CS6510
Applied Machine Learning

Classifier System Design and Learning Theory

16 Sep 2017

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

Administrivia

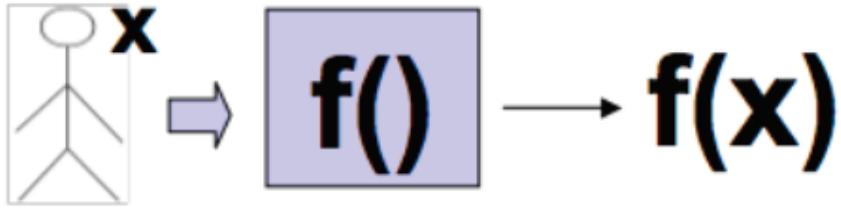
- Mid-semester Exam date
- Open vs Closed Marks System
- Audit Option

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization
- Bayes Error
- Introduction to Learning Theory

Data Handling Issues

- How to represent x depends on the application
- Attribute-value pairs
  - Example: x = {height=180cm, eyes="blue", job="student"}
- Unordered vs Ordered Attributes
- Attribute Types
  - Numerical: -3.14, 6E23, 0, 1
  - Categorical: Oranges, Apples, Lemons, Mangoes
  - Ordinal: Poor, Satisfactory, Good, Excellent



Pre-Processing

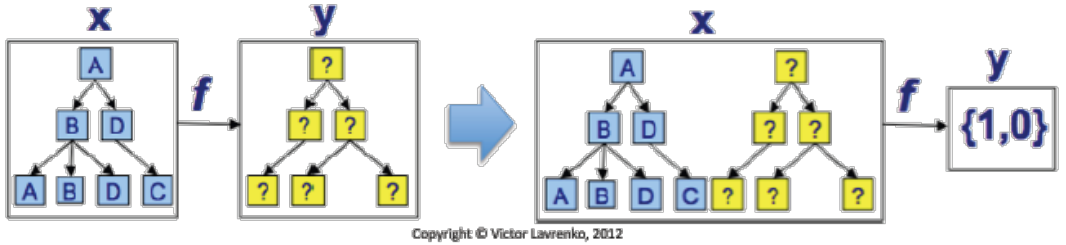
- Numeric
  - Normalization
    - Zero mean, unit variance: x'= (x - mean)/st.dev
    - In interval [0,1]: x'=(x - min)/(ma - min)
- Categorical
  - Usually encoded as numbers
    - Only equality testing is meaningful
- Ordinal
  - Encoded as numbers to preserve ordering
    - <=, >= operations meaningful

Feature Extraction from Data

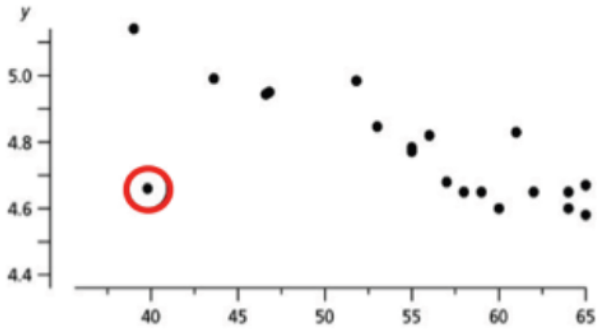
- Images
  - Pixel values, Segment and extract features, Handcrafted features: HOG, SIFT, etc
  - Deep learned features!
- Text
  - Bag of words, N-grams
  - Deep learned features!
- Speech
  - Mel Frequency Cepstral Co-efficients (MFCCs), Other frequency-based features
  - Deep learned features!
- Time-varying sensor Data
  - Statistical and moment-based features (mean, variance, etc)

Challenges

- Structured input/Structured output
  - One fix: Attribute = root-to-leaf paths



- Missing data
  - Fix: Fill in the value, Introduce special label, remove instance, remove attribute, Use classifiers that can handle missing values
- Outliers
  - Fix: Remove, Threshold, Visualize (we will see this later)!



- Data assumptions
  - Generated how? Sources?
  - Smooth? Linear? Noise?

Handling Imbalanced Data

- Assumption of ML methods in general: Data sets are balanced: i.e., there are as many positive examples of the concept as there are negative ones.
- Not true in practice!
  - ***Example:*** Our database of sick and healthy patients contains as many examples of sick patients as it does of healthy ones.
  - Many more examples
    - Helicopter Gearbox Fault Monitoring
    - Discrimination between Earthquakes and Nuclear Explosions
    - Document Filtering
    - Detection of Oil Spills
    - Detection of Fraudulent Telephone Calls

Handling Imbalanced Data

- Standard learners are often biased towards the majority class.
- That is because these classifiers attempt to reduce global quantities such as the error rate, not taking the data distribution into consideration.
- As a result, examples from the overwhelming class are well-classified whereas examples from the minority class tend to be misclassified.

Are all classifiers sensitive to class imbalance?

- **Decision Tree:** Very sensitive to class imbalances. This is because the algorithm works globally, not paying attention to specific data points.

- **Multi-Layer perceptrons (MLPs):** MLPs are less prone to the class imbalance problem. This is because of their flexibility: their solution gets adjusted by each data point in a bottom-up manner as well as by the overall data set in a top-down manner.

- **Support Vector Machines (SVMs)** SVMs are even less prone to the class imbalance problem than MLPs because they are only concerned with a few support vectors, the data points located close to the boundaries.

See Nathalie Japkowicz' work on "Inductive Learning from Imbalanced Datasets"

Handling Class Imbalance: Ideas

- Collect more data!
- Change your performance metric:
  - Confusion Matrix, Precision-Recall, F1-score, etc.
- Resample dataset
- Generate synthetic samples
- Try penalized models
- Try a different perspective – anomaly/change detection

http://machinelearningmastery.com/

Handling Class Imbalance

- At the data Level: Re-Sampling
  - Oversampling (Random or Directed)
  - Undersampling (Random or Directed)
  - Active Sampling
- At the Algorithmic Level:
  - Adjusting the Costs
  - Adjusting the decision threshold / probabilistic estimate at the tree leaf

