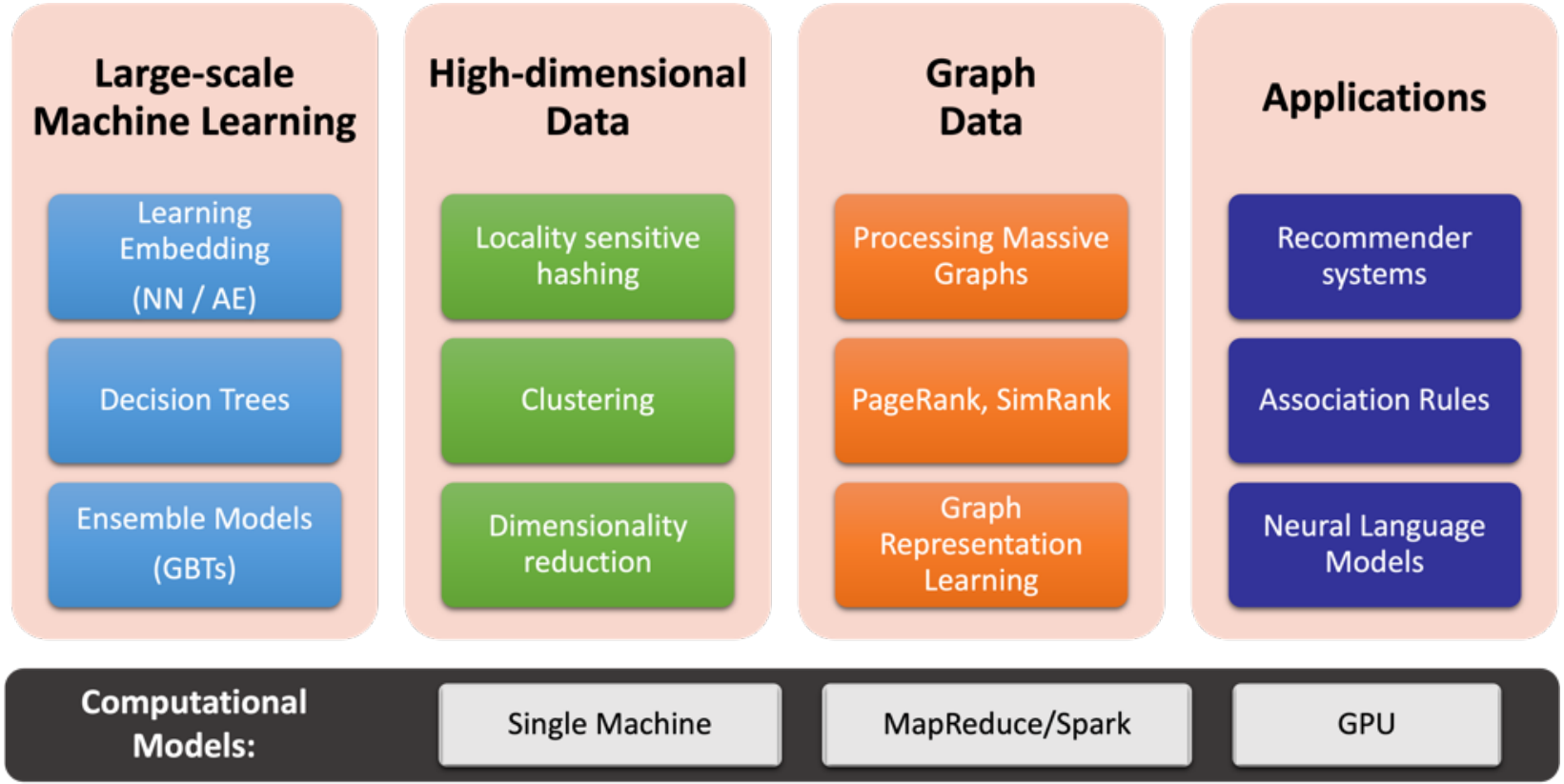
MIE524 Data Mining

Machine Learning: Decision Trees

Slides Credits:
Slides from Leskovec, Rajaraman, Ullman (http://www.mmds.org), Leskovec & Ghashami
Joseph Redmon, Alireza Ghane & Greg Mori, Dan Roth, David Sontag

