An Introduction to Machine Learning

Mitchell Goist
Pennsylvania State University

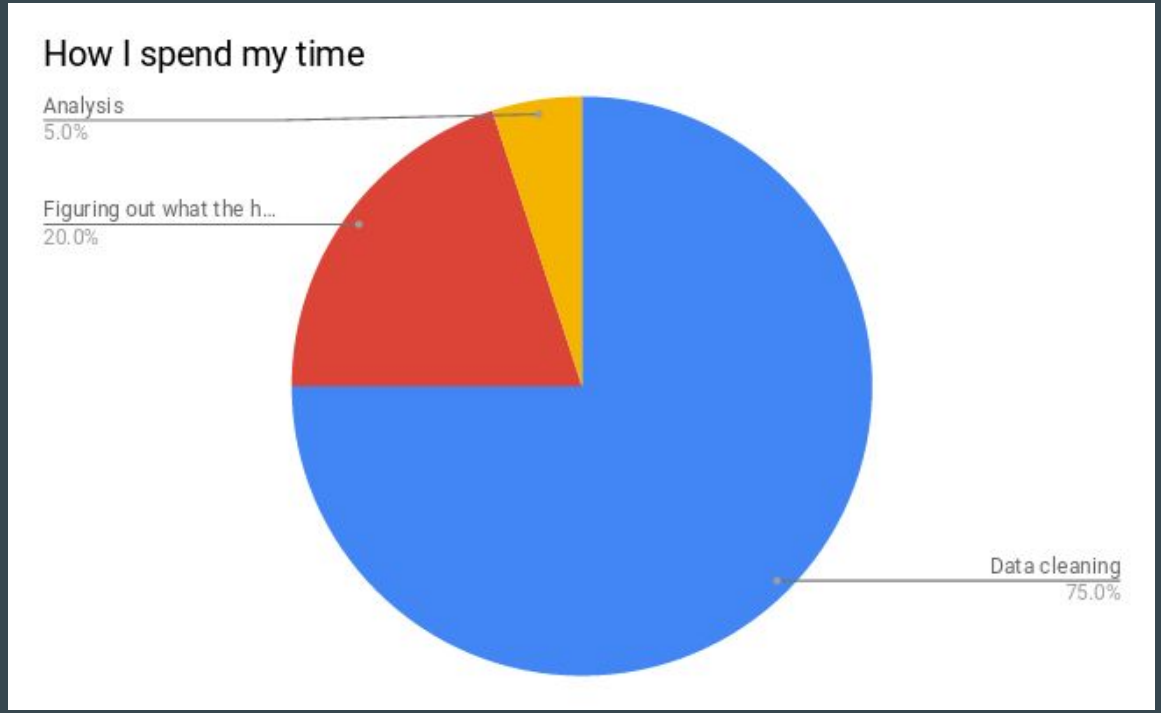
“When in doubt, use brute force.”

-- Ken Thompson

What this talk is about

1. Learning from data
    a. Computational generalization
    b. Bias - variance tradeoff
2. Practical examples of model classes
    a. Nearest neighbors
    b. Ensemble methods
3. Picking the best models
    a. Hyperparameter tuning
    b. Classifier comparison
4. Working with real world social data
    a. Data quality
    b. Complex relationships

What this talk is about

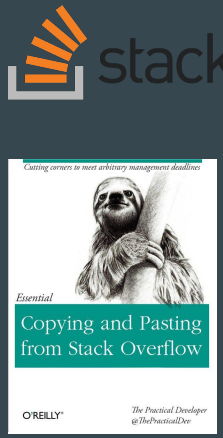


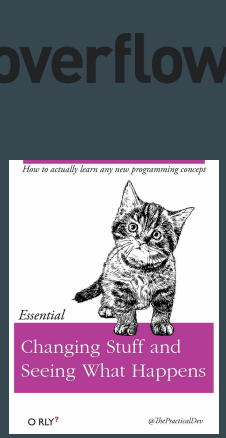
What this talk is about

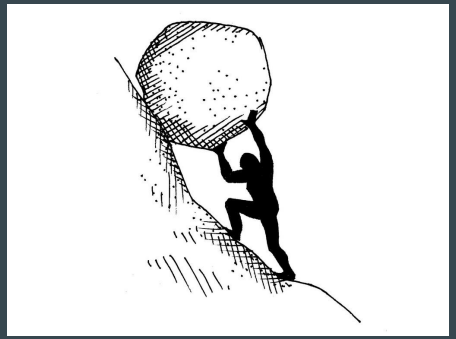
Analysis

Data Cleaning

Figuring out what the hell to do



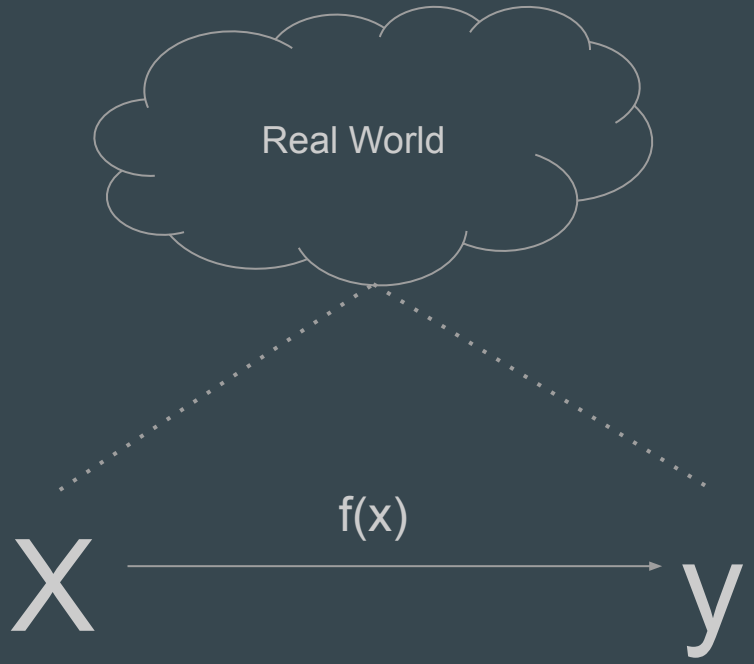




?

RTFM!

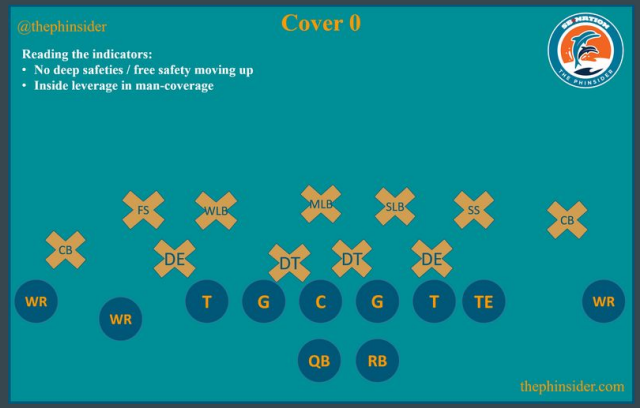
What is machine learning?



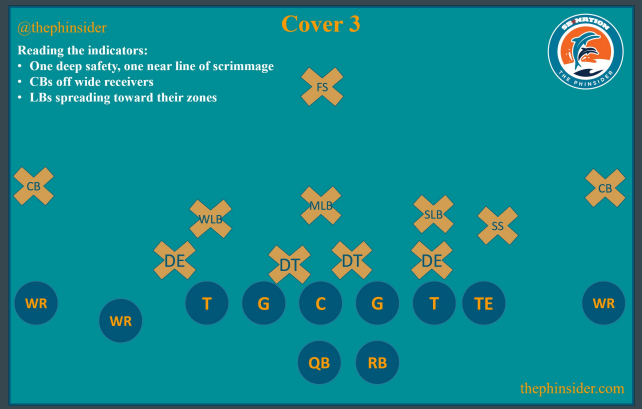
Find g(x) from X_D and y_D, where...