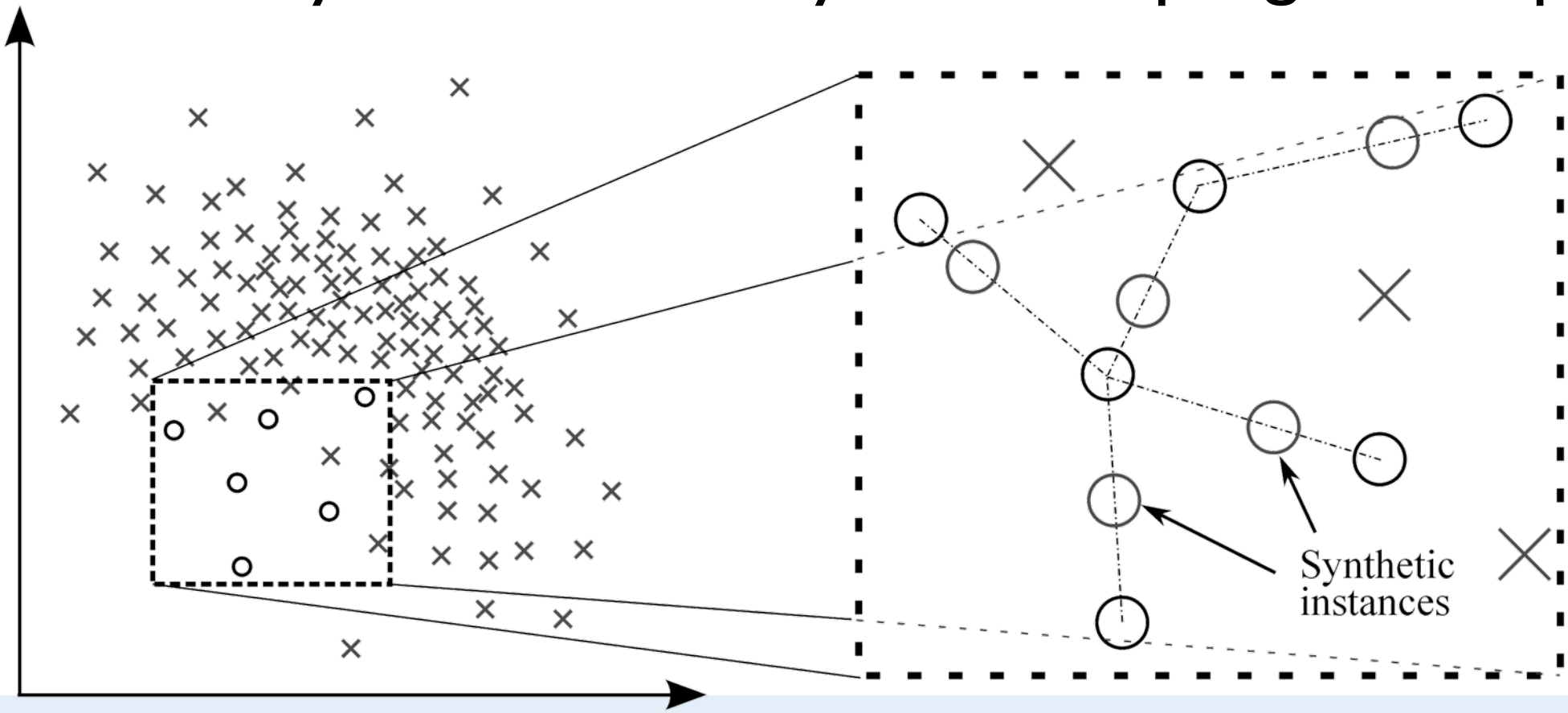
Handling Class Imbalance

- Undersampling (random and directed) is not effective and can even hurt performance.
- Random oversampling helps quite dramatically at all complexity. Directed oversampling makes a bit of a difference by helping slightly more.
- Cost-adjusting is about as effective as Directed oversampling. Generally, however, it is found to be slightly more useful.

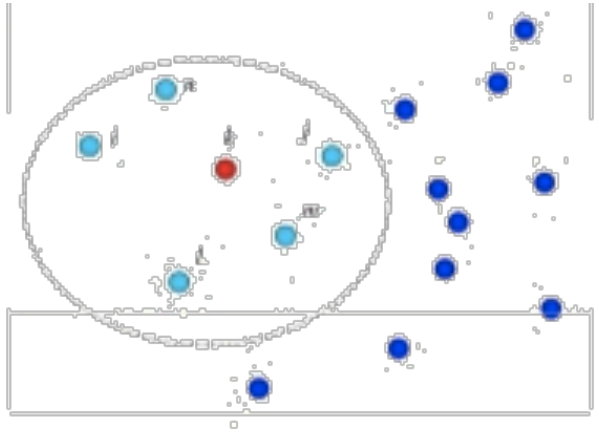
See Nathalie Japkowicz' work on "Inductive Learning from Imbalanced Datasets"

Example: SMOTE

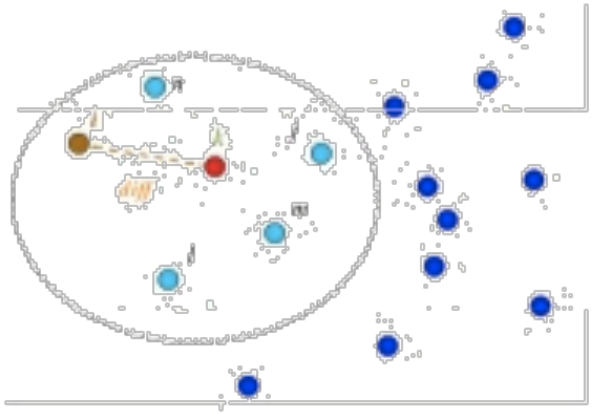
- SMOTE = Synthetic Minority Oversampling Technique



