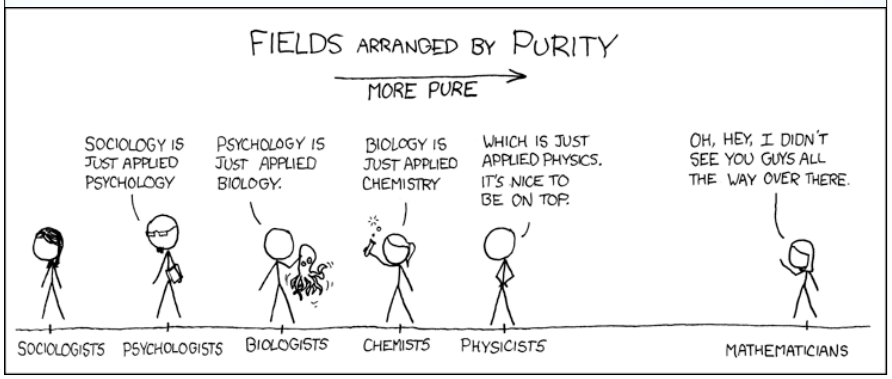
Welcome to the Wonderful World of Machine Learning

Mark Dredze

Machine Learning (CS 601.475/675)



Course Goals

- Learn the fundamentals of machine learning
- Learn to implement machine learning applications
- Learn how to apply machine learning to different settings

Course Policies

- Course website: cs475.org
  - Requires login: JHED ID but separate password system
- Discussion on Piazza
- Assignment submissions and grading on Gradescope
- Readings from Murphy book recommended
- Recitation: Shaffer 303
- Cheating
- cs475@cs.jhu.edu

Prerequisites

Math pre-requisites

Linear algebra (vector spaces, orthogonality, singular value decomposition)
Multivariate calculus (partial derivative, gradient, Hessian, Jacobian)
Probability and Statistics (random variables, probability distributions, expectations, mean, variance, covariance, conditional probability, law of large numbers, Bayes rule, MLE)

CS pre-requisites

Algorithms (Dynamic programming, basic data structures, complexity…)
Programming (Python)

General requirement

Ability to deal with "**abstract mathematical concepts**"

We provide some background, but the class will be fast paced

Todo List

- **Read about the course, including policies on the About section of the website**
- Get password for cs475.org account
- Piazza access
- Get a copy of the textbook or find alternate reading
- Refreshers
  - Mathematical concepts
  - Python

General Course Advice

- Based on course feedback from previous years
  - Course quality above average
  - Course difficulty: way above average
- Read the readings!
  - If you don't like the readings, find other readings
    - Tell me if you find something better!
- Python proficiency essential
- Start assignments early, don't wait

Machine Learning Foundations

Definition 1

- Machine learning allows computers to observe input and produce a desired output, either by example or by identifying latent patterns in the input.
- Data
  - What type of input? What type of output?
- Patterns = algorithm
  - Intuition (empirical) / Objective (theoretical)

Definition 2

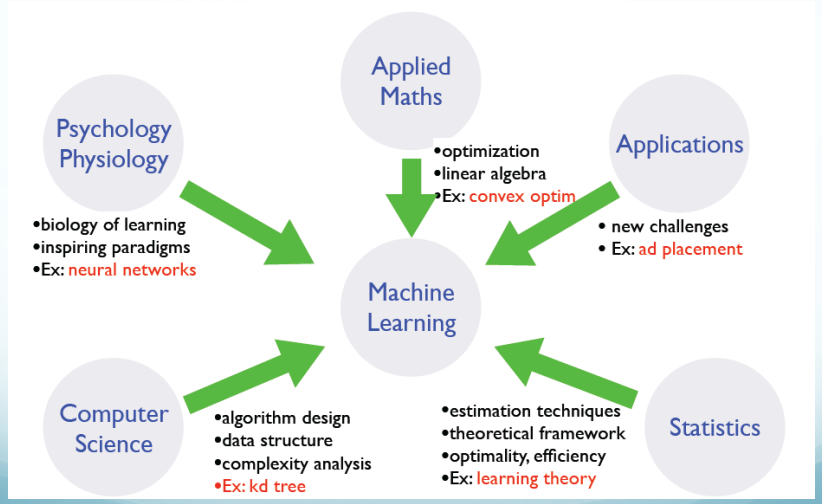
- Using **experience** to gain **expertise**
- Programs that
  - Learn "rules" from data
  - adapt to changes
  - improve performance with experience