MIE524: Course Topics (Tentative)



Major ML Paradigms

- **Supervised:**
  - Given "labeled data" $\{x, y\}$, learn $f(x) = y$
- **Unsupervised:**
  - Given only "unlabeled data" $\{x\}$, learn $f(x)$
- **Semi-supervised:**
  - Given some labeled $\{x, y\}$ and some unlabeled data $\{x\}$, learn $f(x) = y$
- **Active learning:**
  - When we predict $f(x) = y$, we then receive true y^*
- **Transfer learning:**
  - Learn $f(x)$ so that it works well on new domain $f(z)$

Supervised Learning

Given some data:

- "Learn" a function to map from the **input** to the **output**

- **Given:**

  Training examples $\left(\boldsymbol{x_i}, \boldsymbol{y_i} = \boldsymbol{f}(\boldsymbol{x_i})\right)$ for some unknown function $\boldsymbol{f}$
- **Find:**

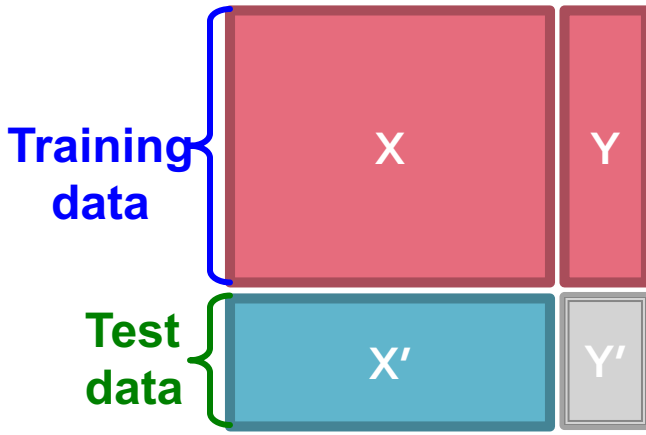
  A good approximation to $\boldsymbol{f}$

Supervised Learning

- **Would like to do prediction:**
  estimate a function $f(x)$ so that $y = f(x)$
- **Where *y* can be:**
  - **Real number:** Regression
  - **Categorical: Classification**
  - **Complex object:**
    - Ranking of items, Parse tree, etc.
- **Data is labeled:**
  - Have many pairs $\{(x, y)\}$
    - **x** … vector of binary, categorical, real valued features
    - **y** … class, or a real number