- D is a given dataset drawn from the real world
- g(x) is the best guess for f(x)

Listing out our guesses

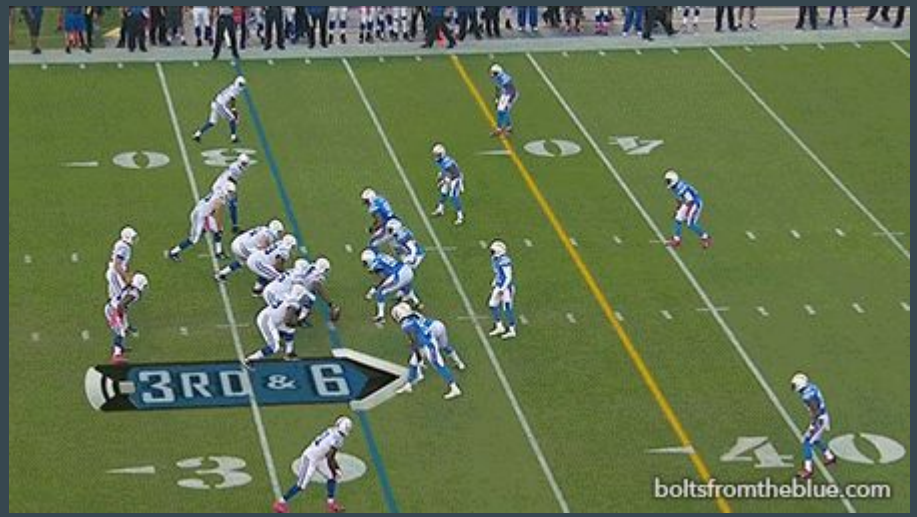


→ Cover 0



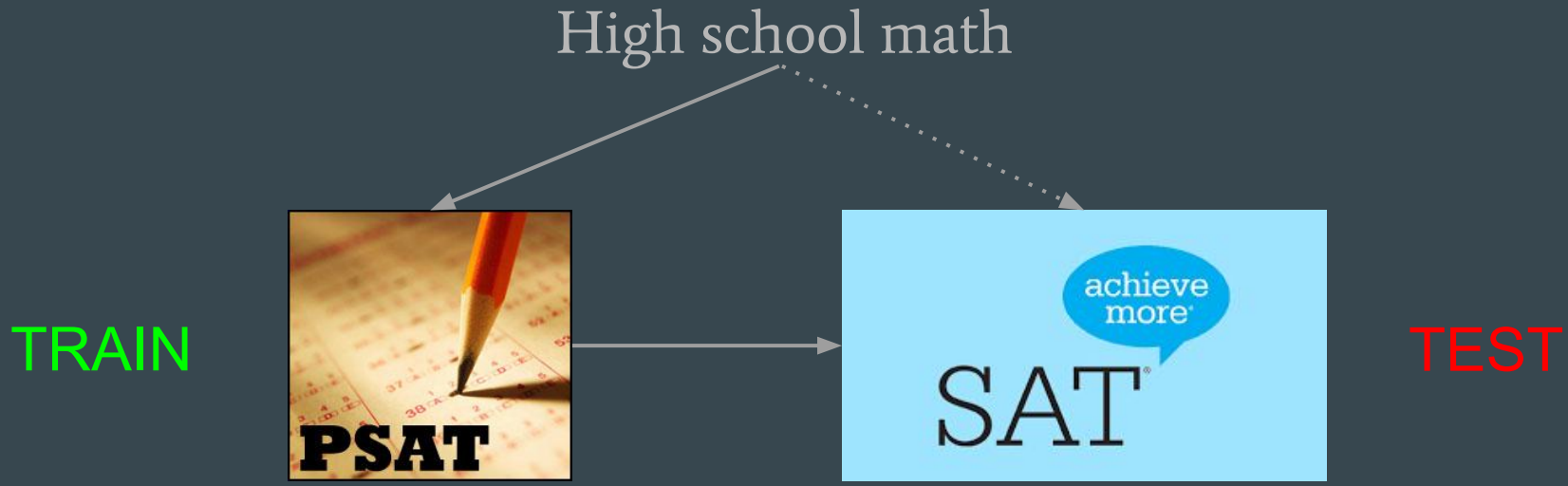
→ Cover 3

But they’re called guesses for a reason



Getting our guesses right before it's too late

We can simulate the process of learning from the real world, by dividing our data into training (our dataset) and test set ("real world"). We assume that if g(x) predicts well on the test set, then it will predict f(x) well.



The goal is generalization

- The goal of machine learning is *generalization*.

To learn a pattern in certain features of the real world, while accounting for the noise inherent in the gathering and analyzing of those features.

Computational Generalization

The Guessing GameHypothesis space

$$h_{1,} \ldots, h_{m} \ni H$$

$$h = (\mathbf{X}, y)h1, \ \ldots, hm \in \mathcal{H}$$

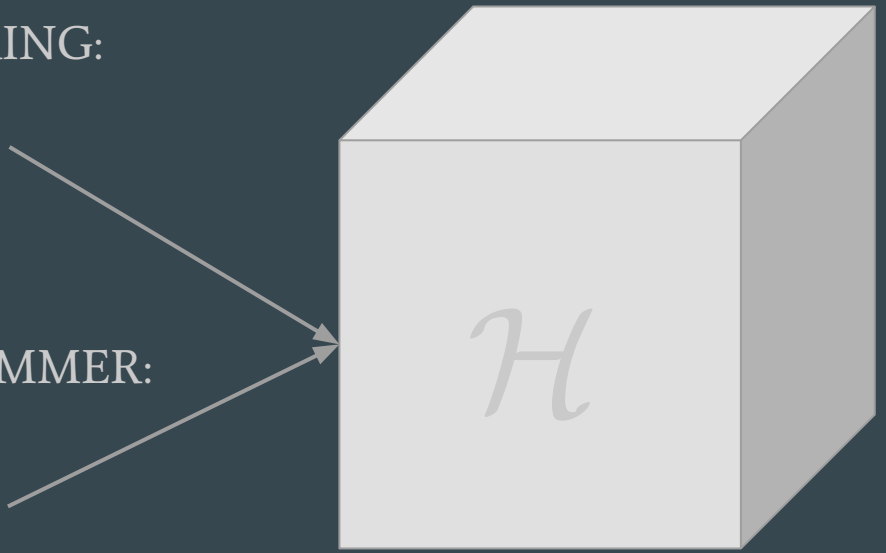
E.g., rule based classifier

IF rain_yesterday==TRUE & season==SPRING:

THEN rain=1

IF rain_yesterday==FALSE & season==SUMMER:

THEN rain=0



Generalization and PAC learning

$$\text{error}(h_i) = P[h(X) \neq c(X)]$$

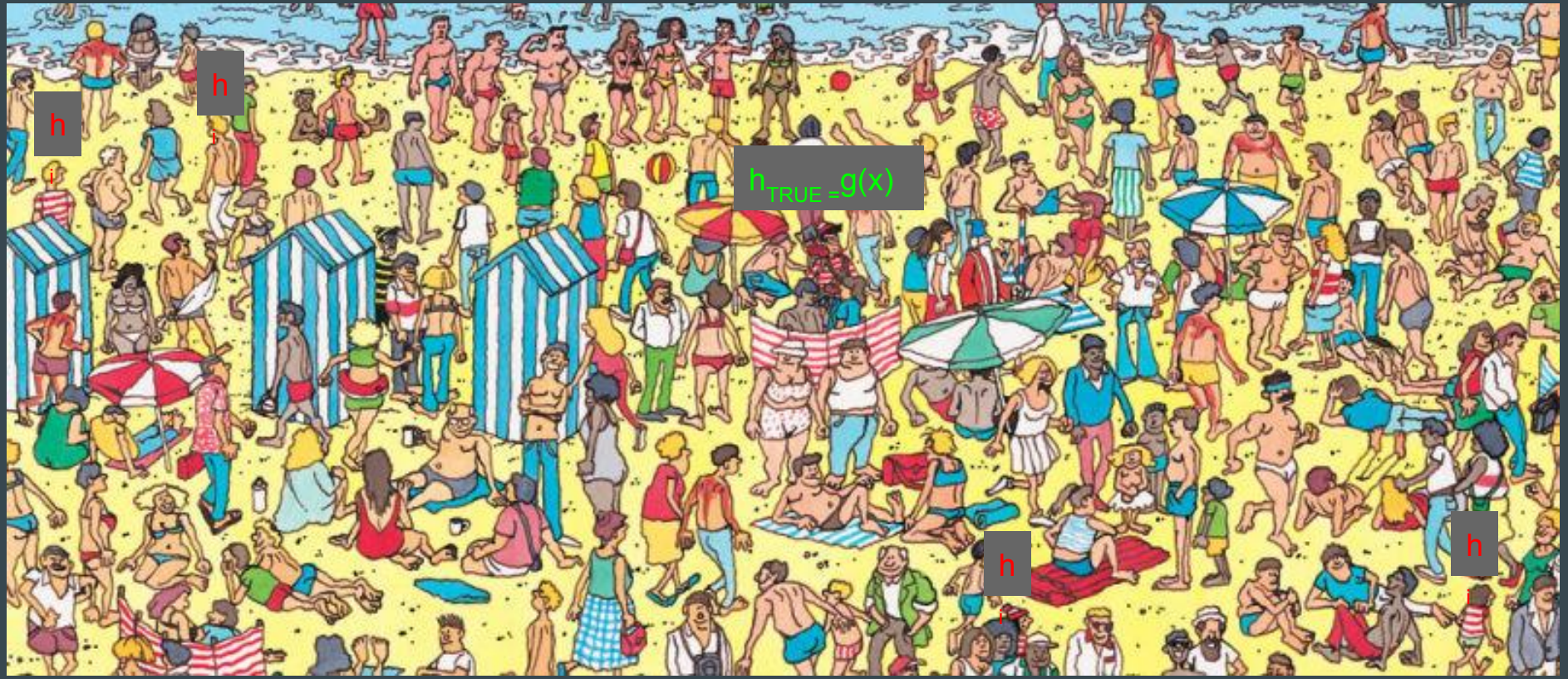
$$\text{true}(h_i) = P[h(X) = c(X)]$$

Probably Approximately Correct (PAC) learning (Valiant, 1984)

$$P[\text{error}(H_M) \leq \epsilon] \geq 1 - \delta$$

- ϵ: “probably” - what is good enough
- $1 - \delta$: “approximately correct” - our overall confidence
- Defines when learning is feasible based on size of hypothesis test, error, and confidence.

