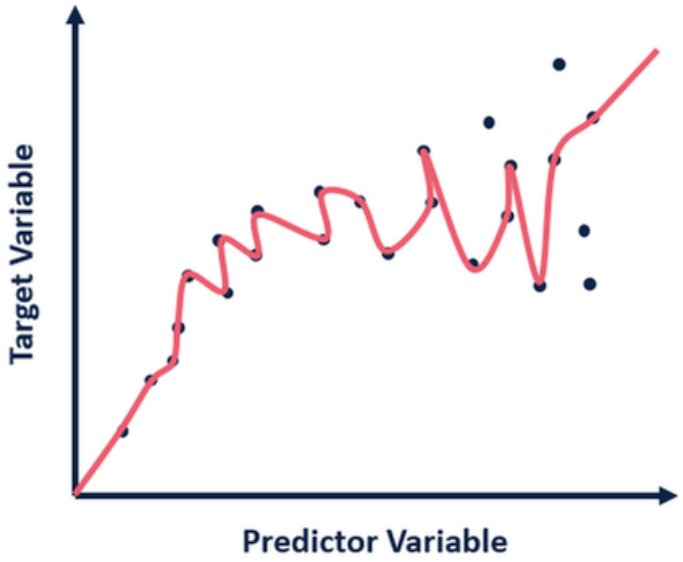
AUC-ROC



- AUC-ROC = "Area under curve"
- Pure ROC is a plot, not a number
  - AUC-ROC converts AUC plot to single number
  - Usually gives single number from 0.5 to 1.0
  - Less than 0.5 means "actively bad"
  - Pros and cons?