Definition 3

Fitting a function to data

- Fitting: Optimization, what parameters can we change?
- Function: Model, loss function
- Data: Data/model assumptions? How we use data?
- ML Algorithms: minimize a function on some data

A Diverse Topic



Slide Credit: Fei Sha

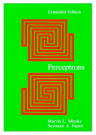
1950: Alan Turing devises the "Turing Test" for AI

1952: Arthur Samuel (IBM) writes first learning program for checkers

1957: Frank Rosenblatt (Cornell) designs the Perceptron- the seed for neural networks
By studying the Perceptron, "the fundamental laws of organization which are common to all information handling systems, machines and men included, may eventually be understood."

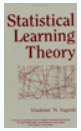
https://en.wikipedia.org/wiki/Timeline_of_machine_learning

1967: Nearest neighbor algorithm created. Used for mapping routes (and for traveling salesman problem)

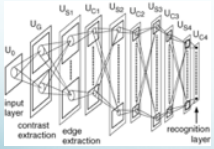


1969: Marvin Minsky, Seymour Papert (MIT) "Perceptrons: An Introduction to Computational Geometry." Criticism of Perceptron demonstrating limitations, research in Perceptrons stops for almost 20 years

1970: Automatic differentiation (precursor to backpropagation) published

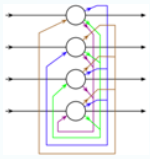


1974: Vapnik and Chervonenkis publish "Statistical learning theory" – Early idea behind SVMs. 1990- kernels, 1992- modern SVM



1980: Neocognitron published by Kunihiko Fukushima – type of ANN which later inspires CNNs

https://en.wikipedia.org/wiki/Timeline_of_machine_learning

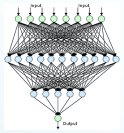


1982: RNNs popularized by John Hopfield

ICML

First workshop on Machine Learning
Second in 1983
Eventually became ICML in 1993

Rise of machine learning as a field
1985- AIStats
1985- UAI
1987- NIPS
1988- COLT

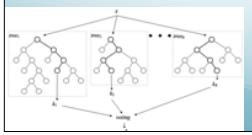


1986: Backpropagation published by David Rumelhart, Geoff Hinton and Ronald J. Williams

1989: Christopher Watkins develops Q-learning for reinforcement learning

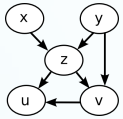
1992: Second death of neural networks

1993: Rise of the SVMs



1995: Tin Kam Ho publishes paper on random forests

1997: IBM's Deep Blue beats Garry Kasparov



2002: Rise of the graphical models

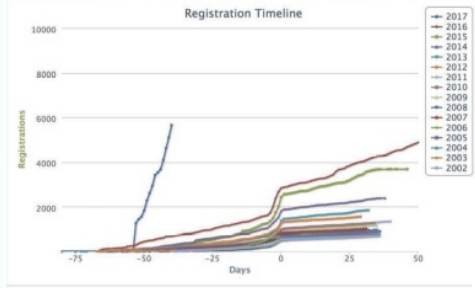
2006: 3rd rise of neural networks (now deep learning)

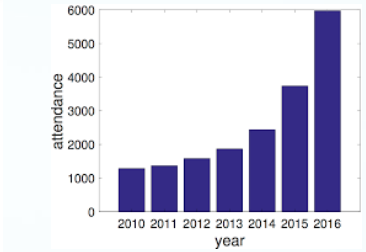
2006: Netflix prize for movie recommendations announced (no winner until 2009)

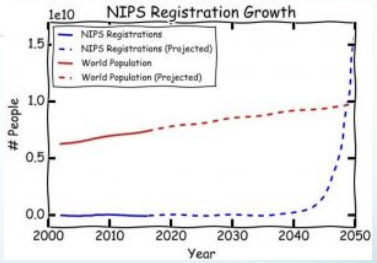
2010: Deep learning takes over

2011: IBM's Watson defeats human champions on Jeopardy!