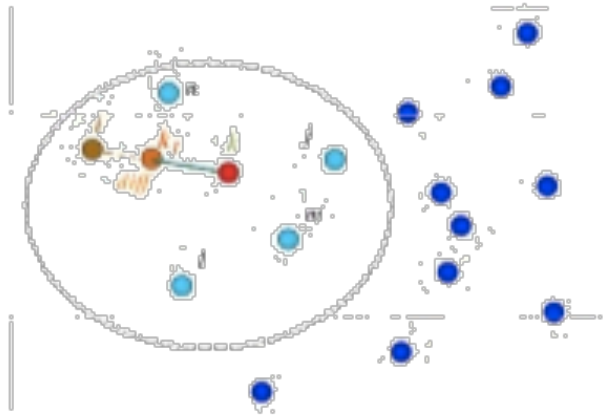
Example: SMOTE



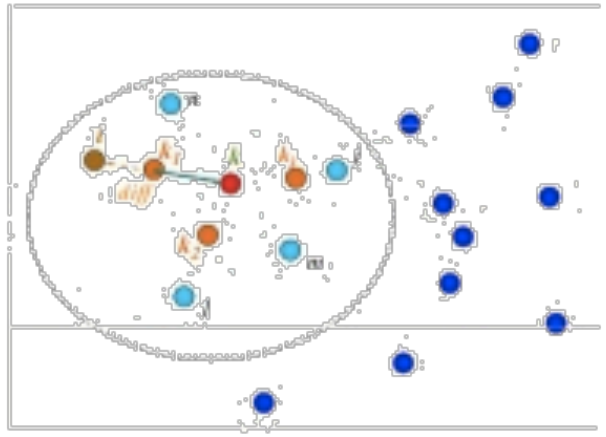
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)



2. Randomly choose an example out of 5 closest points



3. Synthetically generate event k_1, such that k_1 lies between k and i



4. Dataset after applying SMOTE 3 times

Using Large Datasets

- At large data scales, the performance of different algorithms converge such that performance differences virtually disappear.
- Given a large enough data set, the algorithm you'd want to use is the one that is computationally less expensive.
- It's only at smaller data scales that the performance differences between algorithms matter.
- More:
  - Data-Intensive Text Mining with MapReduce by Lin and Dyer
  - Mining of Massive Datasets by Rajaraman, Leskovec and Ullman

Using Large Datasets

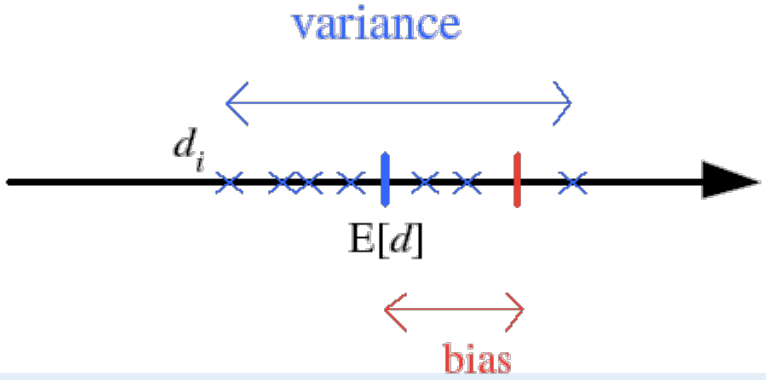
- CPUs vs GPUs
  - Deep learning has greatly benefited from GPUs
- Map-Reduce/Hadoop, Apache Spark, Vowpal Wabbit frameworks
  - Many learning algorithms amenable to partitioning of computations

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization
- Bayes Error
- Introduction to Learning Theory

Generalization Error

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/ simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other



Unknown parameter q
Estimator $f_i = f(X_i)$ on sample X_i

Bias: $b_q(f) = E[f] - q$
Variance: $E[(f - E[f])^2]$

Mean square error:
$r(f,q) = E[(f-q)^2]$
$= (E[f] - q)^2 + E[(f - E[f])^2] +$
Error term
$=$ Bias^2 + Variance + Error term

Bias-Variance Tradeoff

$$E(MSE) = \text{noise} + \text{bias}^2 + \text{variance}$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

See the following for mathematical derivation of bias-variance:

- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Bias-Variance Tradeoff

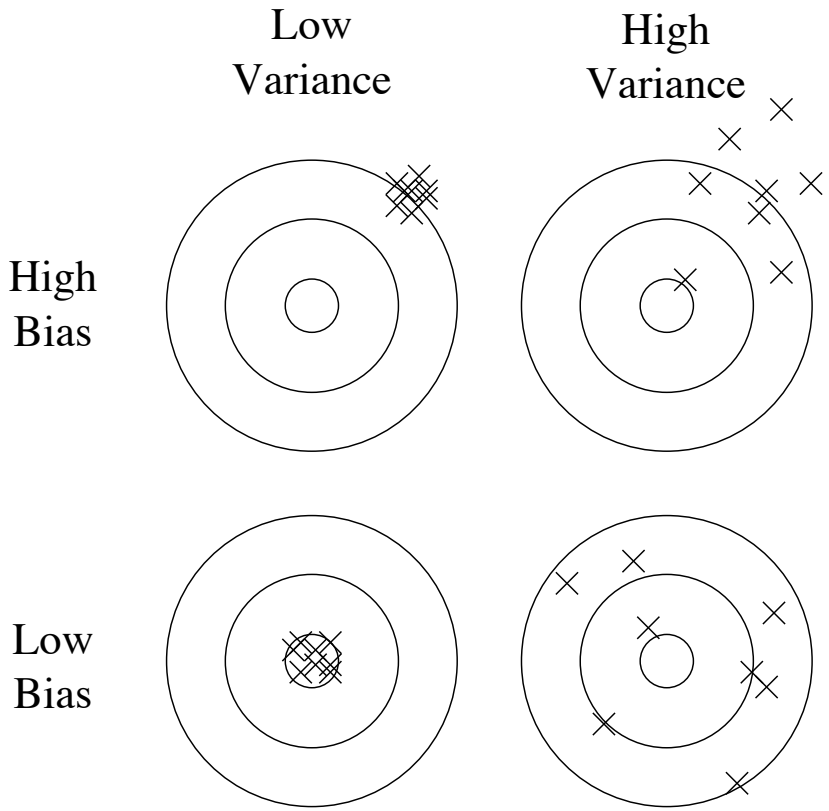
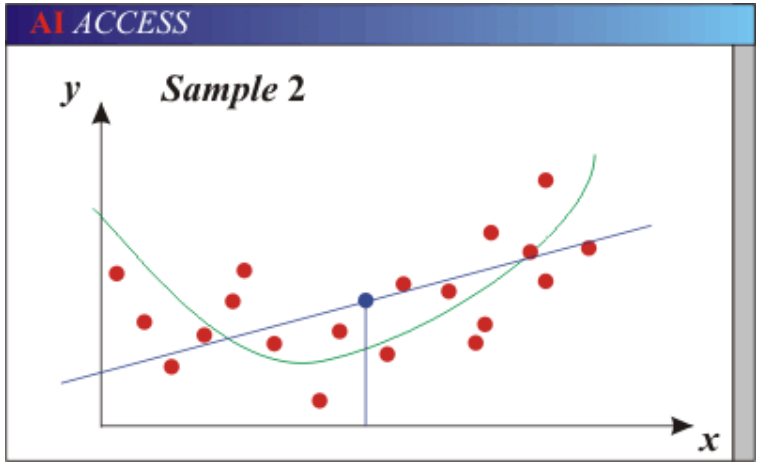
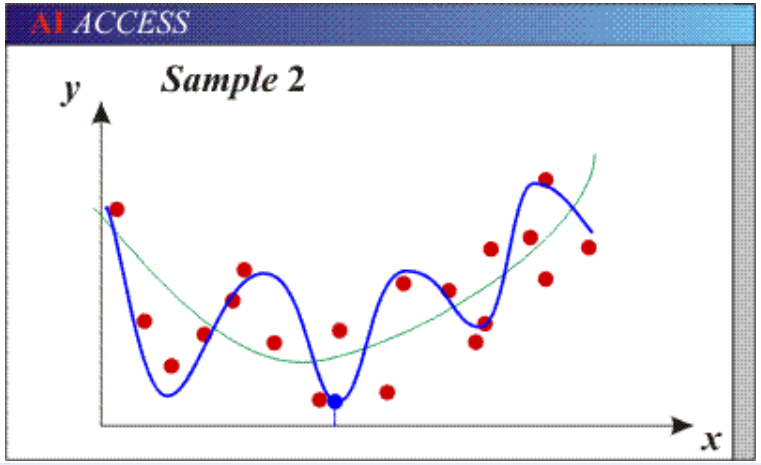