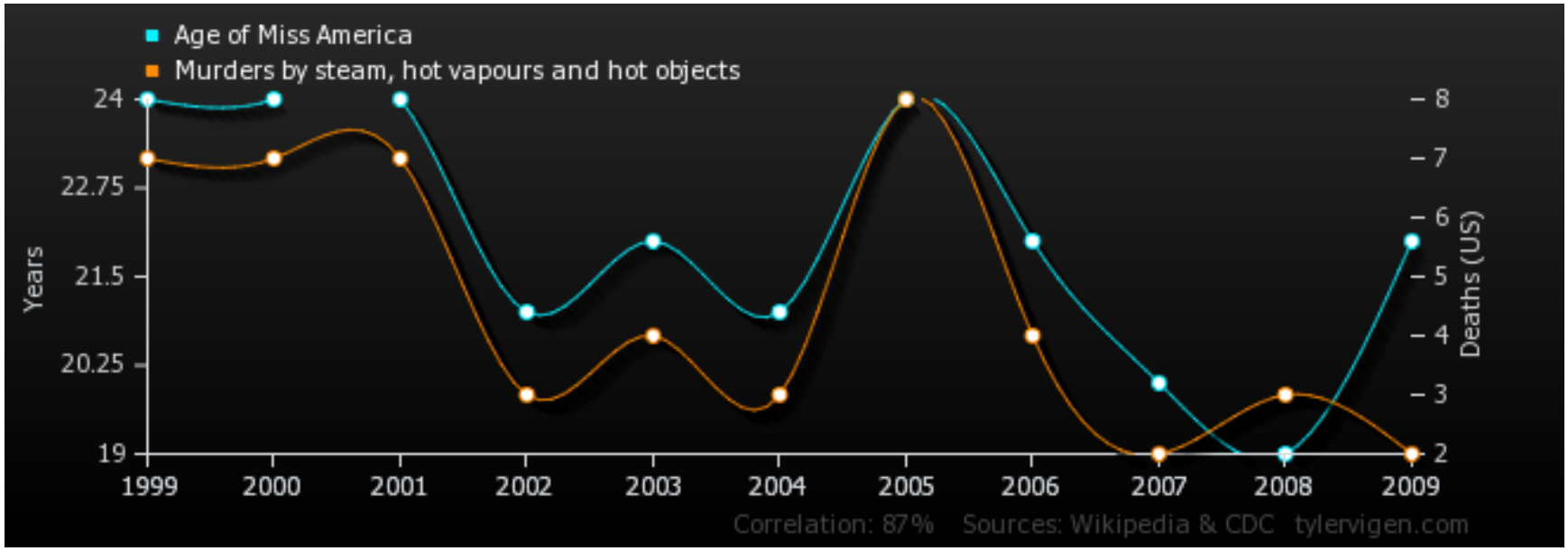
Day 1 AM, Chapter 4: Over-Fitting

Over-Fitting

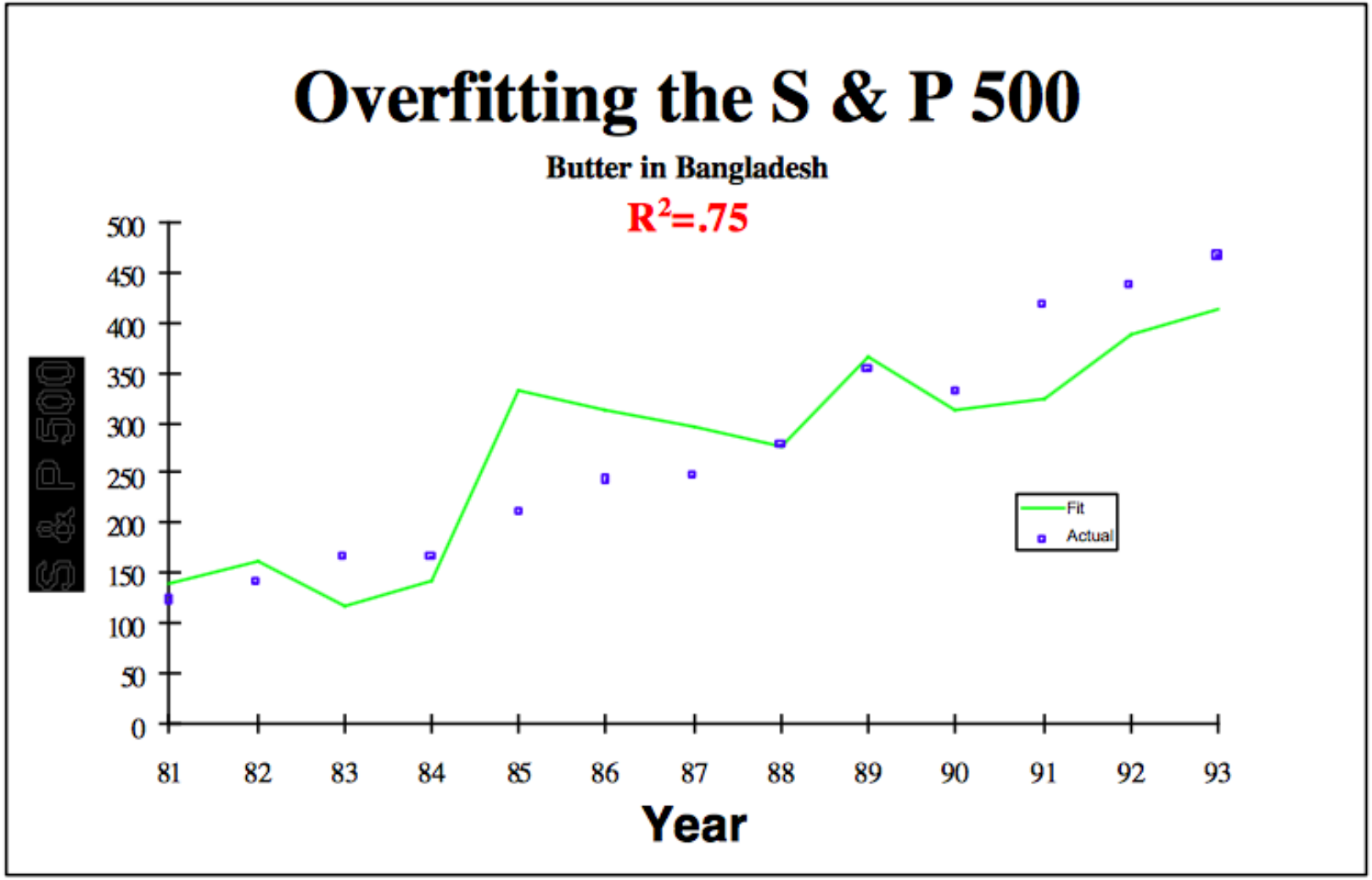


- Fundamental problem in data science/ML
  - Given enough hypotheses to check...
  - One of them is bound to be true