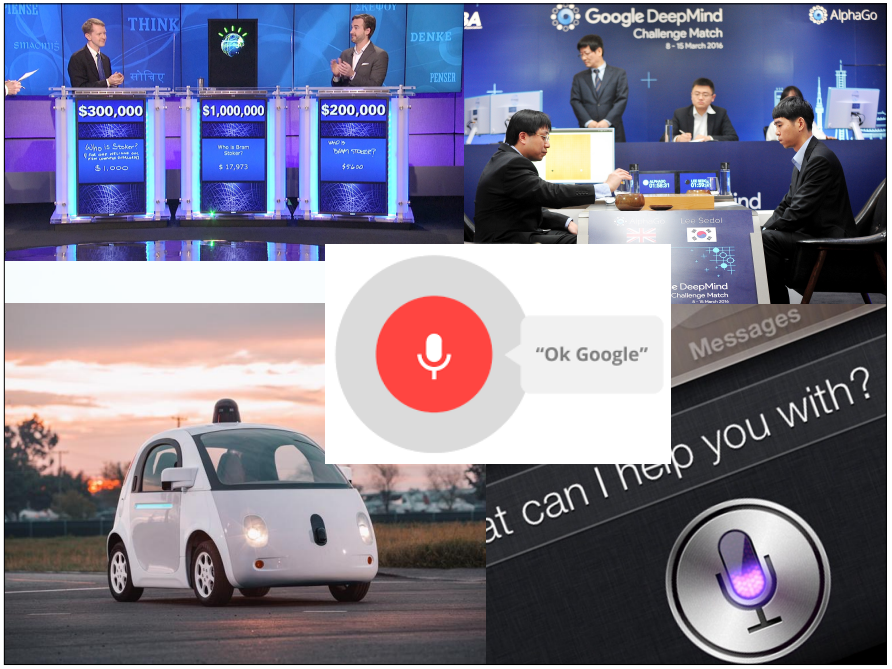
2016: Google's AlphaGo becomes first program to beat an unhandicapped human player in Go





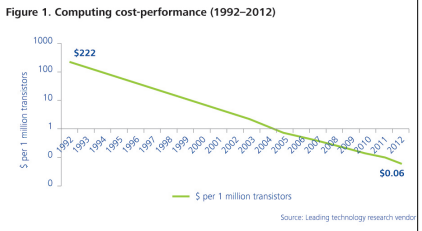


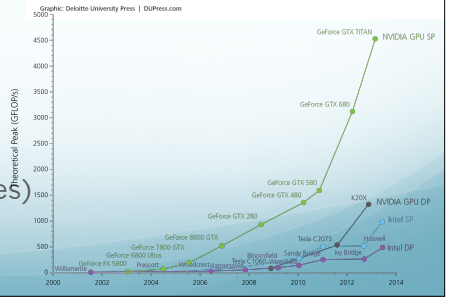
1987·NIPS·90 papers
2014: 1678 submissions, 414 papers
2017: 3240 submissions, 679 papers



Why success?

- (Somewhat) Better algorithms
  - Deep learning
- Better computers
- More data
- The rise of data science
  - More companies hire ML, more success (cycles)





ML vs. other "data sciences"

- Data mining
  - Emphasis on analysis and producing output for human understanding
- Statistics
  - Focused on explanation rather than prediction
- Data science
  - Many definitions. Broadly, combination of data mining, ML and statistics for "data wrangling"

Genres of Machine Learning

Recall our Definition

Fitting a function to data

- Fitting: Optimization, what parameters can we change?
- Function: Model, loss function
- **Data: Data/model assumptions? How we use data?**
- ML Algorithms: minimize a function on some data

Data

- It all comes down to data
  - And the questions we want to answer!
- What information is available for learning?
  - What does the data look like?
  - How is it annotated?
- What output is desired?
  - What should the algorithm produce?
  - How will it be used?

Supervised Learning

- Learning with a teacher
  - Explicit feedback in the form of labeled examples
  - Goal: make prediction
  - Pros: Good performance
  - Cons: Labeled data is difficult to find
- Examples
  - Regression
  - Classification
    - Sort documents by topic
  - Ranking
    - Sort web pages

Unsupervised Learning

- Learning by oneself
  - Only observed unlabeled examples
  - Goal: uncover structure in data
  - Pros: Easy to find lots of data
  - Cons: Finding patterns of interest

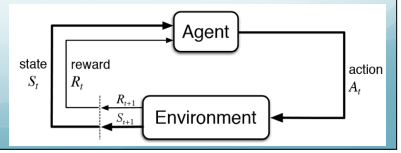
- Examples
  - Clustering
    - Group emails by topic
  - Manifold learning
    - Find a low dimensional data representation