Figure 1: Bias and variance in dart-throwing.

Bias-Variance Tradeoff



- Models with too few parameters are inaccurate because of a **large bias** (not enough flexibility).



- Models with too many parameters are inaccurate because of a **large variance** (too much sensitivity to the sample).

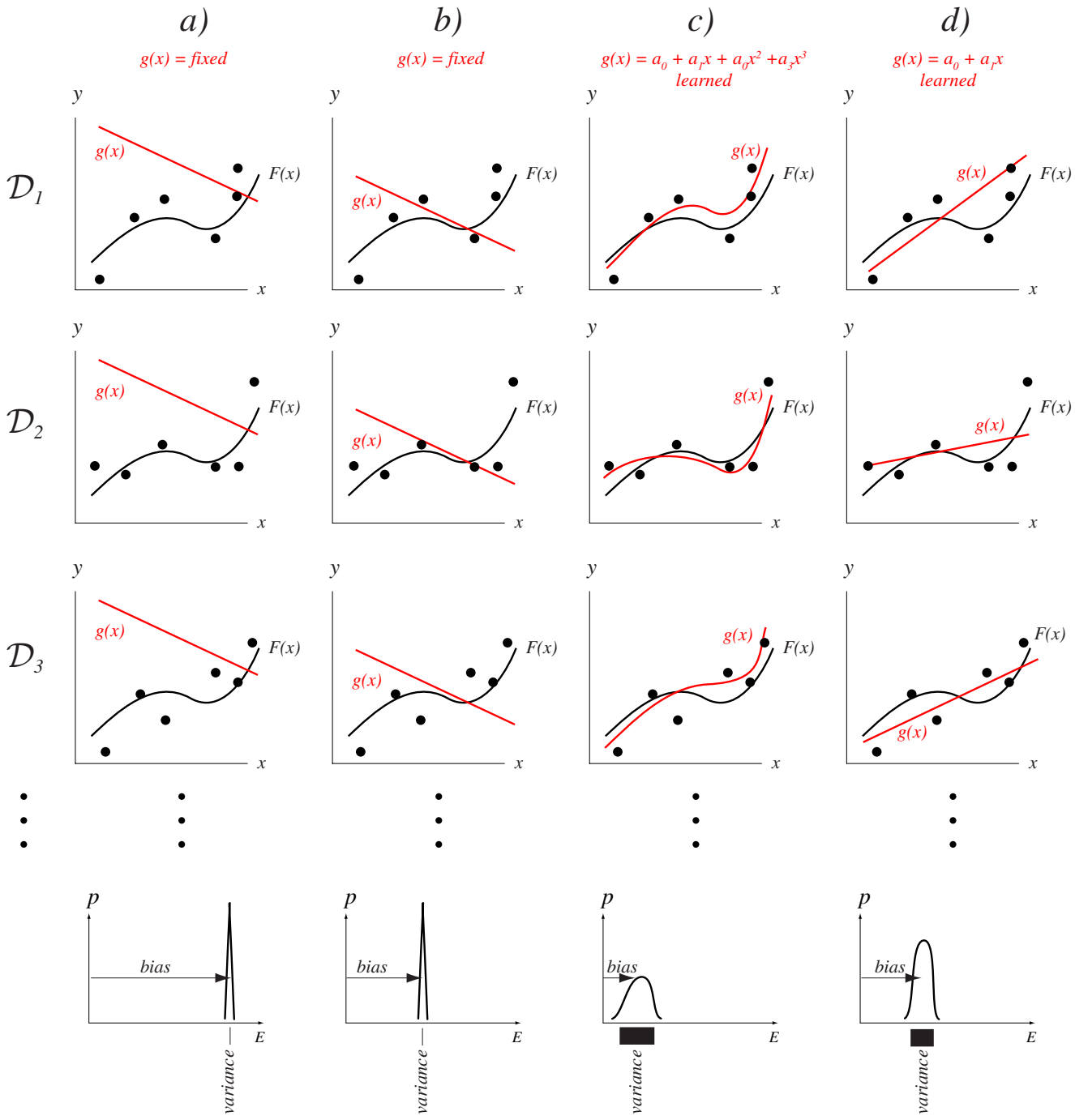
Slide Credit: D Hoiem

Generalization Error

- **Underfitting:** Model is too "simple" to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error

- **Overfitting:** Model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Bias-Variance Tradeoff



Classification

- Can do bias-variance analysis for classifiers as well.
- General inference: variance dominates bias.
- Very roughly, this is because we only need to make a discrete decision rather than get an exact value.

