Foundations of Machine Learning

Classifier System Design

Oct 2021

Vineeth N Balasubramanian

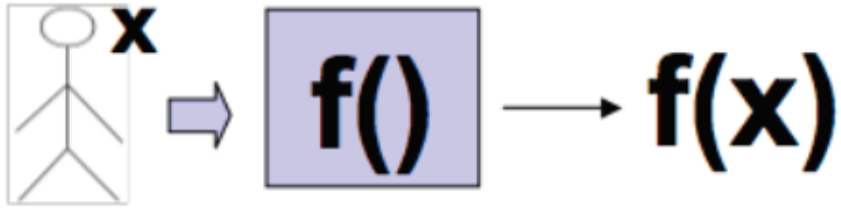
आई आई टी हैदराबाद
IIT Hyderabad

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization

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

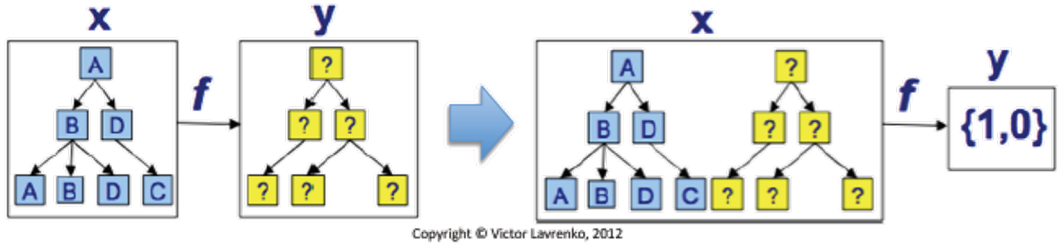
- Numeric
  - Normalization
    - Zero mean, unit variance: x'= (x - mean)/st.dev
    - In interval [0,1]: x'=(x - min)/(ma - min)
- Categorical
  - Usually encoded as numbers
    - Only equality testing is meaningful
- Ordinal
  - Encoded as numbers to preserve ordering
    - <=, >= operations meaningful

Feature Extraction from Data

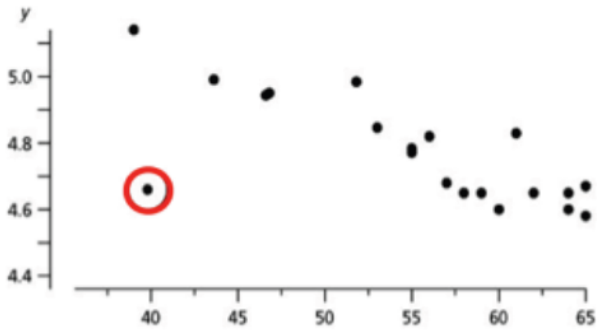
- Images
  - Pixel values, Segment and extract features, Handcrafted features: HOG, SIFT, etc
  - Deep learned features!
- Text
  - Bag of words, N-grams
  - Deep learned features!
- Speech
  - Mel Frequency Cepstral Co-efficients (MFCCs), Other frequency-based features
  - Deep learned features!
- Time-varying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