Semi-Supervised Learning

- Labeled examples + unlabeled examples
- Lots of ways to do this
  - Use unlabeled to guide learning in classification
    - Some documents labeled by topic, lots of unsorted docs
  - Graph based models for labeling new data
    - Label propagation
  - Other weak forms of supervision
    - A list of names, learn to extract more

Reinforcement Learning

- Learn a behavior policy by interacting with the world
  - How to navigate in a world
  - Success measured by rewards received by actions
  - Maximize rewards - costs
- No examples
  - You don't know how you did till its over
    - Ex. Chess- was that a good move? Did you win?
- Examples
  - Checkers, chess, Go, video games
  - Robot control
  - Piloting an airplane
  -



How Do We Represent Data?

- Data is complex
- How does a computer algorithm see data?

High Dimensional Vectors

- A learning example is a vector of length M x
- Examples a drawn from an underlying distribution

$$x_i \in \Re^M$$

- Each dimension represents a feature
  - Feature functions

$$x[j] = f_j(\textbf{item})$$

- A collection of N examples

$$\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^N$$

What do data look like?

What do data look like?

- Word counts from documents in a corpus
- Nucleotides in a sequence of DNA
- Customer satisfaction survey results for online purchases
- Scatter plot of population characteristics
- RGB pixel data from images or videos
- Game logs from expert chess players

Data Features

x

- Designing feature functions is critical
  - Well designed representations greatly effect performance
- How should we design features?
  - Features are application specific
  - You need to know about biology/vision/speech/etc.
- Since this is domain specific we won't talk much about it
- More on this in last lecture

Linear Regression
Our First Algorithm

If she loves you more each and every day,
by linear regression she hated you before you met.

Why Start Here?

- Well-known algorithm
- Not strictly a “machine learning” algorithm
- Can learn about fundamentals with a simple example

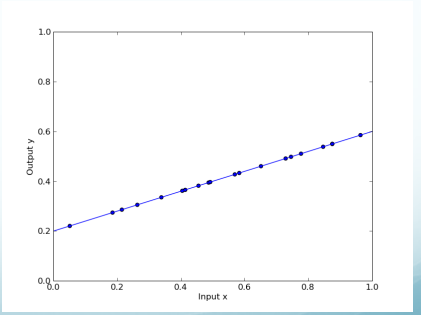
Example

- I run a real estate website. I want to list properties for sale and provide estimates of how much they will sell for
- Goal: Predict home prices
- Data: Previous home sales
  - Housing facts: size, bedrooms, age, neighborhood, listing price, ...
- How do I do this?

Zillow

Data Model

- Assume dependent variable (y) can be modeled by a *linear function* of the input variables (x)
- $y = w\mathbf{x} + b$
- 2 dimensions
  - Compute w and b from two points
- Solution?
  - Given y and x, solve for w and b



Regression

- Data $\{(x_i, y_i)\}_{i=1}^{N} \quad x_i \in \Re^M \quad y_i \in \Re$ ← Continuous Values
- Learn: a mapping from x to real valued y
  - f(x) = y
- Examples
  - GPAs
  - Stock price
  - Miles per gallon
  - Age of author

Try it at Home!

- Linear regression demo
- http://mste.illinois.edu/users/exner/java.f/leastsquares/
- http://onlinestatbook.com/2/regression/linear_fit_demo.html
- https://www.geogebra.org/m/FUe3HfRf

Recall Definition

Fitting a function to data

- Fitting: Solve for w given y and x
- Function: linear function
- Data: assume dependent variable linear combination of independent variables
- Minimize a function
  - What function are we minimizing?

What is the goal?

- You probably know linear regression from statistics
  - "In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X." (Wikipedia)
- Our goal: predict correctly the next example
  - Minimize: reduce prediction error
    - More to say about this later

Loss Functions

- Machine learning algorithms minimize loss functions
  - Or some substitute for a loss function
  - The best solution minimizes the loss function*
- Definition
  - A function that maps between (true label, prediction) -> non-negative number
  - 0 = perfect prediction

* Maybe

Loss: What We Minimize

- Loss measures the badness of our prediction
- What's a good loss function?
  - It depends on task and goals
- Regression loss function?
  - Proposal: How far are you from the correct answer

Sum of Squares Loss

$$f(x) - y$$

$$(f(x) - y)^2$$

$$\sum_{i=1}^{n} (f(x_i) - y_i)^2$$

Over all answers

Correct answer

$$\sum_{i=1}^{n} (w \cdot x_i - y_i)^2$$

Predicted answer

Recall Definition

Fitting a function to data

- Fitting: Solve for w given y and x
- **Function: linear function**
- Data: assume dependent variable linear combination of independent variables
- Minimize a function
  - What function are we minimizing?