Measuring Bias and Variance

- In practice (unlike in theory), we have only ONE training set S.
- We can simulate multiple training sets by bootstrap replicates
  - S' = {x | x is drawn at random with replacement from S} and |S'| = |S|.
- Construct B bootstrap replicates of S (e.g., B = 200): $S_1, \ldots, S_B$
- Apply learning algorithm to each replicate S_b to obtain hypothesis h_b
- Let $T_b = S \setminus S_b$ be the data points that do not appear in S_b (out of bag points)
- Compute predicted value $h_b(x)$ for each x in T_b

Measuring Bias and Variance

- For each data point x, we will now have the observed corresponding value y and several predictions $y_1, \ldots, y_K$.
- Compute the average prediction $\underline{h}$.
- Estimate bias as $(\underline{h} - y)$
- Estimate variance as $\Sigma_k (y_k - \underline{h})^2/(K - 1)$
- Assume noise is 0
  - If we have multiple data points with the same x value, then we can estimate the noise – not generally available in machine learning
  - We can also estimate noise by pooling y values from nearby x values

Some Inferences

- How to reduce variance of classifier
  - Choose a simpler classifier
  - Regularize the parameters
  - Get more training data
- Training Error and Cross-Validation
  - Suppose we use the *training error* to estimate the difference between the true model prediction and the learned model prediction.
  - The training error is downward biased: on average it *underestimates* the generalization error.
  - Cross-validation is nearly unbiased; it slightly overestimates the generalization error.

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization
- Bayes Error
- Introduction to Learning Theory

Regularizers so far

- K-NN
  - Choose higher k
- Decision Trees
  - Pruning
- Naïve Bayes
  - Parametric models automatically act as regularizers
- SVMs?
- Neural Networks?

Model-based Machine Learning

1. pick a model

$$0 = b + \sum_{j=1}^{m} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} 1\left[y_i(w \cdot x_i + b) \leq 0\right]$$

3. develop a learning algorithm

$$\text{argmin}_{w,b} \sum_{i=1}^{n} 1\left[y_i(w \cdot x_i + b) \leq 0\right]$$

Find w and b that minimize the 0/1 loss

Slide Credit: David Kauchak, Pomona University

Regularization in Model-based ML

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

- Should we allow all possible weights? Any preferences?
- What makes for a "simpler" model for a linear model?
- Generally, we don't want huge weights
  - If weights are large, a small change in a feature can result in a large change in the prediction
  - Also, can give too much weight to any one feature
- Might also prefer weights of 0 for features that aren't useful
- How do we encourage small weights? or penalize large weights?

Slide Credit: David Kauchak, Pomona University

Regularization in Model-based ML

- A regularizer is an additional criteria to the loss function to make sure that we don't overfit
- It's called a regularizer since it tries to keep the parameters more normal/regular
- It is a bias on the model forces the learning to prefer certain types of weights over others

$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda \; regularizer(w,b)$$

Slide Credit: David Kauchak, Pomona University

Regularizers

$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda \, regularizer(w,b)$$

sum of the weights (1-norm) $r(w,b) = \sum_{w_j} |w_j|$

sum of the squared weights (2-norm) $r(w,b) = \sqrt{\sum_{w_j} |w_j|^2}$

---

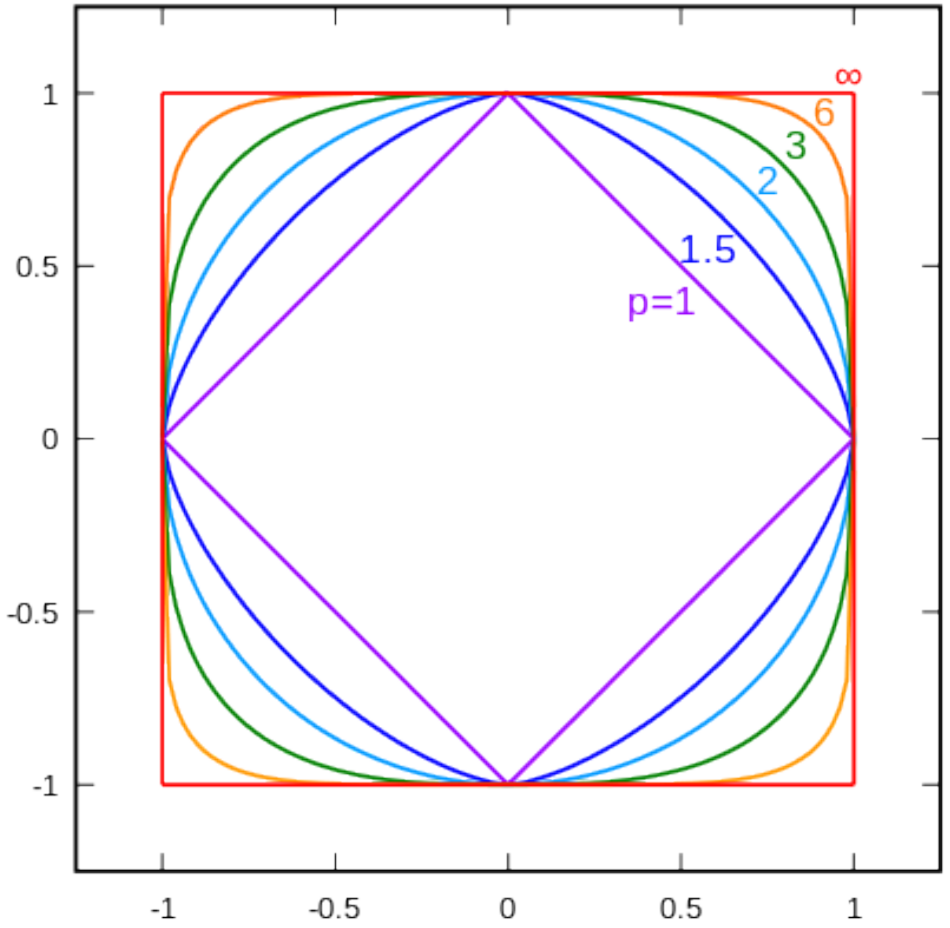
p-norm $r(w,b) = \sqrt[p]{\sum_{w_j} |w_j|^p} = \|w\|^p$