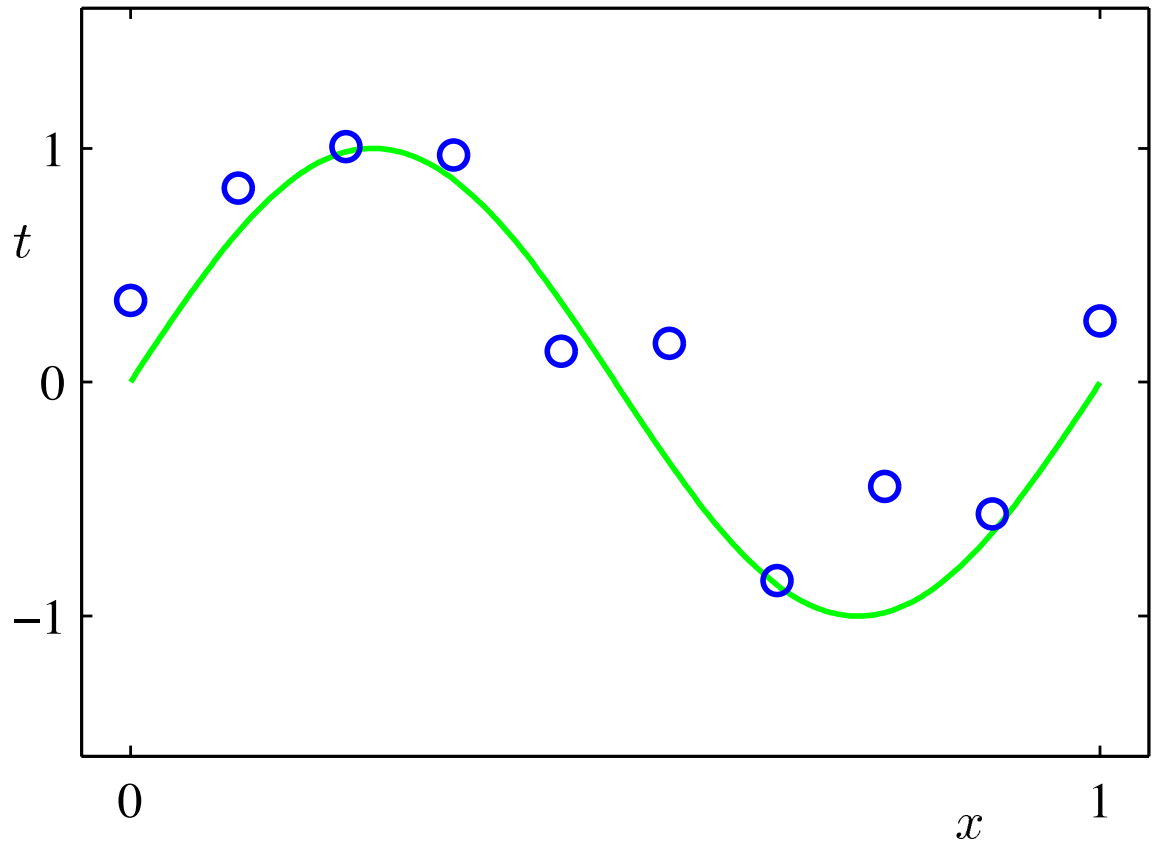
Supervised Learning

- **Task:** Given data (X, Y) build a model $f()$ to predict Y' based on X'
- **Strategy:** **Estimate $y = f(x)$ on (X, Y)**
  Hope that the same $f(x)$ also works to predict unknown Y'



  - The **"hope"** is called **generalization**
    - **Overfitting:** **If $f(x)$ predicts well Y but is unable to predict Y'**
  - **We want to build a model that generalizes well to unseen data**

An Example - Polynomial Curve Fitting



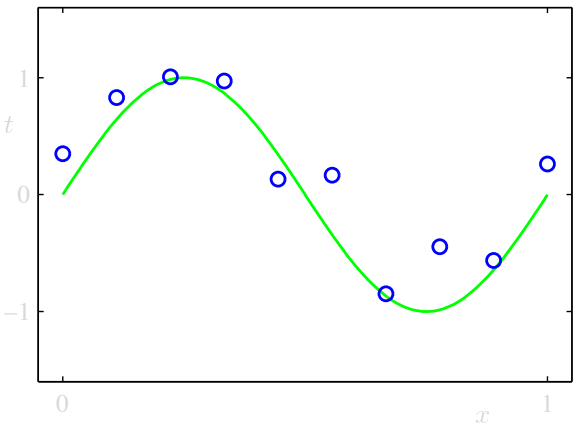
- Suppose we are given training set of N observations $(x_1, \ldots, x_N)$ and $(t_1, \ldots, t_N)$, $x_i, t_i \in \mathbb{R}$
- Regression problem, estimate $y(x)$ from these data

Polynomial Curve Fitting

- What form is $y(x)$?
  - Let's try polynomials of degree M:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.



- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$

- Among functions in the class, choose that which minimizes this error

Polynomial Curve Fitting

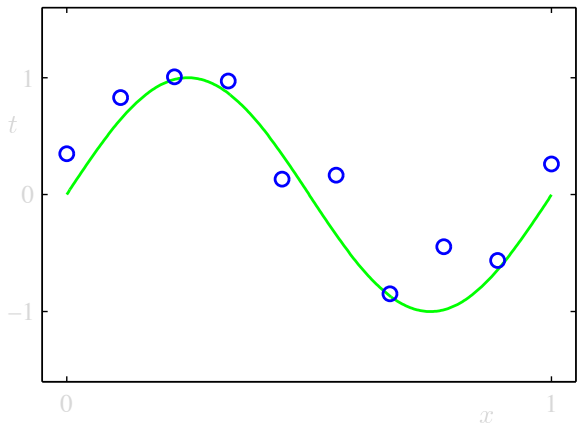
- What form is $y(x)$?
  - Let's try polynomials of degree M:

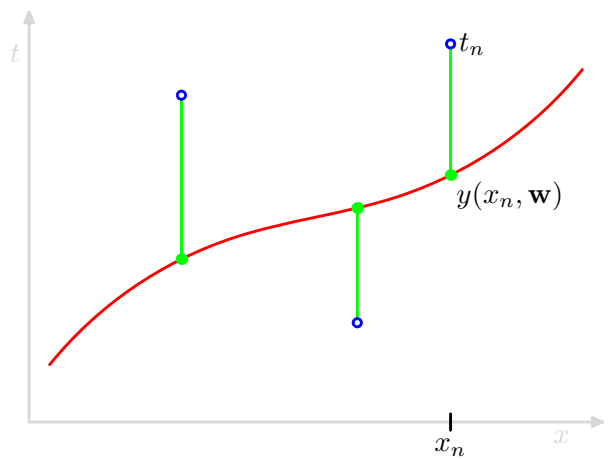
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.
- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$

- Among functions in the class, choose that which minimizes this error





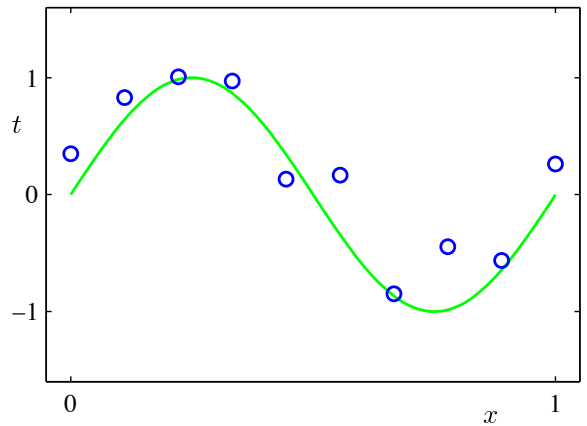
Polynomial Curve Fitting

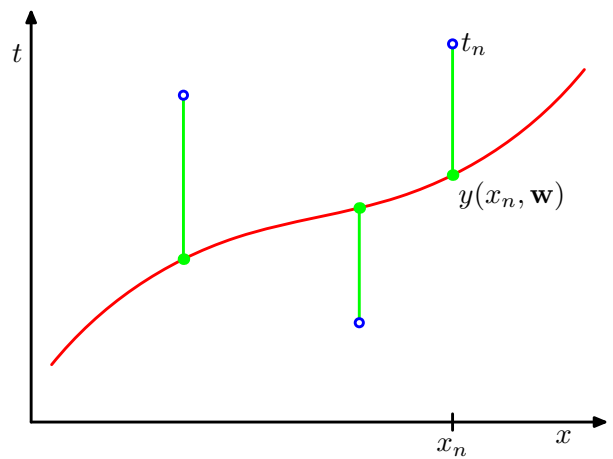
- What form is $y(x)$?
  - Let's try polynomials of degree M:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.
- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$