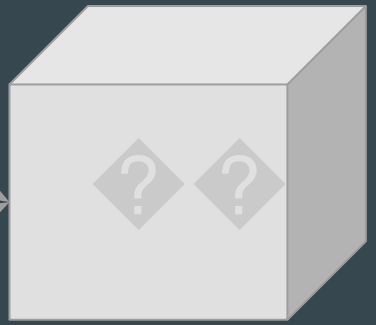
Where's waldo?



Overfitting

$h_i = \{H, T, H, T, H, T\}$

$h_j = \{H, H, H, H, H, H\}$

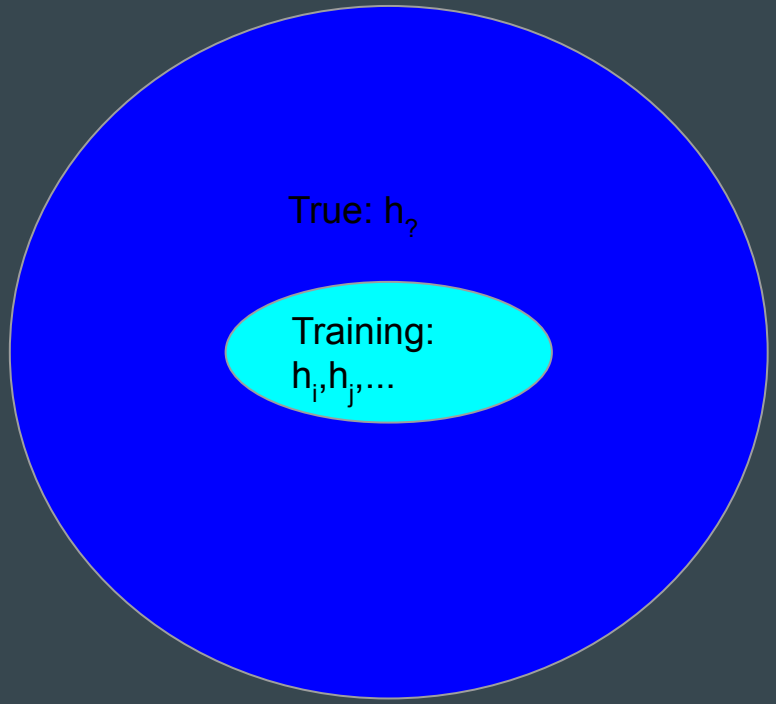


Training set:

Source: Free Vector Graphics

Bad guesses and more complex $\mathcal{H}$

What we want to know is how likely we are to guess the wrong thing?



$$P[h_{train} \neq h_?] =$$

$$P(h_i \neq h_?) \vee P(h_i \neq h_?) \vee \ldots$$

Things will obviously start to get out of hand as the complexity of H increases!

Intuition behind complex $\mathcal{H}$ and difficulty of learning

Chess

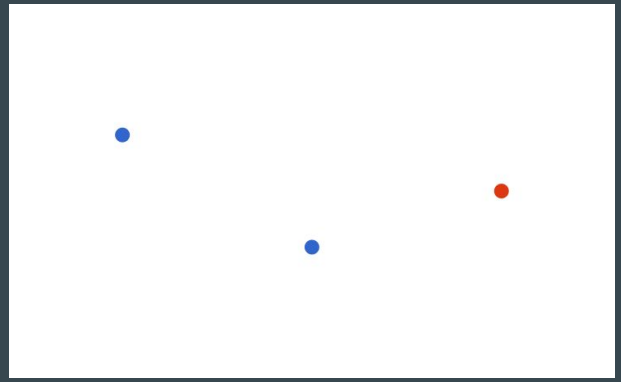
VS.

Topic Model

- Complicated models designed for messy target function and very large, sparse matrix
- Gold standard is human coded

How complex does $\mathcal{H}$ have to be?

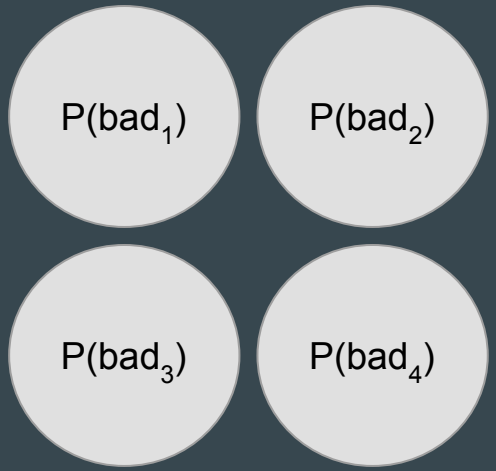
- What is $|\mathcal{H}|$ for a line?

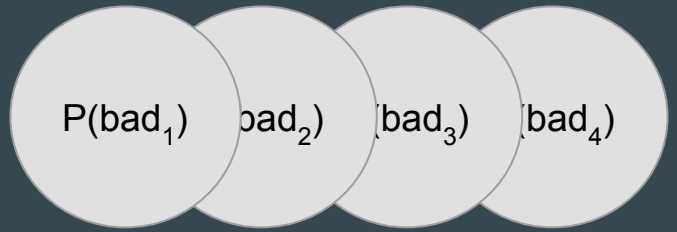


$$h_{TRUE} = w_{TRUE}x + b_{TRUE}$$

Effective number of hypotheses

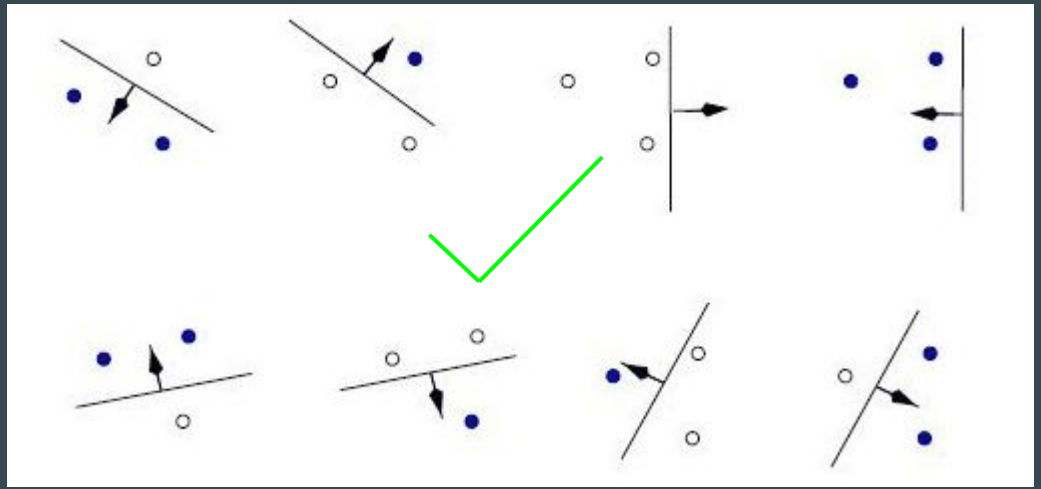
- Luckily, union bound is based on the unreasonable assumption of no overlap between our bad guesses, because there should be some pattern in the resampled data





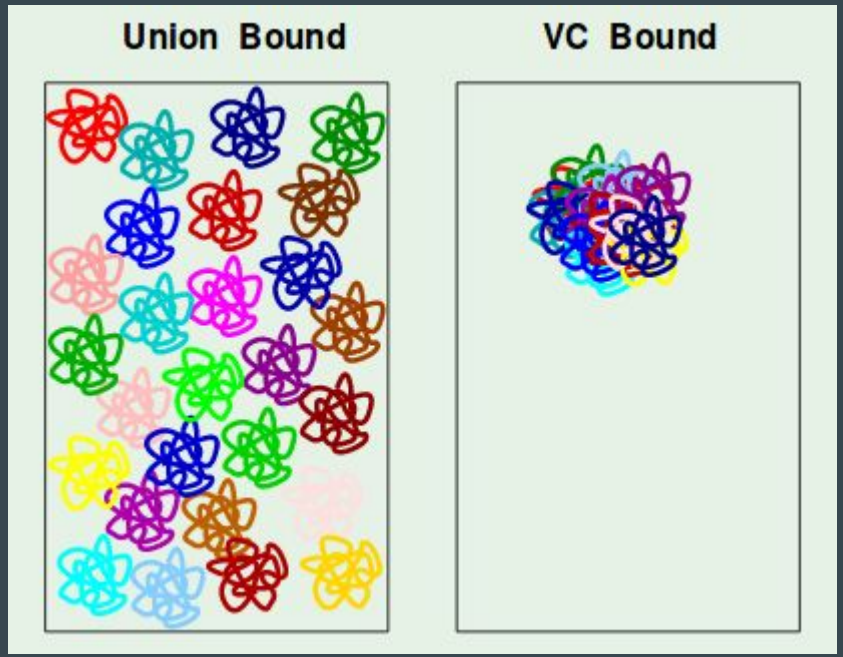
VC Dimension

- Using effective number of hypotheses, can we split our data into all possible combinations?