Hypothesis Class

- Learning algorithm selects hypothesis from hypothesis class
- Hypothesis class
  - A set of possible hypotheses (functions) that can be used to label the data
  - Can be finite or infinite
- Each learning algorithm has a hypothesis class
  - Fitting selects the best hypothesis using observed data

What is best hypothesis?

Choosing Hypothesis Class

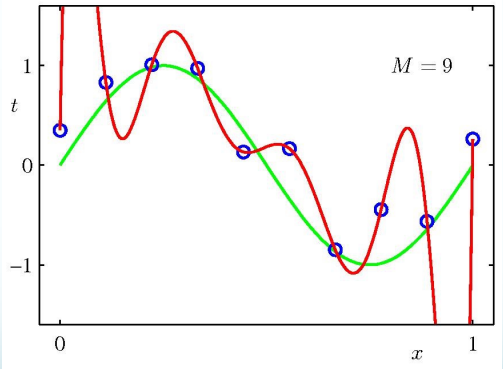
- What sort of hypothesis class do we want?
- The hypothesis class should contain the optimal hypothesis
- The hypothesis class for which our algorithm will find the best performing hypothesis

What is the difference?

Hypothesis Tradeoff

- Rich hypothesis class
  - Over-fitting
- Easier to search hypothesis class
  - Under-fitting
- Realization: we are unlikely to ever find the hypothesis that exactly explains the data
- Simplifying assumptions help find a reasonable hypothesis
- Select hypothesis class based on knowledge of data

Under/Over Fitting



Hypothesis Class

- What is the hypothesis class of linear regression?
- Linear functions
  - All possible linear functions encoded by parameters w
  - Hypothesis chosen by setting parameter values w
- How large is the hypothesis class?
  - Infinite

What is Learning? Another View

- Select a hypothesis from the hypothesis class
  - Model parameters correspond to hypotheses
  - Learn parameters of model based on data
- How do we write learning algorithms?
  - Theory: Objective driven
    - Write an objective that you want to minimize
    - Develop a procedure to minimize the objective
      - This is called the learning algorithm
  - Empirical: intuition driven
    - Many algorithms based on motivation and heuristics
    - Post hoc analysis of objective

Recall Definition

Fitting a function to data

- Fitting: Solve for w given y and x
- Function: linear function
- **Data: assume dependent variable linear combination of independent variables**
- Minimize a function
  - What function are we minimizing?

Learning Settings

- What information is available for learning?
  - What does the data look like?
  - How is it annotated?
- What output is desired?
  - What should the algorithm produce?
  - How will it be used?

Recall

- Our goal: predict correctly the next example
  - Minimize: reduce prediction error

Goal of Learning

- "Reduce prediction error"
  - On what?
- True error

$$error_D(h) = P_{x \in D}[\ell(h(x_i), y_i) \neq 0]$$

  - We need infinite data to measure this!

Goal of Learning

- If we can't measure true error, how do we judge learning success?
- Should an algorithm maximize performance on observed data?
- Proposal: measure error on the given data
  - Call this the "training data"
  - Is this a good idea?

Measuring Error

- Very bad idea
- Recall: machine learning cares about prediction (the future)
  - How well will the system do once deployed?
- Memorizing the training data is easy
  - Most hypothesis classes are rich enough to exactly learn the training data

Bayes Optimal Rule

Ideal goal: Construct **prediction rule** $f^* : \mathcal{X} \to \mathcal{Y}$

$$f^* = \arg\min_f \mathbb{E}_{XY}\left[\text{loss}(Y, f(X))\right]$$

Bayes optimal rule

Best possible performance:

Bayes Risk $R(f^*) \leq R(f)$ for all f

BUT… Optimal rule is not computable - depends on unknown P_{XY} !

Slide from Raman Arora

Experience - Training Data

Can't minimize risk since P_{XY} unknown!