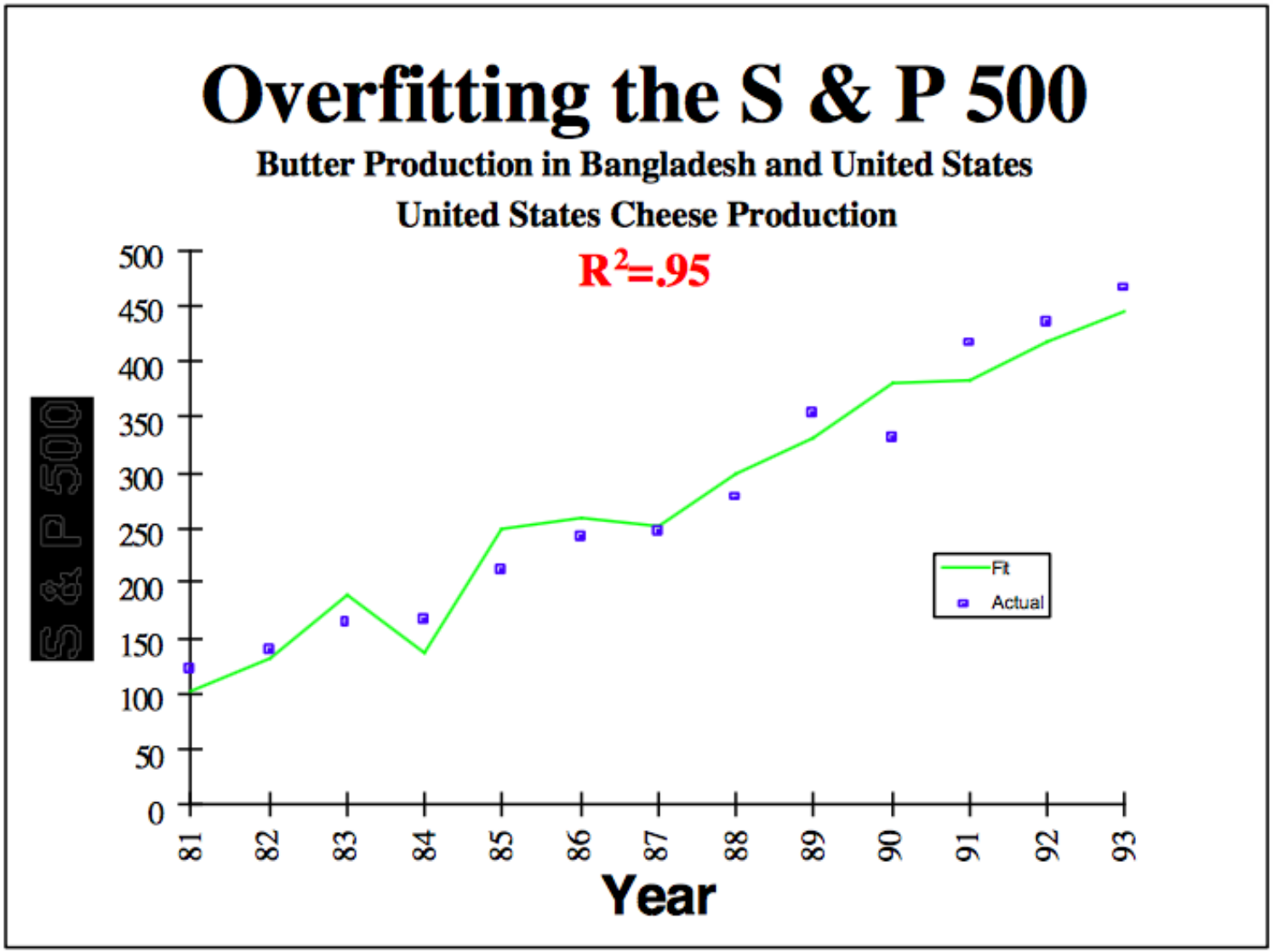
Miss America and Murder-By-Steam



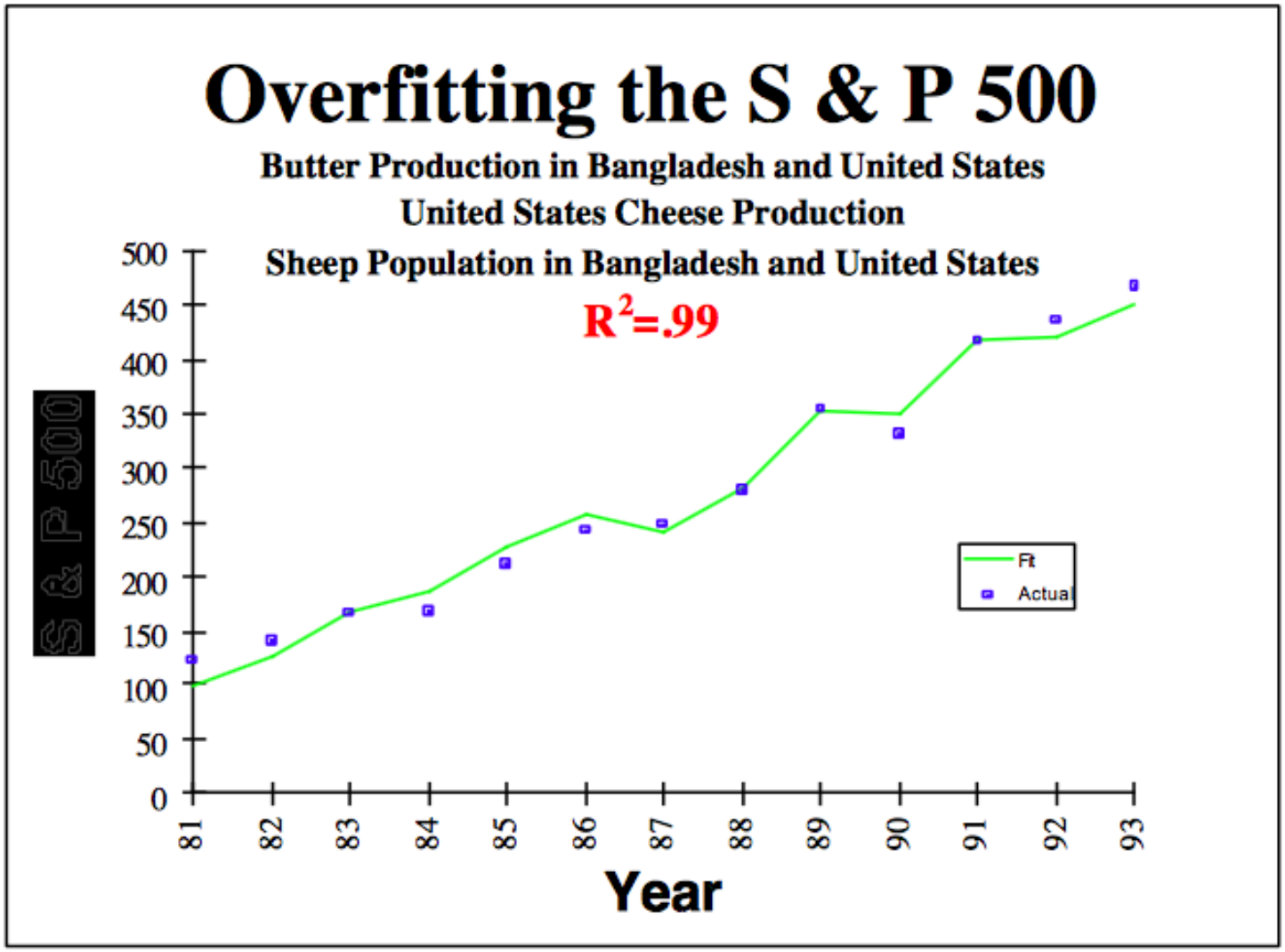
Predicting the S&P 500



Predicting the S&P 500



Predicting the S&P 500



None of these Models Likely to Generalize

- That means:
  - They've learned the input data
  - Not any underlying truth
  - When deployed in the field, likely to fail

"Data Mining"

- Was originally a derogatory term used by stats
  - Meant that you could always find something if you look hard enough

Why Do We Over-Fit?

- Let's dive into this a bit...
- In a nutshell, we have three main sources of error in SL
  - "Bias": error from incorrect model assumptions
  - "Variance": sensitivity of model to data
  - "Uncontrollable Nastiness": weakness in link between x and $Y_{|x}$

Where Do These Errors Come From?

Expected squared error or any prediction is:

$$E[(Y_{|x} - \hat{f}(x))^2]$$

- Here
- x is the data we use for prediction (ex: predict height from weight)
- Y produces the value we are trying to predict from x
  - $Y_{|x}$ is a random variable, distribution conditioned on x
  - Ex: height is random, distribution conditioned on weight
- $\hat{f}(.)$ is the model we are learning
  - Is a random variable because learned from observed data

Expanding This, We Have...

Expected squared error of any prediction is $E[(Y - \hat{f}(x))^2]$

- Expanding:

$$
\begin{aligned}
E[(Y_{|x} - \hat{f}(x))^2] &= E[Y_{|x}^2 + \hat{f}^2(x) - 2Y_{|x}\hat{f}(x)] \\
&= E[Y_{|x}^2] + E[\hat{f}^2(x)] - E[2Y_{|x}\hat{f}(x)] \\
&= Var(Y_{|x}) + E^2[Y_{|x}] + E[\hat{f}^2(x)] - E[2Y_{|x}\hat{f}(x)] \\
&= Var(Y_{|x}) + E^2[Y_{|x}] + Var(\hat{f}(x)) + E^2[\hat{f}(x)] - E[2Y_{|x}\hat{f}(x)] \\
&= Var(Y_{|x}) + Var(\hat{f}(x)) + (E^2[Y_{|x}] - E[2Y_{|x}\hat{f}(x)] + E^2[\hat{f}(x)]) \\
&= Var(Y_{|x}) + Var(\hat{f}(x)) + Bias^2(\hat{f}(x))
\end{aligned}
$$

So There Are Three Sources of Error

Since we have:

$$E[(Y_{|x} - \hat{f}(x))^2] = Var(Y_{|x}) + Var(\hat{f}(x)) + Bias^2(\hat{f}(x))$$

Means error of supervised learner is a sum of:

- "Looseness" of relationship between x and $Y_{|x}$: $Var(Y_{|x})$ (not controllable)