Source: Abhisek Ghose

VC Dimension



Source: Yaser Abu-Mustafa

VC Generalization

- The VC dimension expresses the *break point* of a hypothesis space
- This break point allows us to express |H| (infinite) as VC(H) (polynomial)
- Upshot: with enough data, we can learn g(x) from an infinite hypothesis space

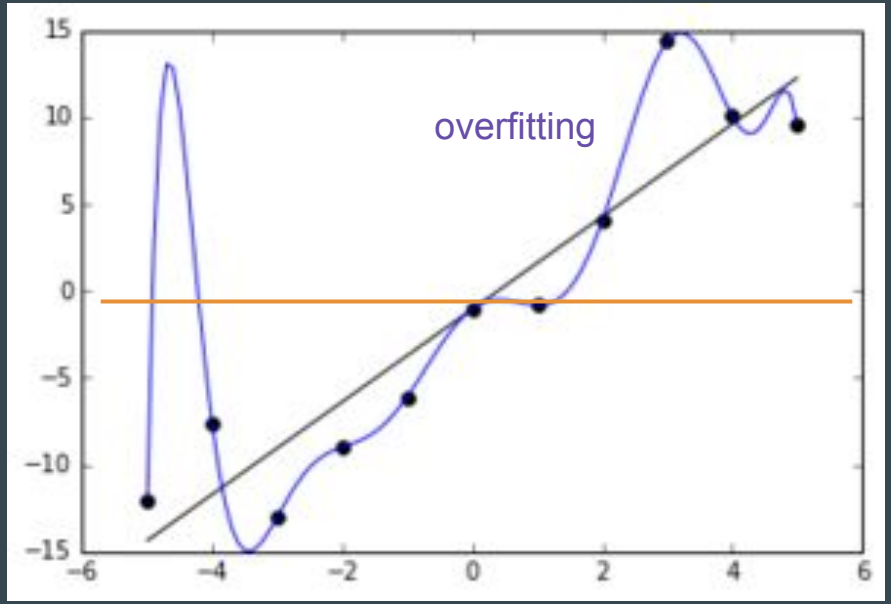
$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}}$$

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)\left(\ln \frac{2m}{VC(H)} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

Source: Dr. Carlos Guestrin

Bias-Variance Trade Off

Making bad guesses for g(x)



From wikipedia

Motivation: population and samples

- Looking for some true f(x)
- Try to find it by drawing a sample of the complete universe of X and y (population)
- For each sample (D) we find our best guess for the target function, g(x)
- Average for all g(x) is u
- *Variance:* $[g(x) - u]^2$
- *Bias:* $[u - f(x)]^2$

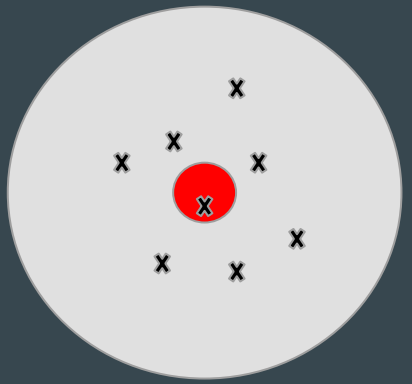
Bias and variance

Bias

- g(x) makes wrong predictions
- g(x) should be *flexible*
- Being really good at making really bad predictions

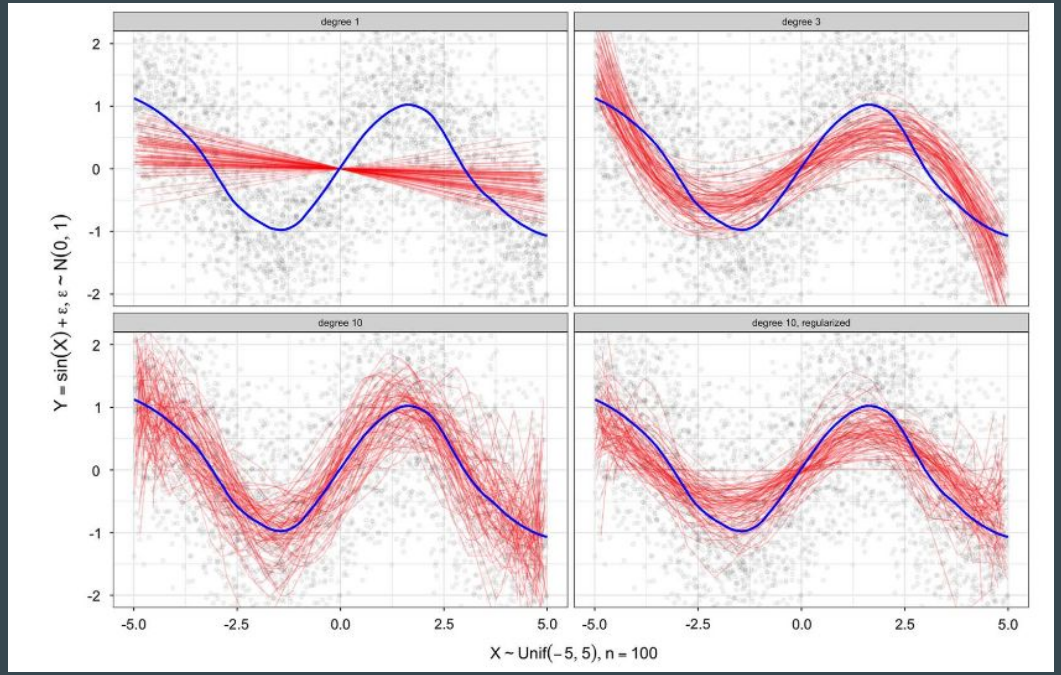
Variance

- g(x) captures too much noise
- g(x) should be *simple*
- Being really bad about making really good predictions



Bias and variance

High Bias + Variance



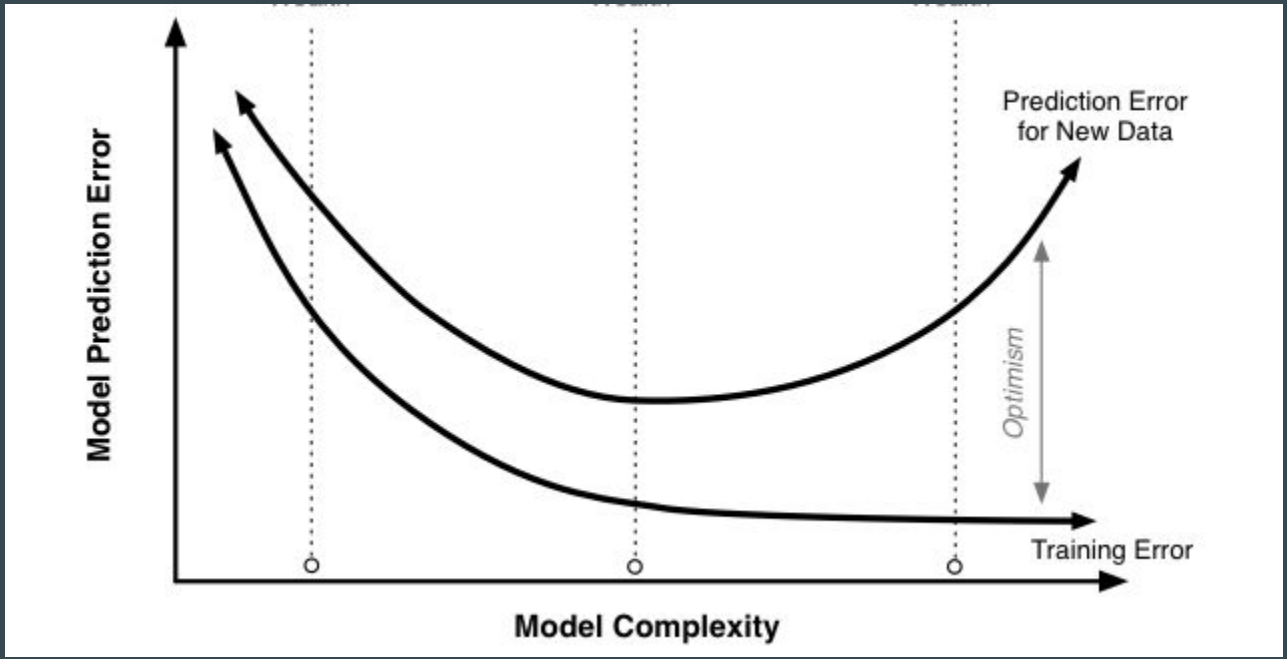
High Bias

High Variance

...just right!

Source: Jones and Fariss (2015)

The Danger Zone



Source: Damian Sowinski

Cost = loss + regularization

- How we manage the bias-variance tradeoff: tells us when too much fit is a bad thing

sum of squared errors: $\sum(\mathbf{w}^T\mathbf{x} - \hat{y})^2$

w/ regularization: $\sum(\mathbf{w}^T\mathbf{x} - \hat{y})^2 + \lambda||\mathbf{w}||^2$

- We know this intuitively. We need to find a way to shrink $\mathbf{w}$, or else we risk overfitting.