Training data (experience) provides a glimpse of P_{XY}

(observed) $\{(X_i, Y_i)\}_{i=1}^n \overset{i.i.d.}{\sim} P_{XY}$ **(unknown)**

independent, identically distributed

Provided by expert, measuring device, some experiment, …

Slide from Raman Arora

Performance Revisited

Performance: (of a learning algorithm)

How well does the algorithm do on average
1. for a test instance X drawn at random, and
2. for a set of training instances and labels $D_n = \{(X_i, Y_i)\}_{i=1}^n$ drawn at random

Expected Risk (aka Generalization Error)

$$\mathbb{E}_{D_n}\left[R(\hat{f}_n)\right] \equiv \mathbb{E}_{D_n}\left[\mathbb{E}_{XY}\left[\text{loss}(Y, \hat{f}_n(X))\right]\right]$$

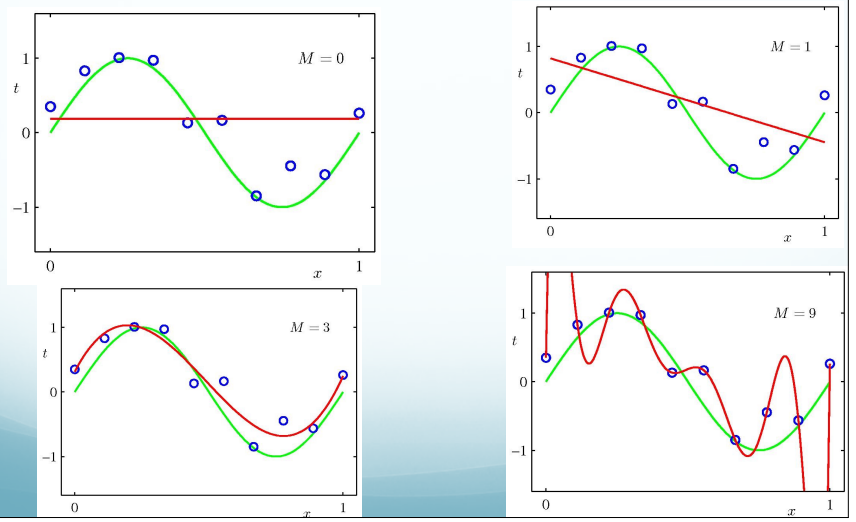
Generalization

- Generalization
  - The ability of an algorithm to generalize knowledge learned from observed data to new data
- Simple example: memory based classifier
  - Binary classification
  - Train: remember each example
  - Test: if we have seen an example before, report label
    - Otherwise, guess randomly
- Train error: 0% Test error: 50%
  - This is called over-fitting

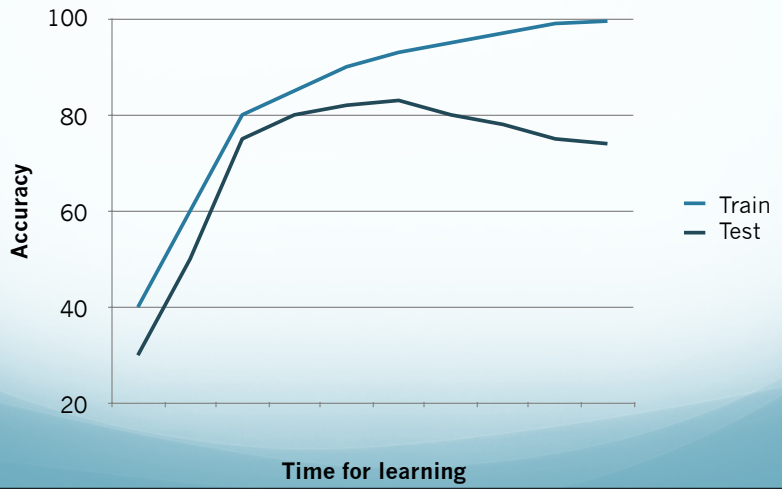
Bias vs. Variance

- How do we achieve good generalization?
- Tradeoff bias and variance
  - Bias- favor certain predictions
  - Variance- diversity of predictions
- Under-fit observed data
  - Favors bias- large changes to input have same output
- Over-fit observed data
  - Favors variance- small changes to input can dramatically change output
- Tradeoff key to generalization to new data

Under/Over Fitting



Typical Learning Curves



No Free Lunch Theorem

- Models make assumptions about the world
  - An assumption less model — allows every hypothesis — would be easily distracted by unnecessary detail and have difficulty learning
  - We make assumptions to focus the models on what's important
  - These assumption limit the model if we are wrong
- There is no single model that works best for every problem