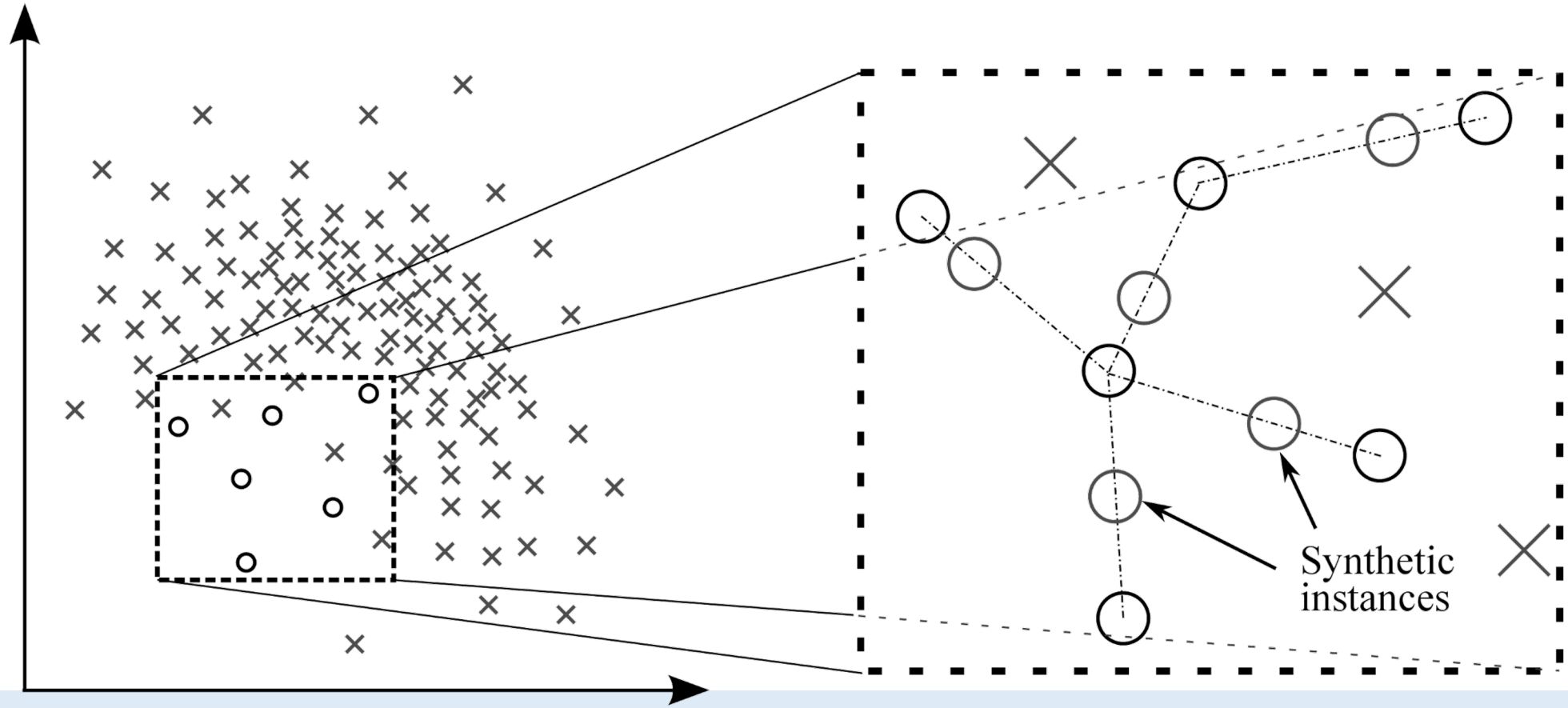
Handling Class Imbalance

- Undersampling (random and directed) is not effective and can even hurt performance.
- Random oversampling helps quite dramatically. 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.

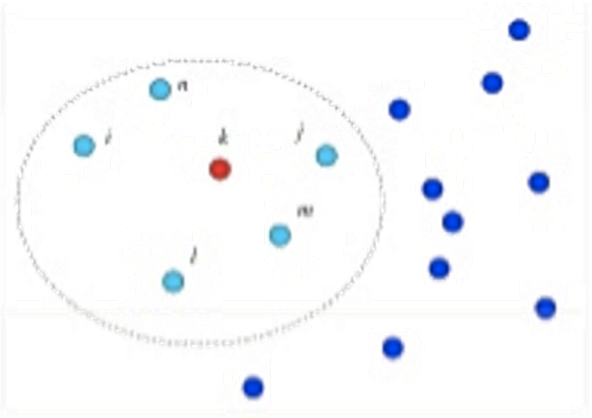
See Nathalie Japkowicz' work on "Inductive Learning from Imbalanced Datasets"