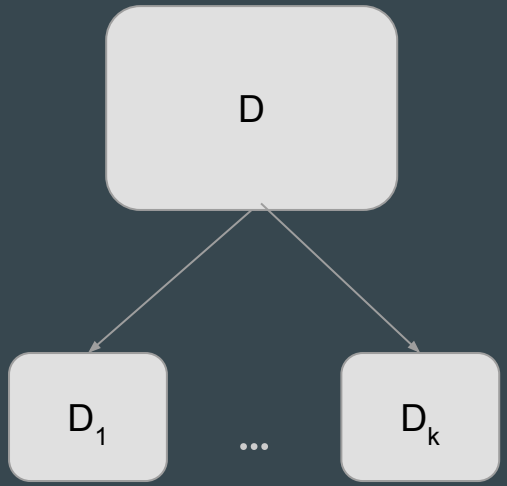
Common cost functions

Loss and Regularizer	Classification
1.Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}^\top x_i - y_i)^2$	• Squared Loss • No Regularization
2.Ridge Regression $\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}^\top x_i - y_i)^2 + \lambda \|w\|_2^2$	• Squared Loss • l_2-Regularization
3.Lasso $\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i - \overrightarrow{y_i})^2 + \lambda \|\mathbf{w}\|_1$	• + sparsity inducing (good for feature selection) • + Convex • - Not strictly convex (no unique solution) • - Not differentiable (at 0)
4.Logistic Regression $\min_{\mathbf{w},b} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i(\mathbf{w}^\top \mathbf{x}_i + b)})$	• Often l_1 or l_2 Regularized

- Most cost functions are used because of their “pliability”
- But there are many other that we can optimize without an analytical solution!

Source: Kilian Weinberger

Cross-validation



- Split data into k samples
- *Test*: one sub-sample
- *Training*: k-1 sub-samples
- Pool model estimates for each test set

So far...

- We have defined the goal of machine learning as minimizing *generalization*
- Intuitively, generalization is trade-off between flexibility and simplicity
- Shown this in two different cases
  - *Computationally*: as hypothesis space gets more complex, it is more likely to contain the correct hypothesis, but you are less likely to be able to find it
  - *Statistically*: models for a particular dataset are designed to balance informative and uninformative information (simplicity vs flexibility)
- There are two things we have to worry about
  - *Underfitting*: f(x) cannot be identified within the hypothesis (not flexible, or not enough data)
  - *Overfitting*: g(x) thinks that error in y is actually part of f(x)
- Use resampling methods (like cross-validation) to pick the best regularizer

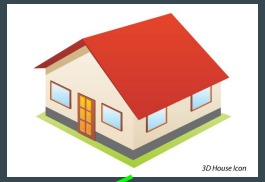
Learning = Representation + Evaluation + Optimization

1. Representation
   - Express real world phenomena as informative features and an outcome of interest
   - Choose a class of models to relate features and outcomes
2. Evaluation
   - *Cost function* helps determine which g(x) to pick
   - Regularization helps reduce generalization error
3. Optimization
   - Out of all the different representations, find the one that minimizes the total cost of all of our screw ups

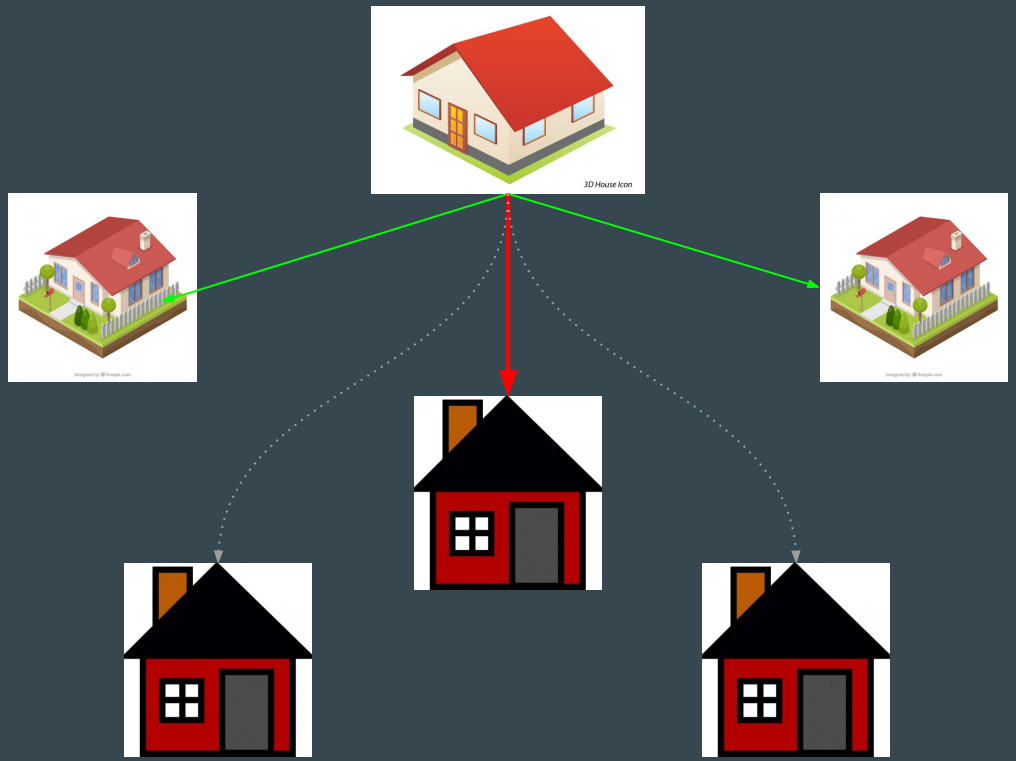
Source: Pedros Domingos

Nearest Neighbors

Should I get a lawn?



Should I get a lawn?

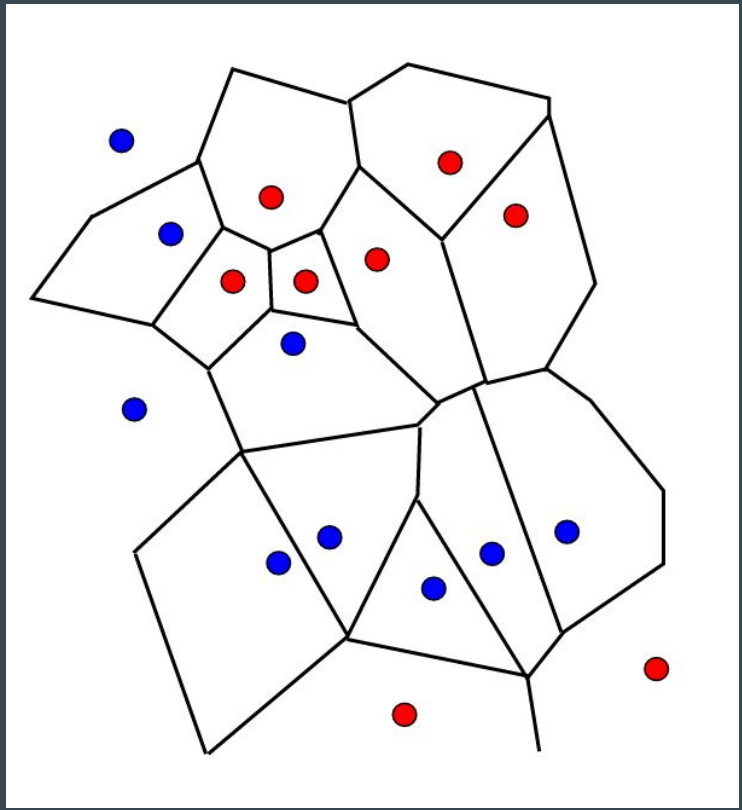


- Instead of just a single comparison, we rely on a neighborhood of size k
- The majority class of the nearest k neighbors wins!
- *Nearest*: defined by distance
- Obvious question: what should k be?