- Sensitivity of the learner to the training data: $Var(\hat{f}(x))$
- Inability of the learner $\hat{f}$ to learn the relationship between x and $Y_{|x}$: $Bias^2(\hat{f}(x))$

It is the sensitivity of the learner to the training data

that leads to over-fitting

In General, Is a Bias and Variance Trade-Off

In "real life"...

- Exceedingly general models are a problem
- They are difficult to train (need tons of data)
- Lacking lots of data, they learn the data set
- So you constrain the model beforehand
- Lowers $Var(\hat{f}(X))$ (damage due to over-fitting)
- But increases so-called "inductive bias" (bias introduced in the model you chose)
- Best we can do: choose sweet spot where error is minimized

Regularization

- Massively important idea in ML

Regularization in ML

- Regularization
  - has come to mean any method to protect against over fitting
- Original meaning consistent with Occam's Razor
- The Razor stated simply: When you have many hypotheses that match observed facts equally well, the simplest one is preferred.
  - Been around for a long time!
  - Credited to William of Ockham (died 1347)
  - First stated explicitly by John Punch, 1639: "Entities must not be multiplied beyond necessity"

Regularization and the Razor

- Bias the algorithm towards a simpler model—lowers variance
  - ▷ But if cost of low variance seems to be high inaccuracy, lower the bias
- But DO NOT have ML practitioner make the choice
  - ▷ Give learning algorithm ability to choose complexity of model
- Done by adding a penalty term to objective function
  - ▷ Penalizes model for complexity

Example: Logistic Regression

- Standard cross entropy loss function:
  - For a data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$:

$$L(\Theta) = -\sum_i \log f(x_i, y_i|\Theta)$$

- Change loss function to:

$$L(\Theta) = \textbf{Penalty}(\Theta) - \sum_i \log f(x_i, y_i|\Theta)$$

- Here, $\textbf{Penalty}(\Theta) = \sum_j \lambda \textbf{abs}(r_j)$
  - λ controls the magnitude of the penalty
  - Typically, try different values of λ during validation

The Lasso

- This is called "the lasso"
- More regularization in the afternoon session!!