Example: SMOTE

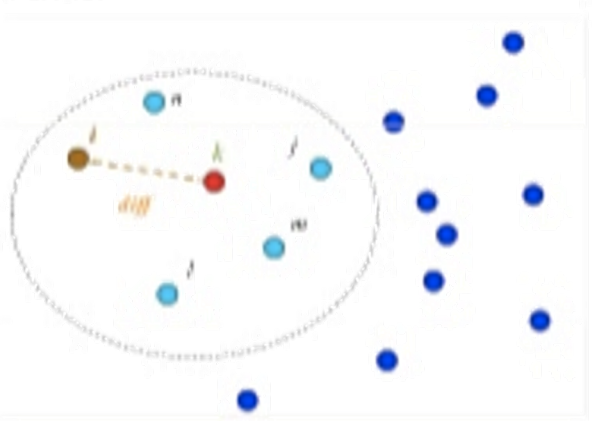
- SMOTE = Synthetic Minority Oversampling Technique



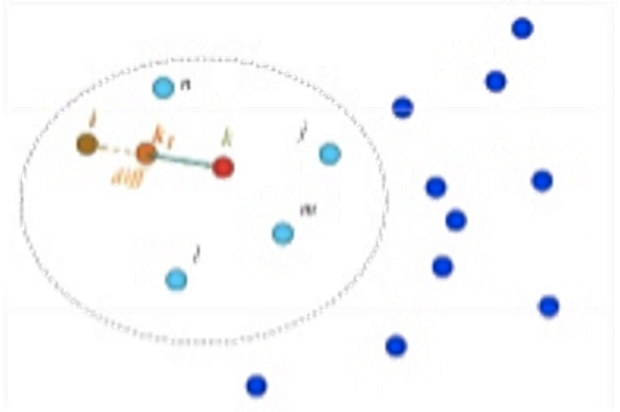
Example: SMOTE



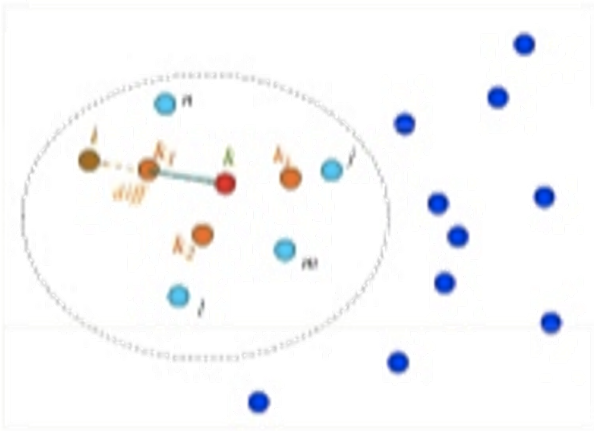
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)



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

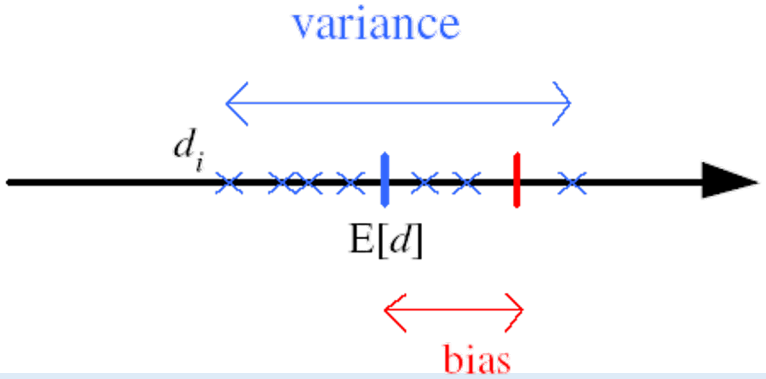
- CPUs vs GPUs
  - Deep learning has greatly benefited from GPUs
- Map-Reduce/Hadoop, Apache Spark, Vowpal Wabbit frameworks
  - Many learning algorithms amenable to partitioning of computations