Finding k

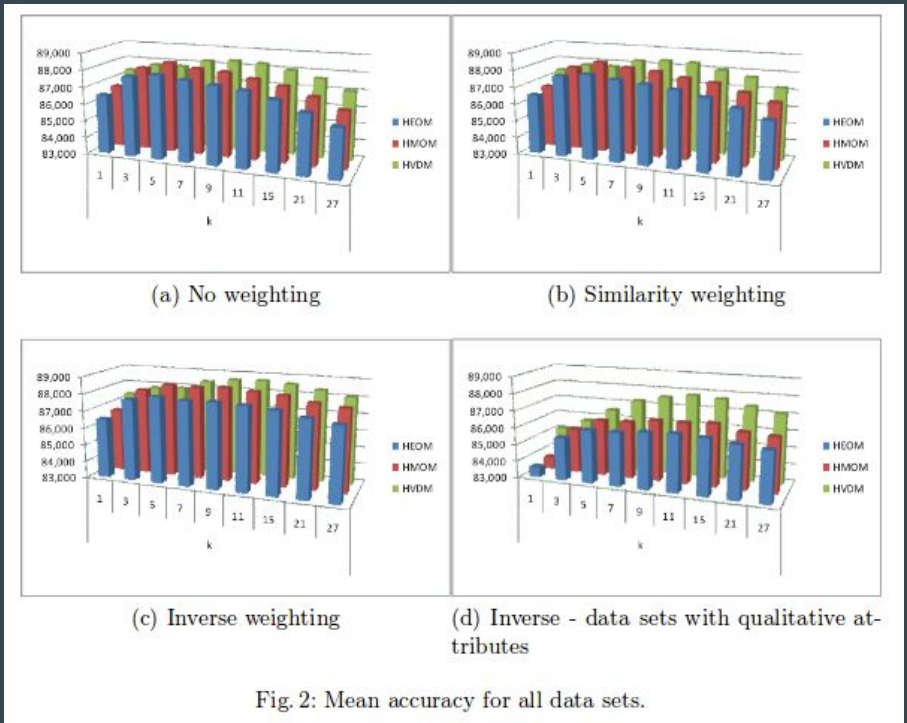
- A generalization problem:
  - *Underfitting*: one giant neighborhood
  - *Overfitting*: smallest possible k
- Use cross-validation or other resampling technique to find optimal k for test set
- This returns our best guess for g(x)
- Mimic the learning problem by evaluate this best guess on the test set
  - *Validation set* used to determine the best guesses

Complexity of $\mathcal{H}$ for nearest neighbors



- Hypothesis space for kNN is a voronoi diagram
- Complexity: $O(m^{N/2})$
- Difficulty with high-dimensional data

kNN extensions



(a) No weighting

(b) Similarity weighting

(c) Inverse weighting

(d) Inverse - data sets with qualitative attributes

Fig. 2: Mean accuracy for all data sets.

- k is not the only parameter you can tune
- Distance and weights can also be learned
- Highlights how we can increase flexibility without sacrificing simplicity

Source: Batistia and Silva (2009)

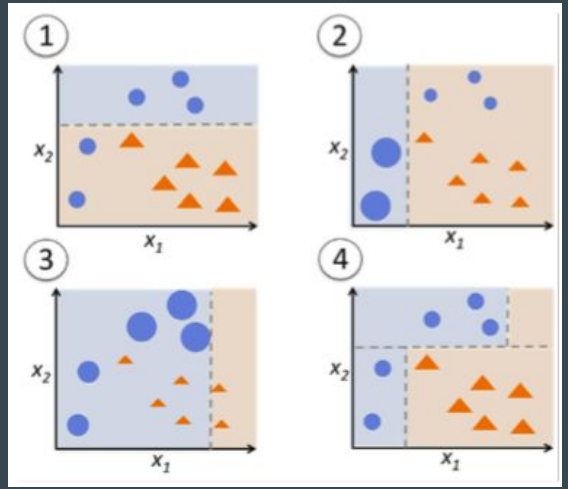
Ensemble Methods

Stumps

- We want to know if we need to bring an umbrella tomorrow (y: {0, 1})
- We base this off of whether it rained (1, 0) or what season it is (W, Sp, Su, F) (X: {rain, season})
- A stump randomly selects x_i and uses that to guess the label

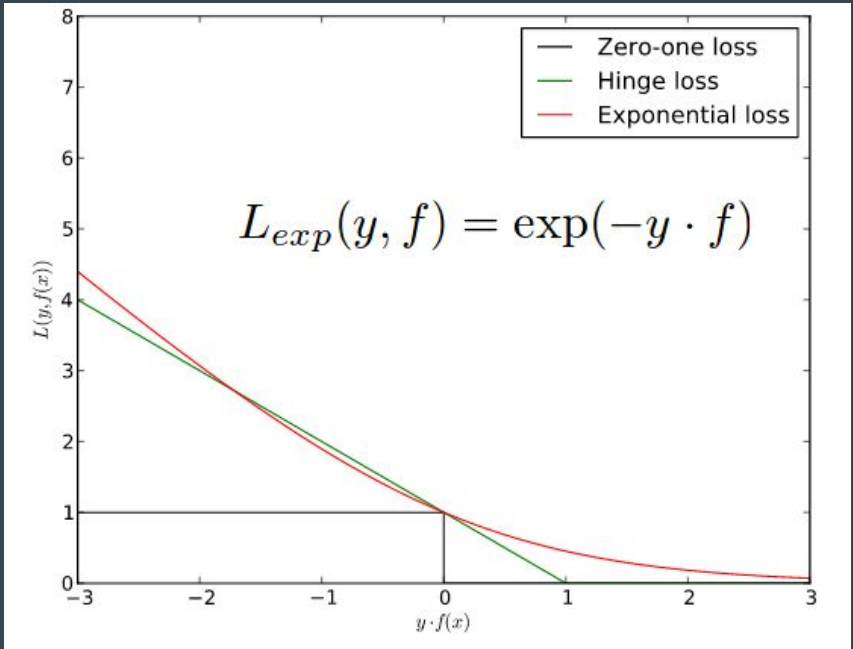
Boosting

- Uses *weak learners* (i.e. stumps)
  - As you can imagine, these generate a lot of errors!
- We weight these learners to penalize those errors
- Pick the optimal weight so as to maximize prediction for y



Source: Sebastian Rashka

Why boosting works



- Down-weights points consistently predicted correctly
- Up-weights points that the stumps have difficulty classifying
- Learns additive models