Smaller values of p (p < 2) encourage sparser vectors

Larger values of p discourage large weights more

Slide Credit: David Kauchak, Pomona University

L_p-Norm Regularizers: Visualization



all p-norms penalize larger weights

p < 2 tends to create sparse (i.e. lots of 0 weights)

p > 2 tends to like similar weights

Slide Credit: David Kauchak, Pomona University

L_p-Norm Regularizers: Summary

- L1 is popular because it tends to result in sparse solutions (i.e. lots of zero weights)
  - However, it is not differentiable, so it only works for gradient descent solvers
- L2 is also popular because for some loss functions, it can be solved directly (no gradient descent required, though often iterative solvers still)
- Lp is less popular since they don't tend to shrink the weights enough

Slide Credit: David Kauchak, Pomona University

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization
- Bayes Error
- Introduction to Learning Theory

Optimality of Bayes Decision Rule

- Let X be a *random variable* over the space Ω
- Two category decision problem:

$$H_1: X \in \omega_1$$

$$H_2: X \in \omega_2$$

- Optimal decision is:

$$\text{Choose } H_1 \text{ when } p(x|\omega_1)P(\omega_1) \geq p(x|\omega_2)P(\omega_2)$$

$$\text{Choose } H_2 \text{ when } p(x|\omega_1)P(\omega_1) < p(x|\omega_2)P(\omega_2)$$

Bayes Decision Rule

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

- Consider an arbitrary decision rule:
  - Partition Ω into two disjoint regions: $\mathcal{R}_1$ and $\mathcal{R}_2$

    Choose H_1 if $X \in \mathcal{R}_1$

    Choose H_2 if $X \in \mathcal{R}_2$
  - This decision rule is possibly non-optimal.

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

- Consider the Bayes decision rule:

Partition Ω into two disjoint regions: Ω_1 and Ω_2

$$\text{Choose } H_1 \text{ if } X \in \Omega_1$$

$$\text{Choose } H_2 \text{ if } X \in \Omega_2$$

Where:

$$\Omega_1 = \{x \in \Omega: p(x|\omega_1)P(\omega_1) \geq p(x|\omega_2)P(\omega_2)\}$$

$$\Omega_2 = \{x \in \Omega: p(x|\omega_1)P(\omega_1) < p(x|\omega_2)P(\omega_2)\}$$

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

- Two types of error: one for each of the decision rules
- For the arbitrary decision rule:

$$P(error) = P(X \in \mathcal{R}_1, \omega_2) + P(X \in \mathcal{R}_2, \omega_1)$$

$$= P(X \in \mathcal{R}_1 | \omega_2) P(\omega_2) + P(X \in \mathcal{R}_2 | \omega_1) P(\omega_1)$$

$$= \int_{\mathcal{R}_1} p(x|\omega_2) P(\omega_2) dx + \int_{\mathcal{R}_2} p(x|\omega_1) P(\omega_1) dx$$

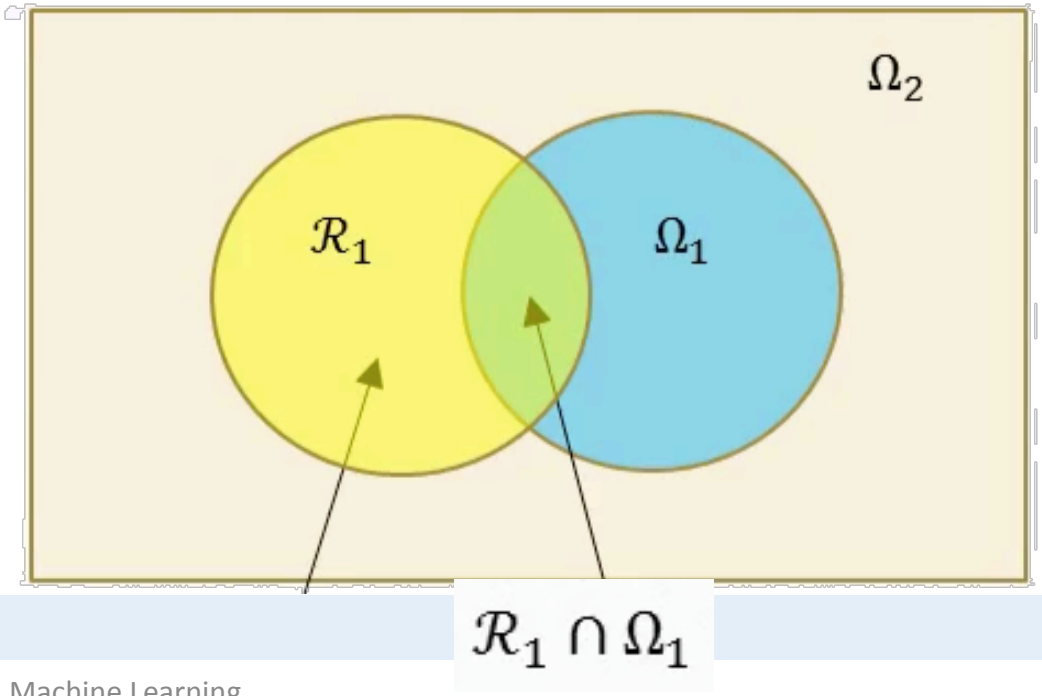
- For the Bayesian decision rule

$$P(error_{Bayes}) = \int_{\Omega_1} p(x|\omega_2) P(\omega_2) dx + \int_{\Omega_2} p(x|\omega_1) P(\omega_1) dx$$

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

- Let $\Delta(error) \equiv P(error) - P(error_{Bayes})$
- First recall that both $\mathcal{R}_1, \mathcal{R}_2$ and Ω_1, Ω_2 are partitions of Ω
  - $\mathcal{R}_1 = (\mathcal{R}_1 \cap \Omega_1) \cup (\mathcal{R}_1 \cap \Omega_2)$
  - $\mathcal{R}_2 = (\mathcal{R}_2 \cap \Omega_1) \cup (\mathcal{R}_2 \cap \Omega_2)$
  - $\Omega_1 = (\Omega_1 \cap \mathcal{R}_1) \cup (\Omega_1 \cap \mathcal{R}_2)$
  - $\Omega_2 = (\Omega_2 \cap \mathcal{R}_1) \cup (\Omega_2 \cap \mathcal{R}_2)$



Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

$$\Delta(error) = \int_{\mathcal{R}_1} p(x|\omega_2)P(\omega_2)dx + \int_{\mathcal{R}_2} p(x|\omega_1)P(\omega_1)dx - \int_{\Omega_1} p(x|\omega_2)P(\omega_2)dx - \int_{\Omega_2} p(x|\omega_1)P(\omega_1)dx$$

$$= P(\omega_2)\left[\int_{\mathcal{R}_1} p(x|\omega_2)dx - \int_{\Omega_1} p(x|\omega_2)dx\right] + P(\omega_1)\left[\int_{\mathcal{R}_2} p(x|\omega_1)dx - \int_{\Omega_2} p(x|\omega_1)dx\right]$$

$$= P(\omega_2)\left[\int_{\mathcal{R}_1 \cap \Omega_2} p(x|\omega_2)dx - \int_{\Omega_1 \cap \mathcal{R}_2} p(x|\omega_2)dx\right] + P(\omega_1)\left[\int_{\mathcal{R}_2 \cap \Omega_1} p(x|\omega_1)dx - \int_{\Omega_2 \cap \mathcal{R}_1} p(x|\omega_1)dx\right]$$

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

$$= \int_{\mathcal{R}_1 \cap \Omega_2} [p(x|\omega_2)P(\omega_2) - p(x|\omega_1)P(\omega_1)]dx + \int_{\Omega_1 \cap \mathcal{R}_2} [p(x|\omega_1)P(\omega_1) - p(x|\omega_2)P(\omega_2)]dx$$

Recall,

$$\Omega_1 = \{x \in \Omega : p(x|\omega_1)P(\omega_1) \geq p(x|\omega_2)P(\omega_2)\} \Rightarrow \int_{\mathcal{R}_2 \cap \Omega_1} [p(x|\omega_1)P(\omega_1) - p(x|\omega_2)P(\omega_2)]dx \geq 0$$

$$\Omega_2 = \{x \in \Omega : p(x|\omega_2)P(\omega_2) > p(x|\omega_1)P(\omega_1)\} \Rightarrow \int_{\mathcal{R}_1 \cap \Omega_2} [p(x|\omega_2)P(\omega_2) - p(x|\omega_1)P(\omega_1)]dx \geq 0$$

$$\Delta(error) \geq 0$$

Slide Credit: slecture of Aaron Michaux, Purdue University

Optimality of Bayes Decision Rule

$$\Delta(error) = P(error) - P(error_{Bayes}) \geq 0$$

$$\Leftrightarrow P(error) \geq P(error_{Bayes})$$

Recall:

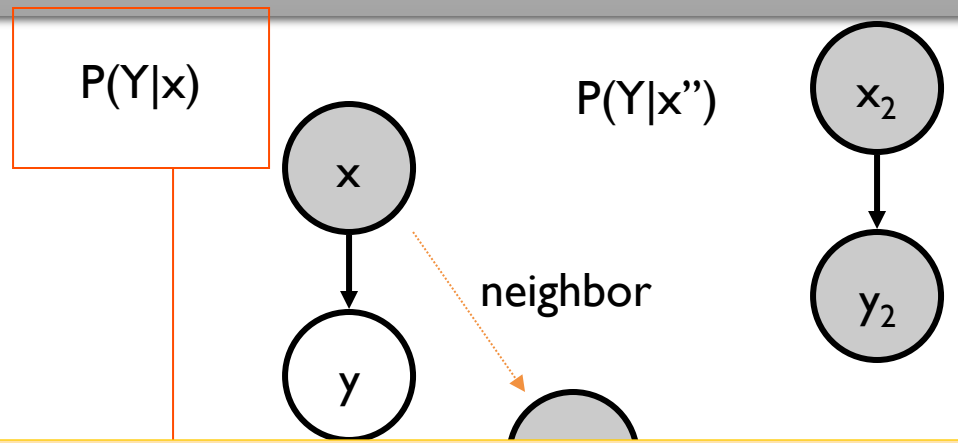
$$\text{Choose } H_1 \text{ when } p(x|\omega_1)P(\omega_1) \geq p(x|\omega_2)P(\omega_2)$$

$$\text{Choose } H_2 \text{ when } p(x|\omega_1)P(\omega_1) < p(x|\omega_2)P(\omega_2)$$

Slide Credit: slecture of Aaron Michaux, Purdue University

Recall: Convergence of 1-NN

$$P(\text{knnError})$$
$$= 1 - \Pr(y = y_1)$$
$$= 1 - \sum \Pr(Y = y' | x)^2$$



Possible to show that: as the size of training data set approaches infinity, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). We will see this later.

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization
- Bayes Error
- Introduction to Learning Theory

Computational Learning Theory: Overview

- Are there general laws that govern learning?
  - ***Sample Complexity:*** How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
  - ***Computational Complexity:*** How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
  - ***Mistake Bound:*** How many training examples will the learner misclassify before converging to a successful hypothesis?
- These questions can be answered within two analytical frameworks:
  - The ***Probably Approximately Correct (PAC)*** framework
  - The ***Mistake Bound*** framework

Ref: Chapter 7 of Machine Learning by Tom Mitchell

Computational Learning Theory: Overview

- Rather than answering these questions for individual learners, we will answer them for broad classes of learners. In particular we will consider:
  - The size or complexity of the hypothesis space considered by the learner.
  - The accuracy to which the target concept must be approximated.
  - The probability that the learner will output a successful hypothesis.
  - The manner in which training examples are presented to the learner.

Ref: Chapter 7 of Machine Learning by Tom Mitchell

PAC Learning Model

- **Definition:** Consider a concept class ***C*** defined over a set of instances ***X*** of length ***n*** and a learner ***L*** using hypothesis space ***H***. ***C*** is ***PAC-learnable*** by ***L*** using ***H*** if for all $c \in C$, distributions ***D*** over ***X***, ε such that $0 < \varepsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner ***L*** will, with probability at least $(1-\delta)$, output a hypothesis $h \in H$ such that $error_D(h) \leq \varepsilon$, in time that is ***polynomial*** in $1/\varepsilon$, $1/\delta$, n, and ***size(c)***.