Today

- Data Handling Issues
- Bias-Variance Tradeoff
- Regularization

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



True function 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

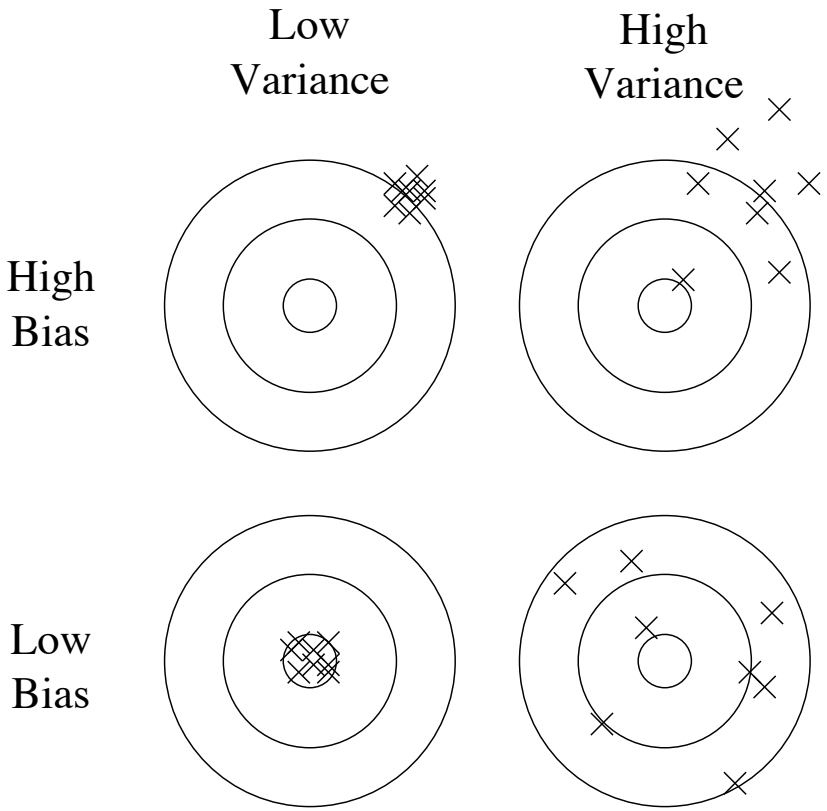
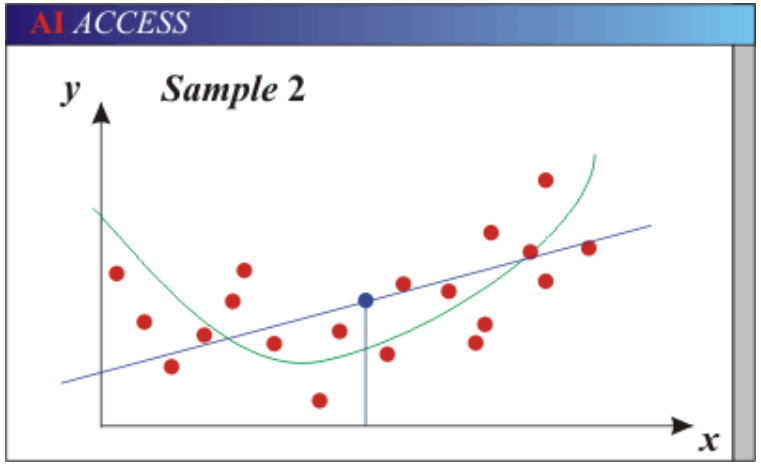
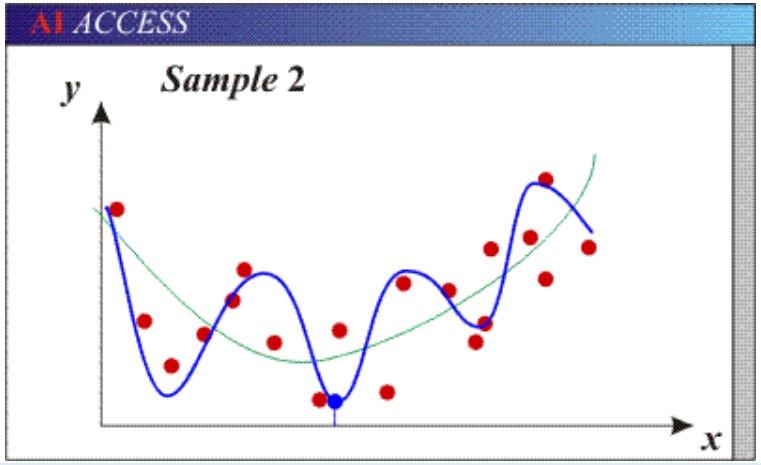