Detecting Over-Fitting

- Even if you use regularization, over-fitting is a worry
- Important that we be able to detect it

The Sniff Test

- Detection method number 1: sniff test
  - Does the model "smell" right?
  - Example: Hospital re-admission prediction...
  - 1000 features, logistic regression model
  - Feature "Post-secondary education = True" weighted $5\times$ as high as all others
  - Does that seem right?

Training/Testing/Validation

- Detection method number 2: independent validation and test sets
- Proper methodology
  - Break data into three subsets:
  - Training, validation, testing
  - Training: used to learn the model (min the loss)
  - Validation: used to see if the model is OK
  - Maybe params are wrong... or bad features, or wrong model
  - Use testing to predict accuracy in deployment
  - You often find test accuracy much lower than validation (over-fitting)

Training/Testing/Validation

- No cheating!
  - Temptation: test results bad? Change params, train/validate again
  - But test results become increasingly unreliable
  - My rule of thumb: you've got 3 chances to test
  - After that, test data is stale, you risk over-fitting
  - Are MHT correction methods, but these don't really apply...

Day 1 AM, Chapter 5: Unsupervised Learning

Learning From Unlabeled Data

Sometimes you have a data set without labels

- (height, weight, age, shoe size) quadruples for this class
- Register transactions from Wal-Mart
- User-Movie rating matrix

The goal is simply to fit a model to the data... why?

- Want to learn a model to help humans "understand" the data -or-
- Want a model that presents a simplified version of the data -or-
- Want a model that can generate data

This is "unsupervised learning"

Classically Three Types of UL

(1) Clustering

- Grouping similar points together
- Example: k-means

(2) Dimensionality Reduction

- Map points to a lower-dimensional space
- But preserve relationships among points
- Example: PCA

(3) Generative modeling

- Learn how to generate data similar to observed data
- Example: GANs

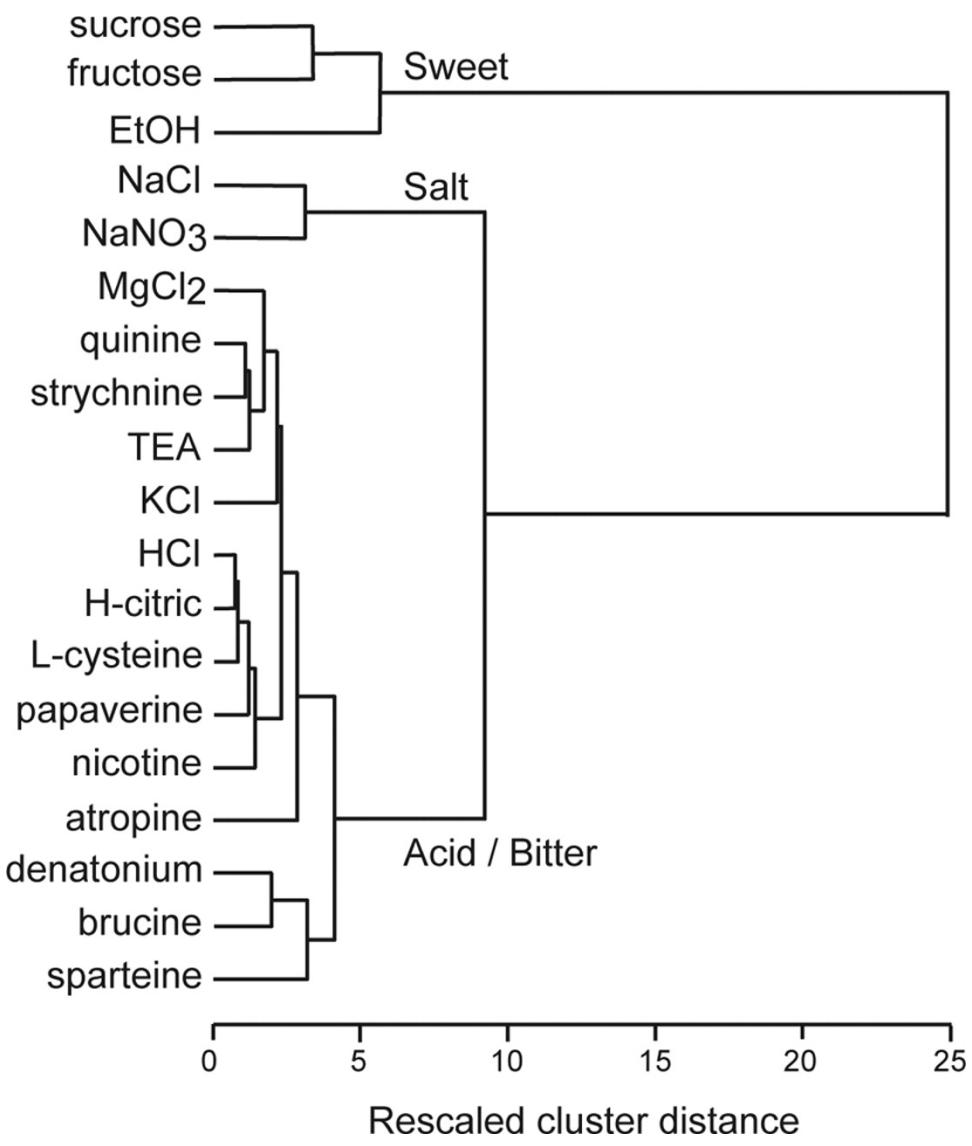
Briefly go over them now...