Measuring Error: The Right Way

- Collect two sets of data
  - Train data- use for training algorithm
  - Test data- only use for evaluation
    - Only good if you've never seen it before, not if you continuously tune on it
- How do we balance bias/variance?
  - Tune parameters on development/validation data

Two Common Methods

- Train/dev/test
  - Use held out sets for evaluation
  - Good when you have lots of data
- Cross validation
  - Create many train/test splits
  - Randomly sample train and test data splits
  - Divide data into folds, use each fold for testing once
  - Reasonable when you don't have enough data

What Causes Error?

- Noise error
  - An example has an incorrect or inconsistent label
  - Our data representation fails to encode necessary information
- Model error
  - Hypothesis class is deficient
- Parameter estimation error
  - The model parameters are wrong
- Search error
  - We made a mistake in scoring a prediction
  - Common in tasks with complex output

Recall Definition

Fitting a function to data

- **Fitting: Solve for w given y and x**
- Function: linear function
- Data: assume dependent variable linear combination of independent variables
- Minimize a function
  - What function are we minimizing?

Linear Regression

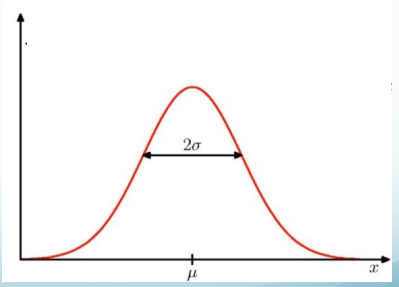
- We assumed output (y) linear combination of inputs
- This is wrong
  - Rarely is data actually linear
- But realistic assumption may be too complex
- A reasonable middle ground?

Noise from a Gaussian Distribution

$$N(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

$$E[x] = \mu$$

$$\text{var}[x] = \sigma^2$$



Noise

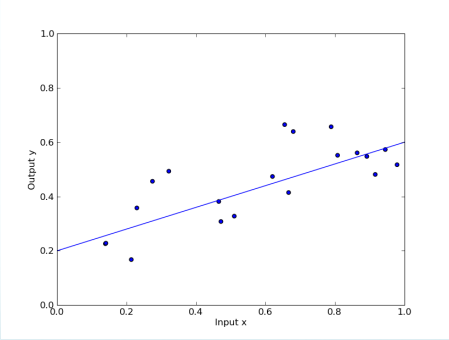
- Assume output permuted by Gaussian noise

$$y = h_w(x) + \varepsilon$$

$$\varepsilon \sim N(\mu, \sigma^2), \quad \mu = 0, \quad \sigma^2 = 1$$

- The data isn't really generated in this way
  - Assume that it is for sake of modeling

Linear Regression with Noise



Probability of Output

$$p(y \mid x, w, \sigma^2) = N(y, \sigma^2)$$
$$= N(h_w(x), \sigma^2)$$

Least Squares Regression

- Fitting: Solve for w given x and y
- Function: Linear function + Gaussian noise
  - Loss function: squared error
  - Assumes output (mostly) a linear combination of input
- Data: fit a model to training data, evaluate on held out data
- Minimize a function
  - What function are we minimizing?

Which Function are we Minimizing?

- Which is the best hypothesis?
  - Which setting of the parameters w is best?
- Select the hypothesis that *best explains* the observed data
- Select the hypothesis that minimizes the error

Explaining the Data

- What does it mean to explain the data?
  - Maximize the likelihood of the data
  - Likelihood = probability of observing data
- Writing likelihood
  - Assume data generated from our linear regression model

Maximum Likelihood For Gaussians

Maximum Likelihood for Regression

Sources of Error

- Noise error
  - An example has an incorrect or inconsistent label
  - Our data representation fails to encode necessary information
- Model error
  - Hypothesis class is deficient
- Parameter estimation error
  - The model parameters are wrong
- Search error
  - We made a mistake in scoring a prediction
  - Common in tasks with complex output

Bias?