Figure 1: Bias and variance in dart-throwing.

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).

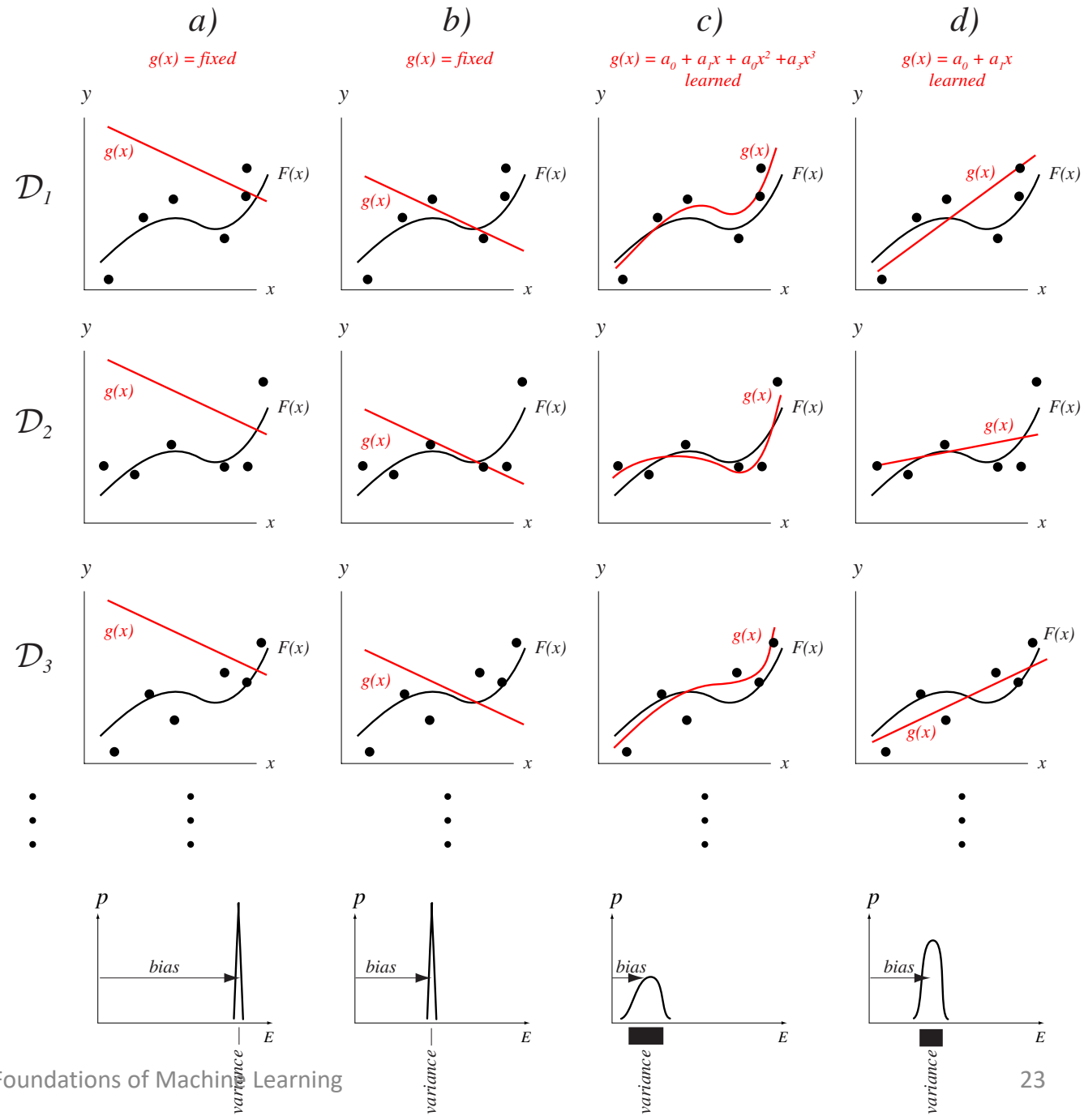
Slide Credit: D Hoiem

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

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