Ref: Chapter 7 of Machine Learning by Tom Mitchell

Sample Complexity for Finite Hypothesis Spaces

- Given any ***consistent*** learner (commits zero errors over training data), the number of examples sufficient to assure that any hypothesis will be probably (with probability $(1-\delta)$) approximately (within error ε) correct is:

$$m = 1/\varepsilon \, (\ln|H| + \ln(1/\delta))$$

- If the learner is ***not consistent***

$$m = 1/2\varepsilon^2 \, (\ln|H| + \ln(1/\delta))$$

Ref: Chapter 7 of Machine Learning by Tom Mitchell

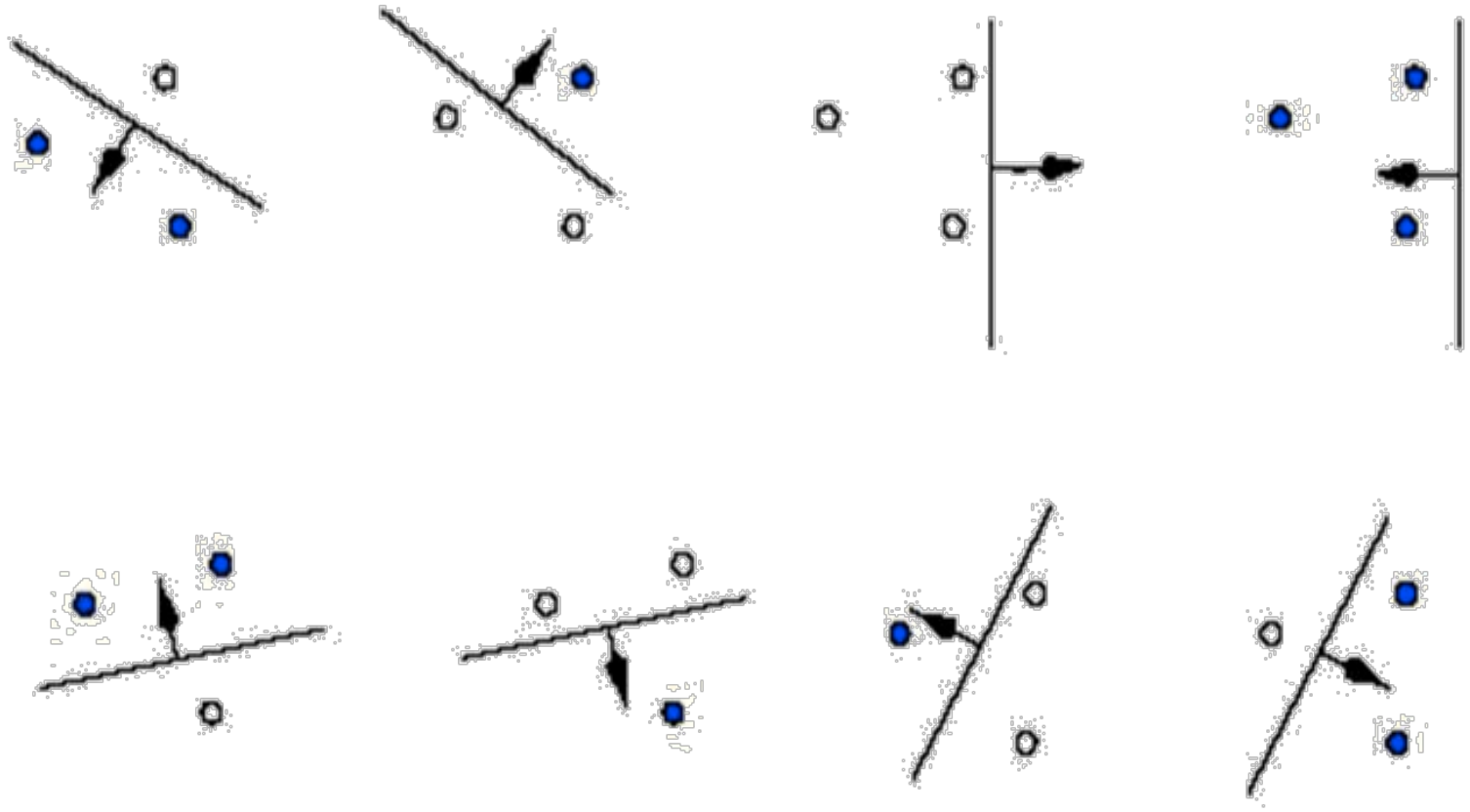
Sample Complexity for Infinite Hypothesis Spaces

- The PAC Learning framework has 2 disadvantages:
  - It can lead to weak bounds
  - Sample Complexity bound cannot be established for infinite hypothesis spaces
- There are other ideas for dealing with these problems:
  - **Definition:** A set of instances S is shattered by hypothesis space H iff for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.
  - **Definition:** The **Vapnik-Chervonenkis dimension**, VC(H), of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H)=\infty$

Ref: Chapter 7 of Machine Learning by Tom Mitchell

VC-Dimension

- VC-Dimension of set of oriented hyperplanes in R^n is (n+1)



Ref: Chapter 7 of Machine Learning by Tom Mitchell

Sample Complexity for Infinite Hypothesis Spaces

- ***Upper-Bound*** on sample complexity, using the ***VC-Dimension***

$$m \geq 1/\varepsilon \,(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon))$$

- ***Lower Bound*** on sample complexity, using the ***VC-Dimension***:

  Consider any concept class ***C*** such that ***VC(C) ≥ 2,*** any learner ***L,*** and any $0 < \varepsilon < 1/8$, and $0 < \delta < 1/100$. Then there exists a distribution ***D*** and target concept in ***C*** such that if ***L*** observes fewer examples than $\max[1/\varepsilon \log(1/\delta), (VC(C)-1)/(32\varepsilon)]$, then with probability at least δ, ***L*** outputs a hypothesis ***h*** having $error_D(h) > \varepsilon$.

Ref: Chapter 7 of Machine Learning by Tom Mitchell

VC Dimension for Neural Networks

- Let G be a layered directed acyclic graph with n input nodes and $s \geq 2$ internal nodes, each having at most r inputs. Let C be a concept class over $\mathbf{R}^r$ of VC dimension d, corresponding to the set of functions that can be described by each of the s internal nodes. Let C_G be the G-composition of C, corresponding to the set of functions that can be represented by G. Then $VC(C_G) \leq 2ds \log(es)$, where e is the base of the natural logarithm.
- This theorem can help us bound the VC-Dimension of a neural network and thus, its sample complexity (See, [Mitchell, p.219])!

Ref: Chapter 7 of Machine Learning by Tom Mitchell

Readings

- "Introduction to Machine Learning" by Ethem Alpaydin, Chapters 4.3-4.8
- (Optional)
  - Machine Learning by Tom Mitchell, Chapter 7