- Gaussians: maximum likelihood estimate is biased
  - This is ok if we have infinite data
  - We never have infinite data!
- Over-fitting: avoid it by favoring certain solutions
- Regularization
  - Add term to objective to favor different considerations
  - What should we favor?
    - Occam's Razor: simpler is better
    - Favor small weights
      - Our parameters should be small

Regularized Least Squares

Regularized Least Squares

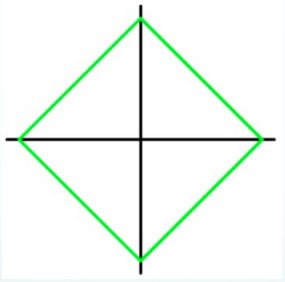
- Tradeoff: low error and small weights

$$E_D(w) = \frac{1}{2}\sum_{i=1}^{n}(y - w \cdot x)^2$$

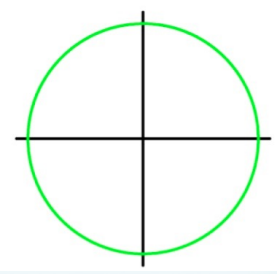
$$E_W(w) = \frac{1}{2}w^T w$$

$$E_D(w) + \lambda E_W(w) = \frac{1}{2}\sum_{i=1}^{n}(y - w \cdot x)^2 + \lambda\frac{1}{2}w^T w$$

Regularization Behavior



q=1 L1 (Lasso)



q=2 L2 (quadratic)

Bias vs. Variance

- Expected squared loss can be written as

$$\mathrm{E}[L] = \int \{f(x) - h(x)\}^2 p(x)dx + \int \{h(x) - y\}^2 p(x,y)dx\,dt$$

  - f(x)- prediction function
  - h(x)- true regression function
  - y- provided label (noisy)

  - First term- minimize loss
  - Second term- error from noise

Bias vs. Variance

- Imagine we can sample many datasets D from the underlying distribution
- Integrate the first term (which represented accuracy of the model)

$$(f(x,D) - h(x))^2$$

  - Depends on a particular sample of data D
- What is the expectation of this term over many samples of D?

Bias vs. Variance

$$\mathrm{E}_D[(f(x,D) - h(x))^2] =$$

$$(\mathrm{E}_D[f(x,D)] - h(x))^2 + \mathrm{E}_D[(f(x,D) - \mathrm{E}_D[f(x,D)])^2]$$

Bias Variance

- For learning we want to minimize this function
- The result is a tradeoff between bias and variance

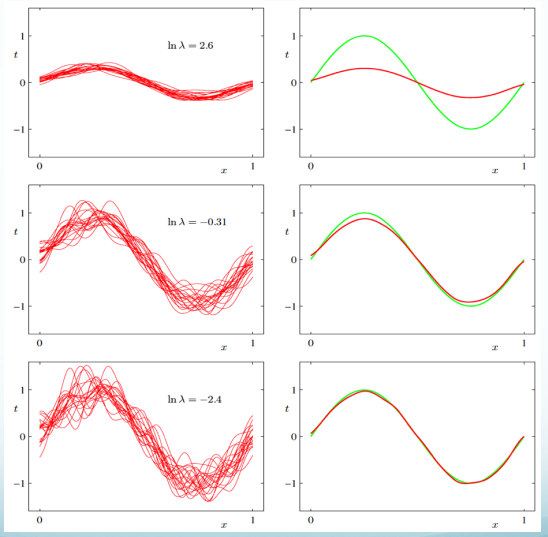
Parameter Tradeoff

- The regularization parameter controls bias vs. variance
- Higher λ = more regularization
  - Favors bias
- Lower λ = less regularization
  - Favors variance

Dr. Philip Graff and Dr. Jared Markowitz

More Regularization

Less Regularization



More Bias

More Variance

Problems

- Maximum likelihood under-estimates variance and over-fits
  - Try to fix using regularization
- How do we decide model complexity?
  - Parameter tuning on held out data
- Is there a better way?
  - Bayesian methods
  - more on this later

Summary: Machine Learning Fundamentals

Fitting a function to data

- Fitting: Optimization, what parameters can we change?
- Function: Model, loss function
- Data: Data/model assumptions? How we use data?
- ML Algorithms: minimize a function on some data

Summary: Machine Learning Fundamentals

- Data representation $\{(x_i, y_i)\}_{i=1}^{N} \quad x_i \in \Re^M \quad y_i \in \Re$
- Loss functions
- Hypothesis class and tradeoffs
- Generalization and bias/variance tradeoff
- Learning settings: Supervised and unsupervised
- Regularization
- Sources of error

Next Time:
On to Classification!