Clustering

- Classic/oldest algorithm is hierarchical clustering
- Basic Algorithm:

```
while num_clusters > 1 do
  // D is the distance function
  find clusters X, Y that minimize D(X,Y)
  join them
end
```

Hierarchical Clustering



Dim Reduction: Curse of Dimensionality

Say we have 200 data points, two classes (100 points each) in our data set

- Class one: dim 1 values from Normal$(-2, 1)$
- Class two: dim 1 values from Normal$(2, 1)$
- Dim 2 is Normal$(0, 1)$ for all data points

Nearest neighbor works well!

- Tried this in Python
- 98/100 NNs of class one points (Euclidean) are also in class one
- So 1NN classification works really well

What if We Add Meaningless Dims?

Rather than one noise dimension...

- Move to nine noise dimensions (ten dims overall)
- Tried this in Python
- Now, 91/100 NNs of class one points are also in class one (not bad!)

But what about 99 noise dimensions?

- Now only 75/100 NNs of class one points are also in class one
- Not too great

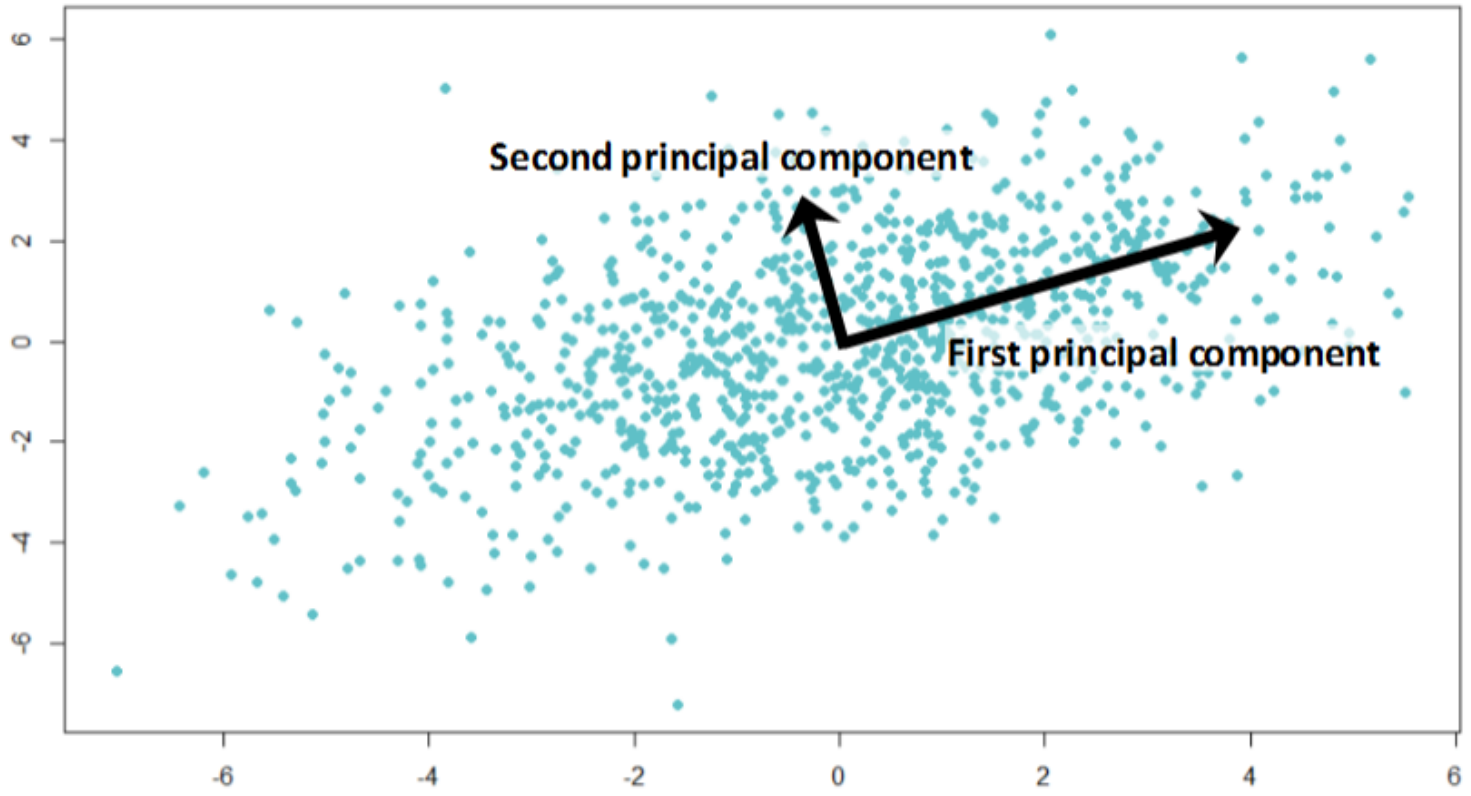
With 999 noise dimensions, only 56/100 NNs are in same class

- Note: a random classifier would expectedly get 50/100 right
- A 1NN classifier in this data would classify only 56/100 correctly
- So not very good!