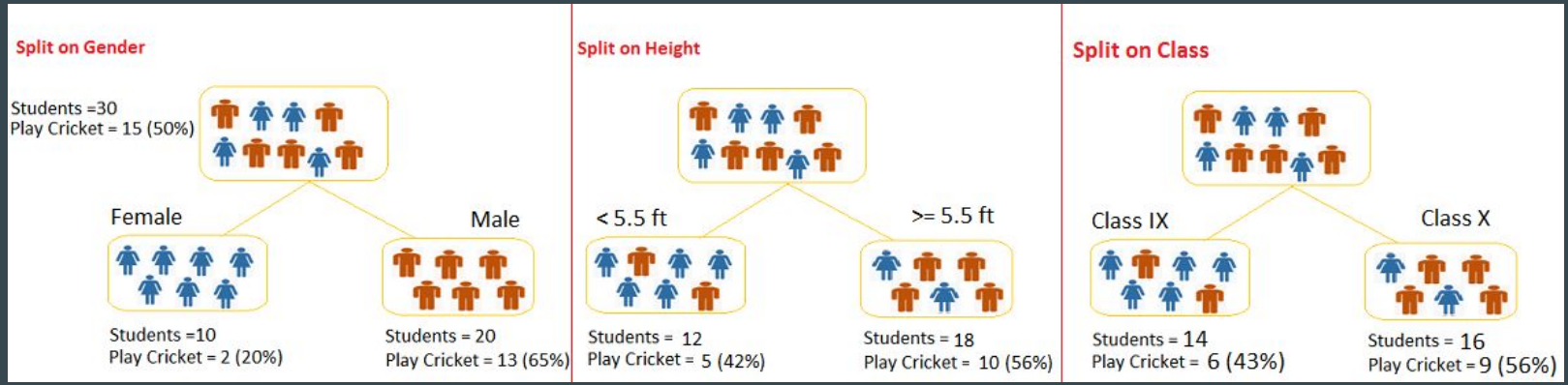
Source: Fabian Pedregosa

Trees

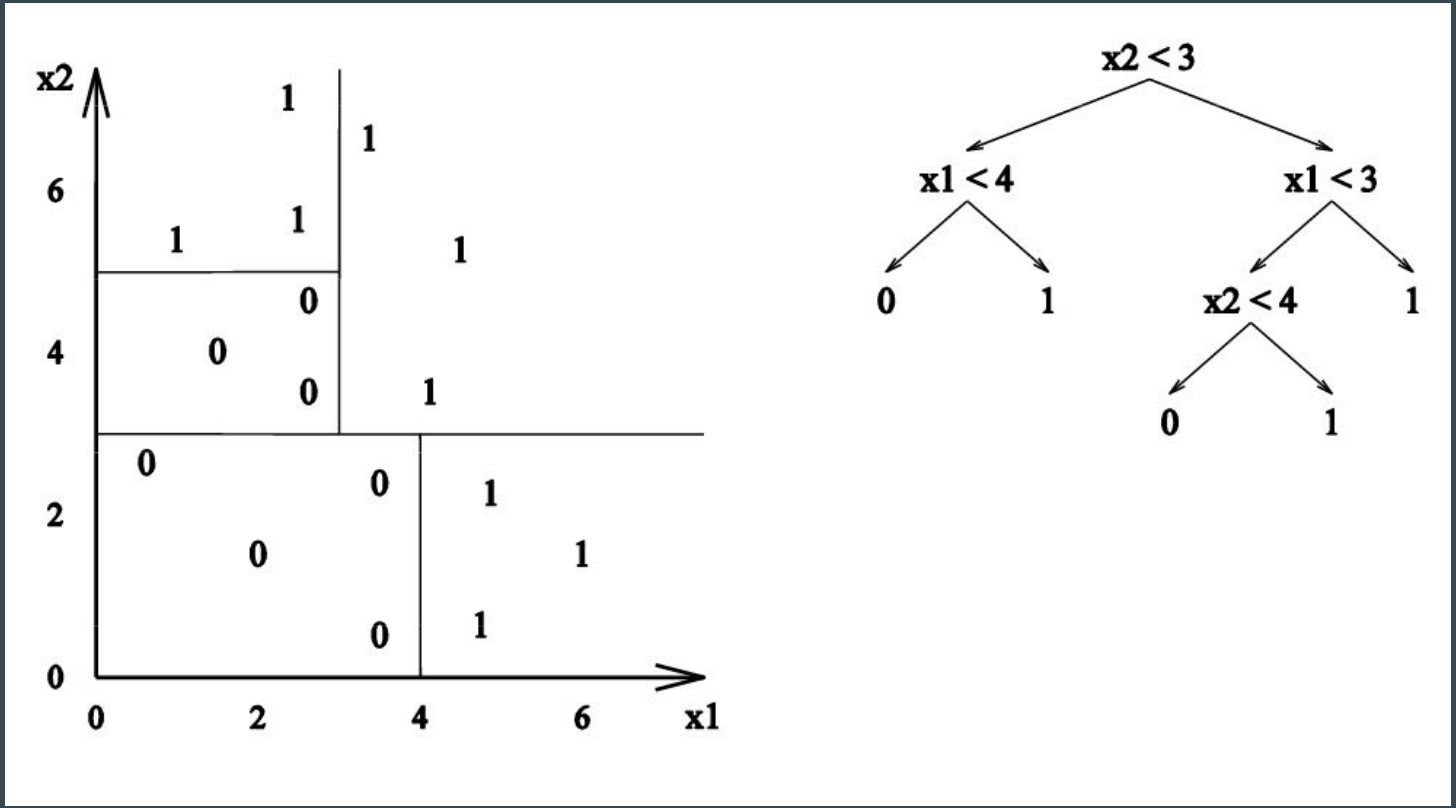
Extend stump to include possible partions of X, instead of picking based on x_i

Tree definitions

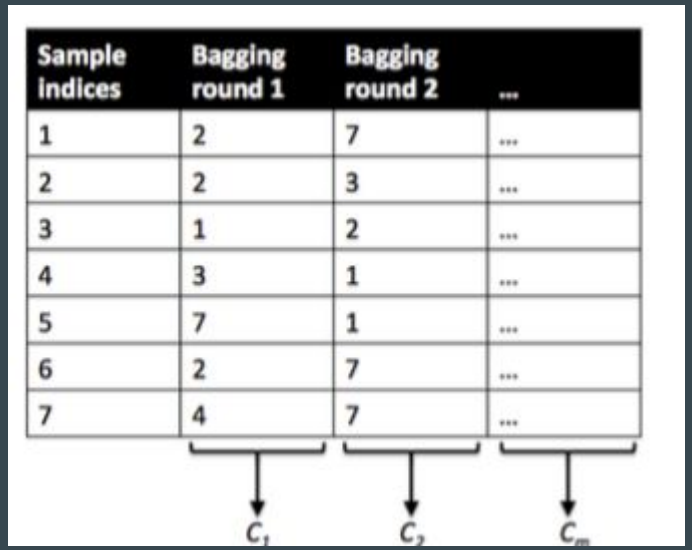
- *Root node:* beginning of tree
- *Leaf*: terminal mode
- *Decision node*: split generated for each x



Hypothesis space for trees



Bagging = bootstrap aggregating



Sample indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...

- Bootstrap training data
- Fit a decision tree for each bootstrapped sample
- Aggregate
  - Classification: majority vote
  - Regression: average prediction
- Bootstrapping is form of *resampling* to regularize our guess of g(x)

Source: Sebastian Rashka

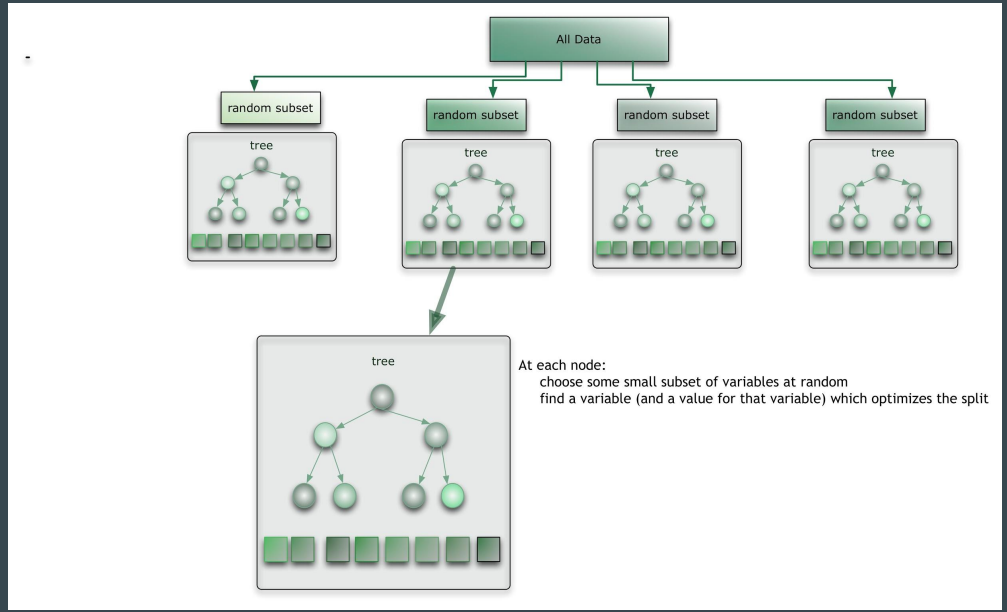
Combining decision trees

- Decision trees are great, but they require lots of, well... decisions
  - What features to use?
  - Where to split?
  - How deep should the tree go?
- Overfitting is easy
  - Maximum tree depth = perfect memorization
- Goal: find trees with lots of information about y
  - If certain splits provide lots of information (e.g., all large animals with black and white stripes are zebras), then we are *certain*
  - Use entropy as a measure of certainty

Random Forests

1. Randomly sample observations

2. From those, choose splits that minimize entropy

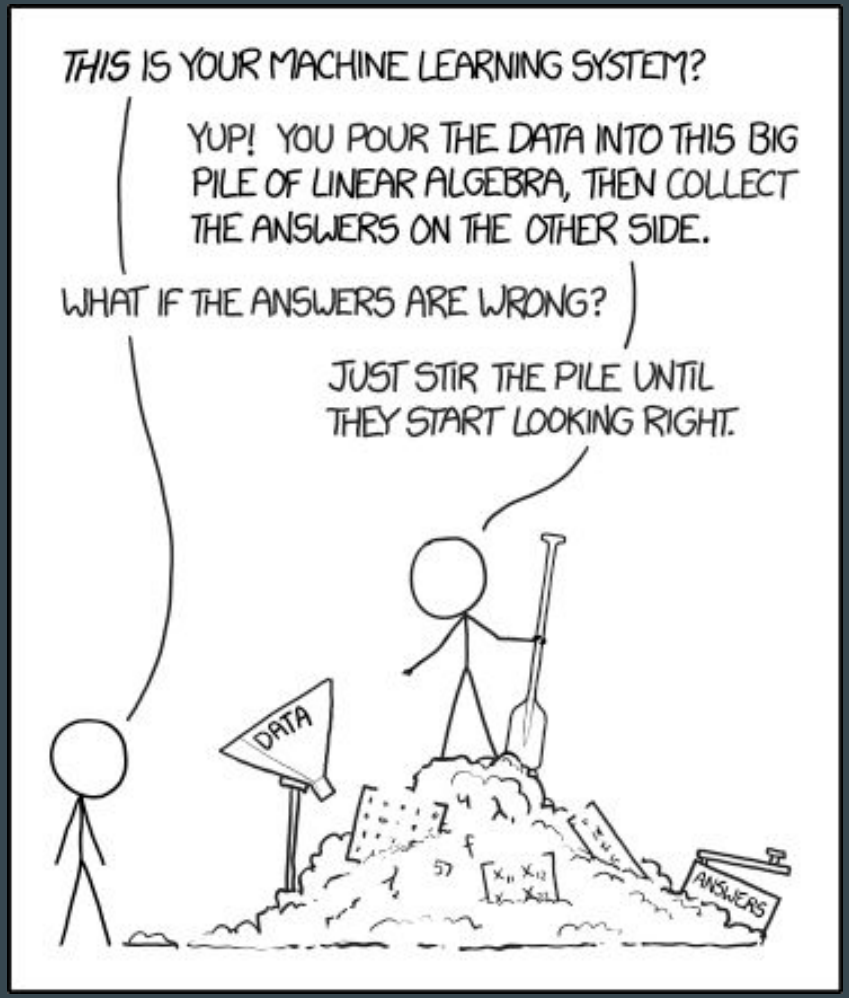


Source: Dan Benyamin

Why random forests are useful

- Combine useful properties of other ensemble learners
  - Random samples reduce variation (e.g. bagging)
  - Minimized cost function (entropy) reduces bias (e.g., boosting)
- Works for continuous and categorical representations
- Learns all sorts of model classes
  - Interactions
  - Non-linearities
- Resampling can be exploited for variable importance metrics
  - What happens when you exclude predictors as potential splits?
- Somewhat data intensive