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) $r(w,b) = \sqrt{\sum_{w_j} |w_j|^2}$

---

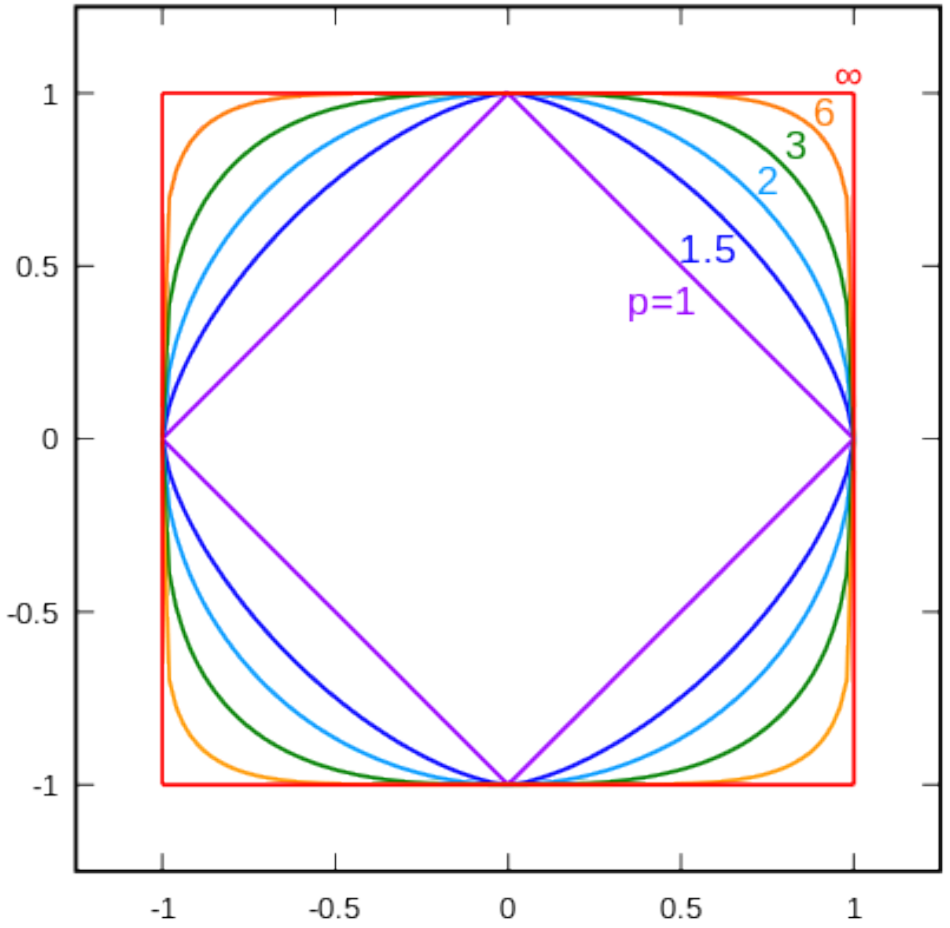
p-norm $r(w,b) = \sqrt[p]{\sum_{w_j} |w_j|^p} = \|w\|^p$

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

Readings

- "Introduction to Machine Learning" by Ethem Alpaydin, Chapters 4.3-4.8