Classic Dimensionality Reduction Method: PCA

Basic idea: compute a set of orthogonal (perpendicular) basis vectors

- Such that data are uncorrelated wrt those basis vectors
- These are called the "principal components" of the data



Generative Modeling

- Idea: choose params of model to model likely to produce data
- Classic method: have a statistical model (ex: Normal)
  - Choose model params to match data
  - Use something like the "method of moments"
  - Compute mean, variance of data
  - Use those as model params
- Modern ML methods are much more sophisticated!!

Day 1 AM, Chapter 5: Features and Feature Engineering

What Do ML Models Take as Input?

Lots of focus in supervised learning on models

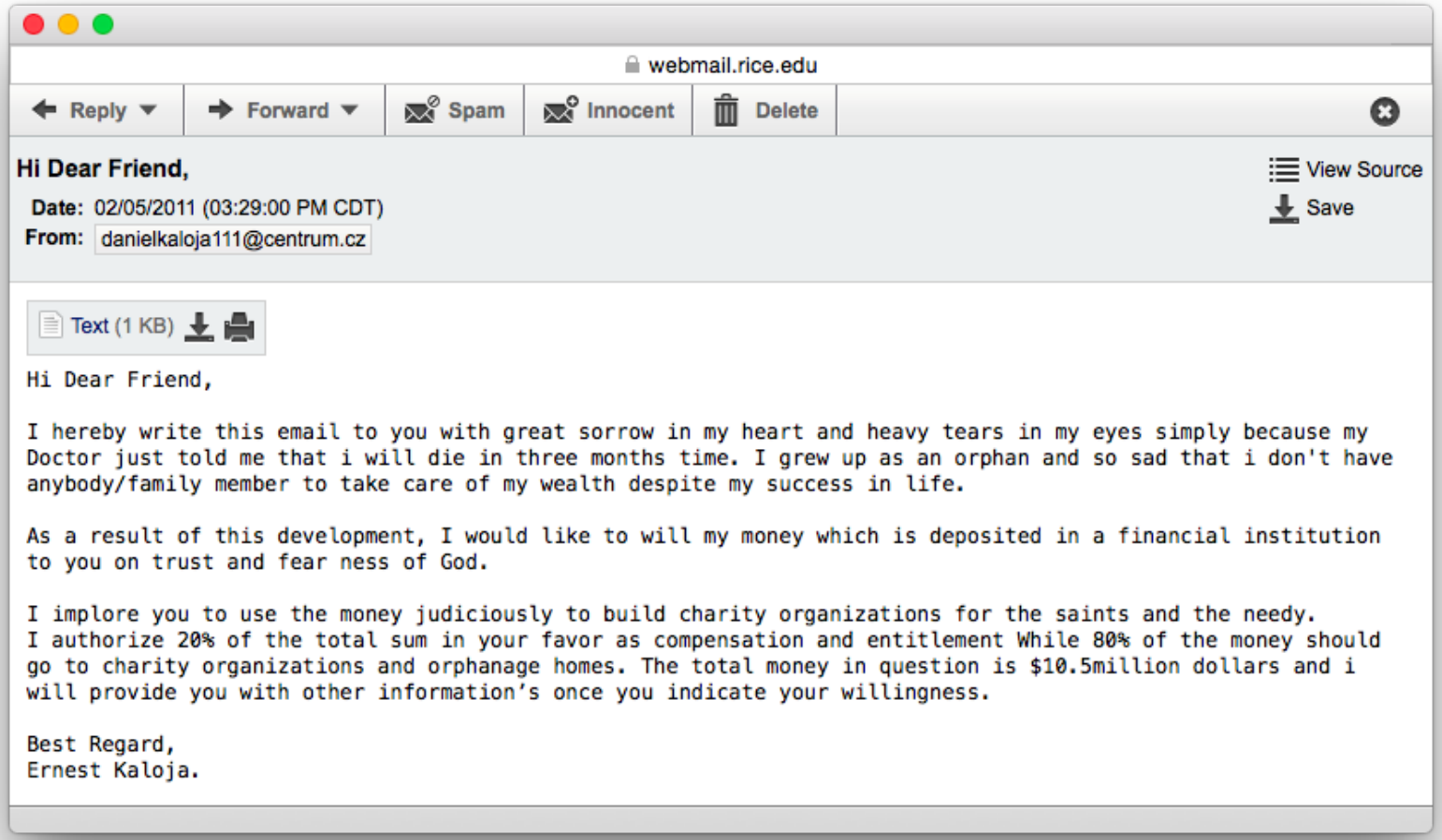
- Linear regression, SVM, kNN, etc.

Almost always less important than feature engineering

- That is, most simple models accept $x = \langle x_1, x_2, \ldots, x_m \rangle$
- Do not accept your raw data!
- How you "vectorize" is often the most important question!

Let's consider feature engineering thru an example...

Example Feature Selection



Feature Vectors via "Bag of Words"

Might build a dictionary

- That is, map from each of m unique words in corpus
- To a number from $\{1 \ldots m\}$
- Then, each email is a vector $\langle 1, 0, 2, 1, 0, 0, \ldots \rangle$
- jth entry is num occurrences of word j
- Problems?