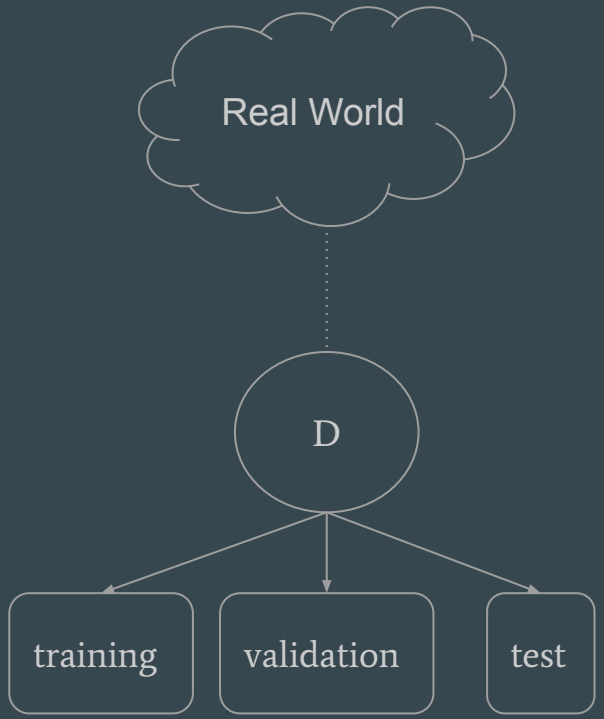
Picking the best models


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Problem: learning about learning

- Major unknown: how to measure generalization error induced by human decision making?
  - Hyperparemeters
  - Model class selection
- These problems aren't really different from others we have already solved
  - If you select only the best hyperparameters that work for a given dataset, you are overfitting
  - If you only look at model classes that might not contain f(x), you are underfitting

Restating the learning problem



- We want to do two things at once
  a. Minimize generalization error for our particular alogirthm
  b. Minimize generalization error for hyperparemeters
- Account for user generated error with usual generalization tools
  a. Find best g(x) for training data
  b. Adjust *model parameters* based on *validation* performance
  c. Adjust hyperparemeters based on *test* performance

Cross-validation

1. *Training* → *Validation*

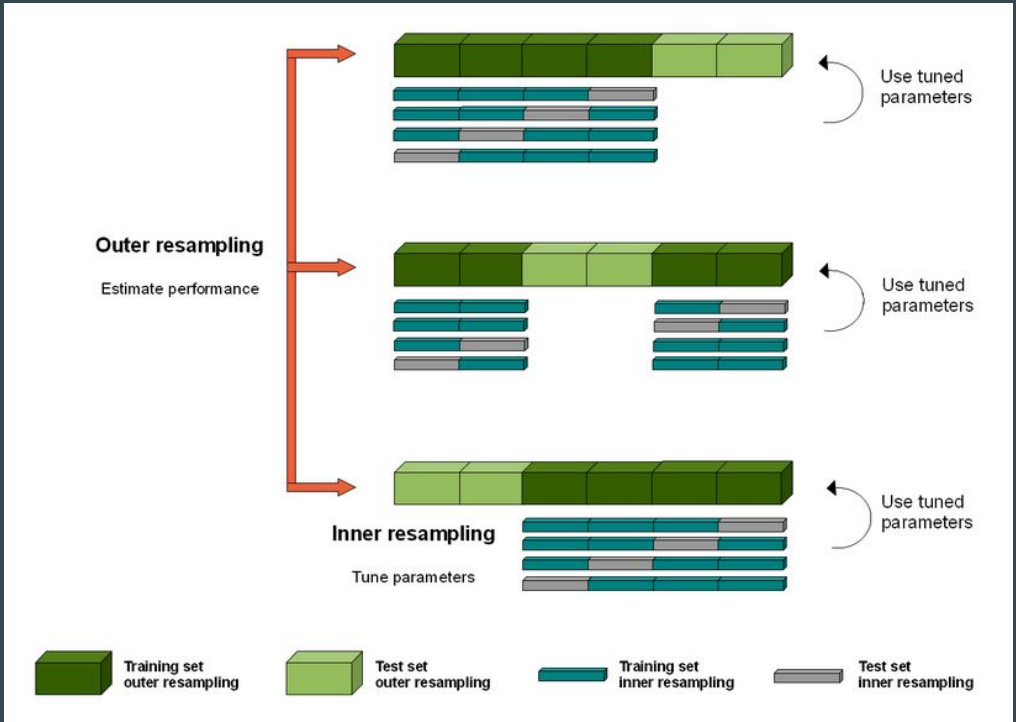
2. Combine training and validation

3. *Train + V* → *Test*

Cross-validation is one simple way of thinking about this

Source: Sebastian Raschka

Nested resampling



Many resampling strategies

- Bootstrap
- K-fold cross-validation
- Bagging
- Grid search
- Bayesian optimization
- Random search

Source: MLR

Realities of social data

Two problems for learning from social data

- Your data sucks!
  a. *Too much* of the wrong information
  b. *Not enough* of the right information
- The world is messy
  a. Complex models require flexible model classes
  b. Need to adjust for comparing different strategies

Your data sucks!

Not enough of the right information

- You are missing critical information that would explain f(x)
- In other words, f(x) could be outside of your hypothesis space

Too much of the wrong information

- Curse of dimensionality
- Need to fit simpler g(x)

To make things worse, these often happen together. Because you're lacking very important information, you end up including lots of features that you don't need.

The world is messy

Do we live in this world?

The world is messy

... or this one?

Weaknesses for conventional learning techniques

1. Require specifying interactions and polynomial terms
2. Misleading covariance from structural dependencies
3. Sensitive to noisy data
4. Pool results for multiple comparisons

Benefits of machine learning

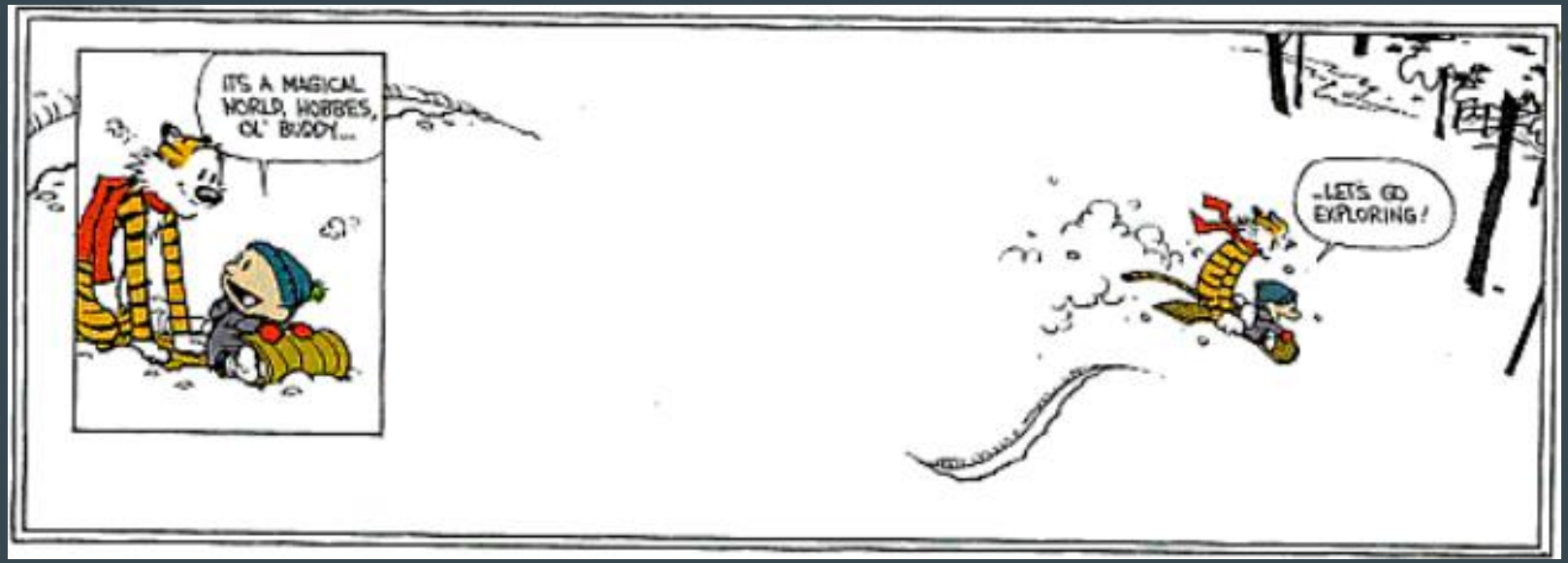
- Model classes have a *data adaptive* parameter selection.
  a. Interaction detection
  b. Ambiguous about functional form
- Dependency structure also learnable
- Allows adjustment for model selection

All of these properties handled by minimizing generalization error.

Learning = Representation + Evaluation + Optimization

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

Source: Pedro Domingos



Thanks for listening!