TF-IDF

"Term Frequency"

- Defined as:

$$TF = \frac{\text{num occurs of word in doc}}{\text{num words in doc}}$$

"Inverse Document Frequency"

- Defined as:

$$IDF = \log \frac{\text{num of docs}}{\text{num of docs having the word}}$$

TD-IDF defined as $TF \times IDF$

N-Grams

Words in this doc might not be suspicious

Might be how they are put together

- "great sorrow"
- "heavy tears"
- "financial institution"
- "fear ness"

Idea: also include all 2-grams, 3-grams, 4-grams, etc. as features

What Else?

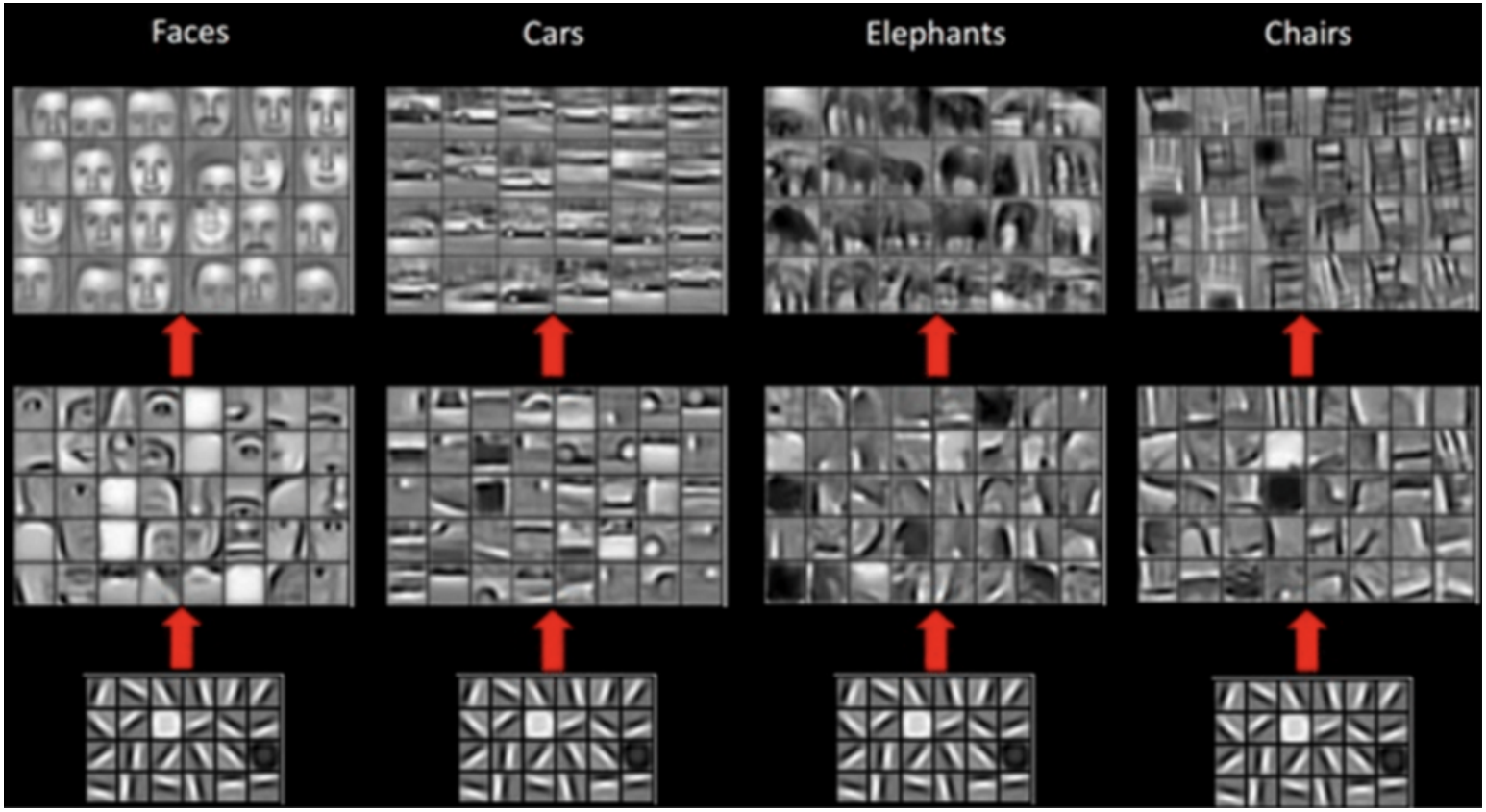
We can include many other features

- Country of sender
- Number of words in email
- Time of day sent
- Was the email sent previously?
- Recipient list disclosed?

Deep Learning and Feature Engineering

- For a long time, ML did poorly on perception tasks
  - Vision
  - Speech recognition
  - Other forms of signal processing
- Why?
  - Unclear how to produce good features
  - What are the features in an image? Colors?
- Deep NNs seem to learn their own features...

Feature Maps



This is Why DL Revolutionary For Perception

- But NOT the best for SL when good features available!
- Then, methods like random forests still preferred

Thank You!

- That's it for the AM session!