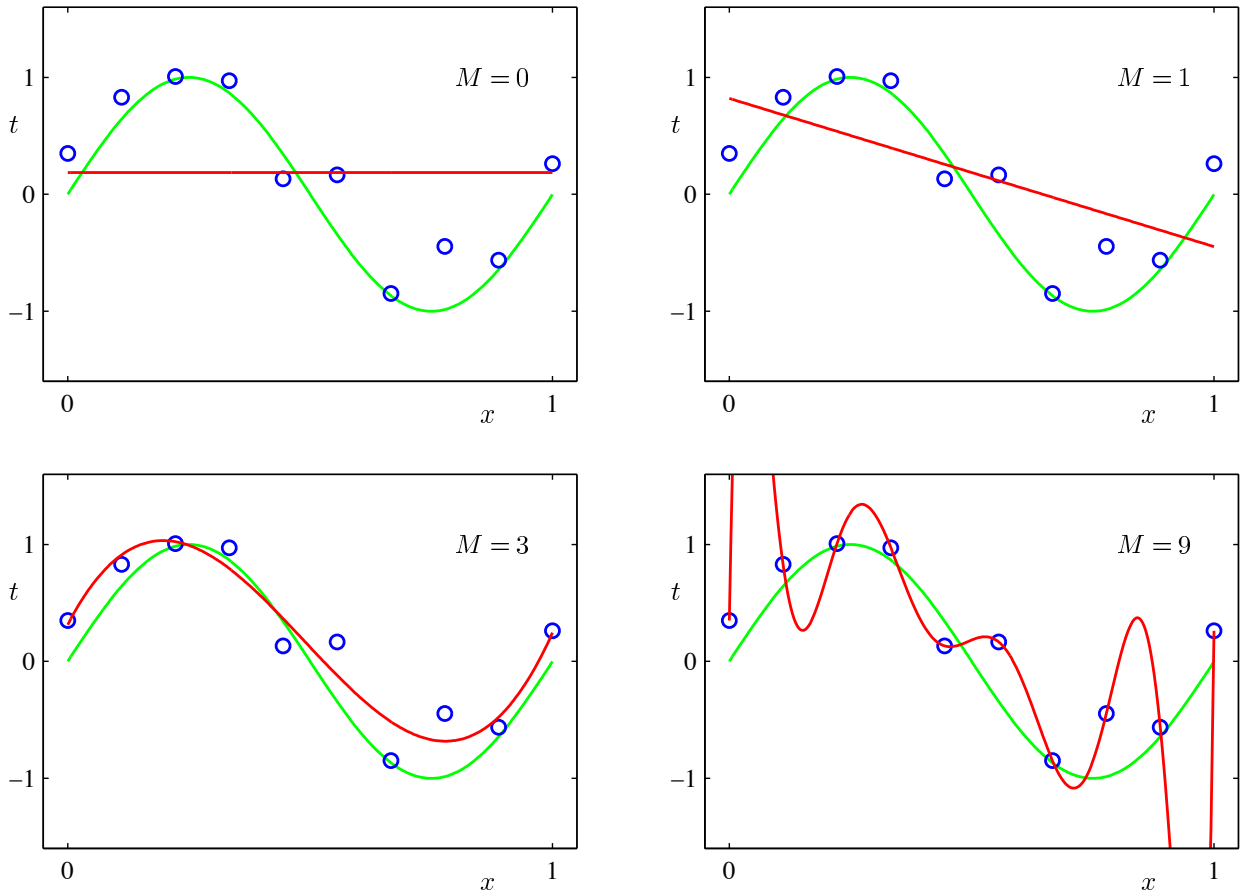




- Among functions in the class, choose that which minimizes this error

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$$

Which Degree of Polynomial?

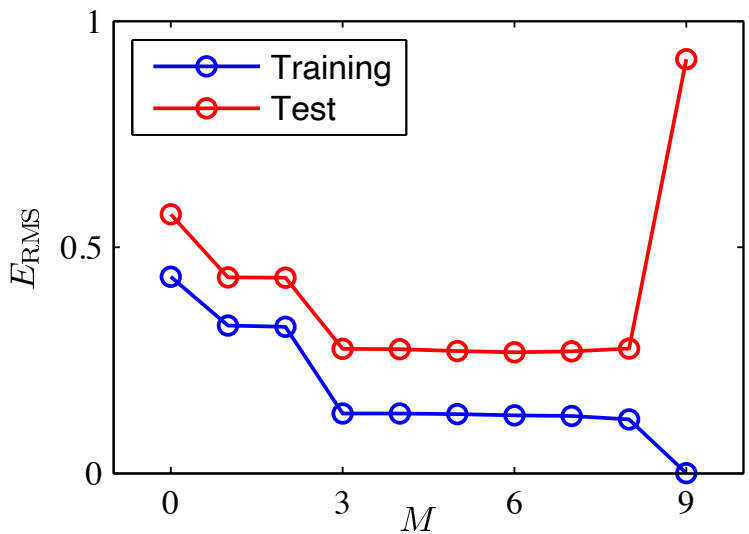


- A model selection problem
- $M = 9 \rightarrow E(\boldsymbol{w}^*) = 0$: This is over-fitting

Generalization

- Want models that **generalize** to new data
  - Train model on training set
  - Measure performance on held-out test set
  - Performance on test set is good estimate of performance on new data

- Measure generalization using a separate, held-out test set
  - Use root-mean-squared (RMS) error: ERMS = $\sqrt{2E(\boldsymbol{w}^*)/N}$



Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

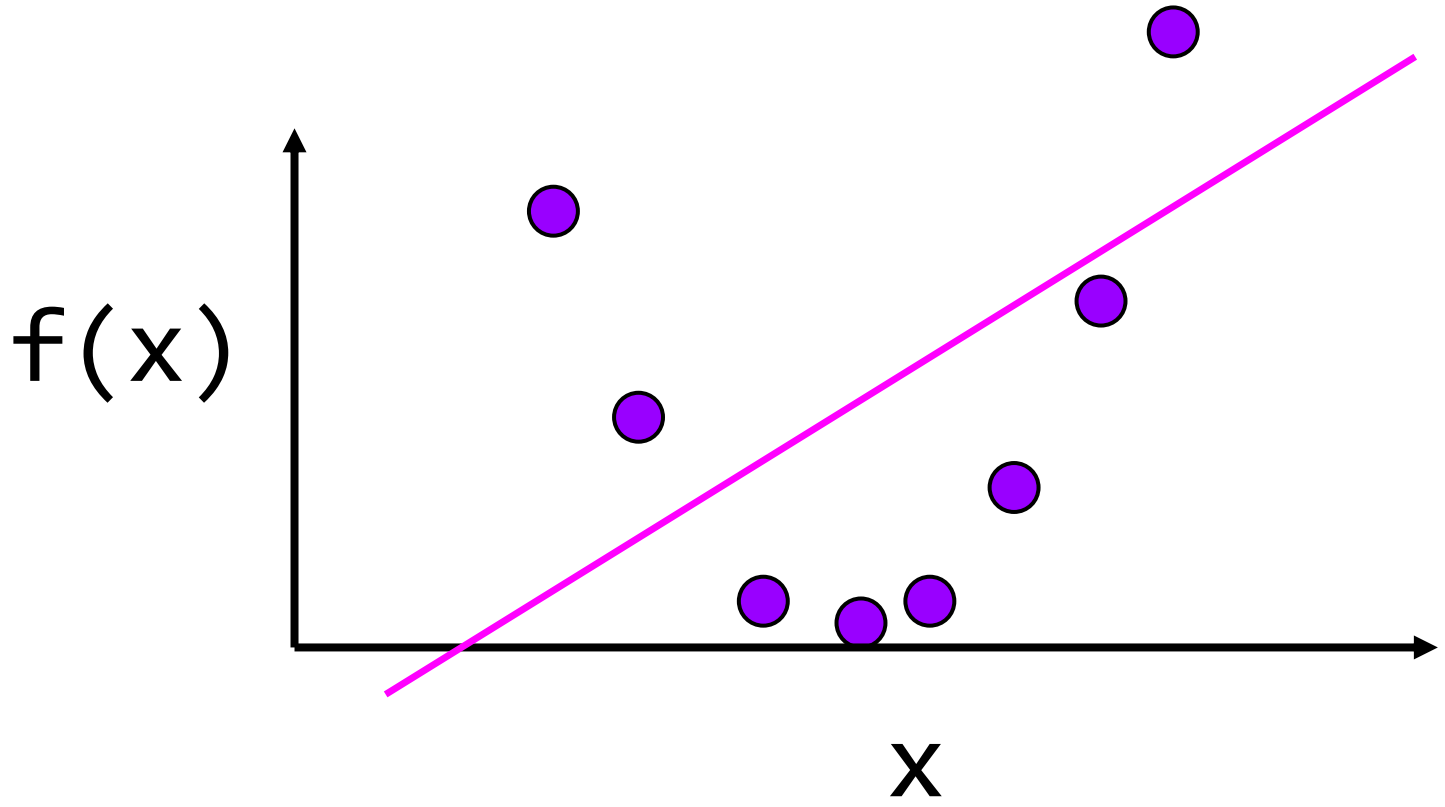
Error = Noise + Bias + Variance

- Noise
  - Random variations in data
- Bias
  - **Error from assumptions** model makes about data
  - Less complex algorithms -> more assumptions about data
- Variance
  - Algorithm's **sensitivity to noise**
  - More complex algorithms are more sensitive!
  - High variance hurts generalization

Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

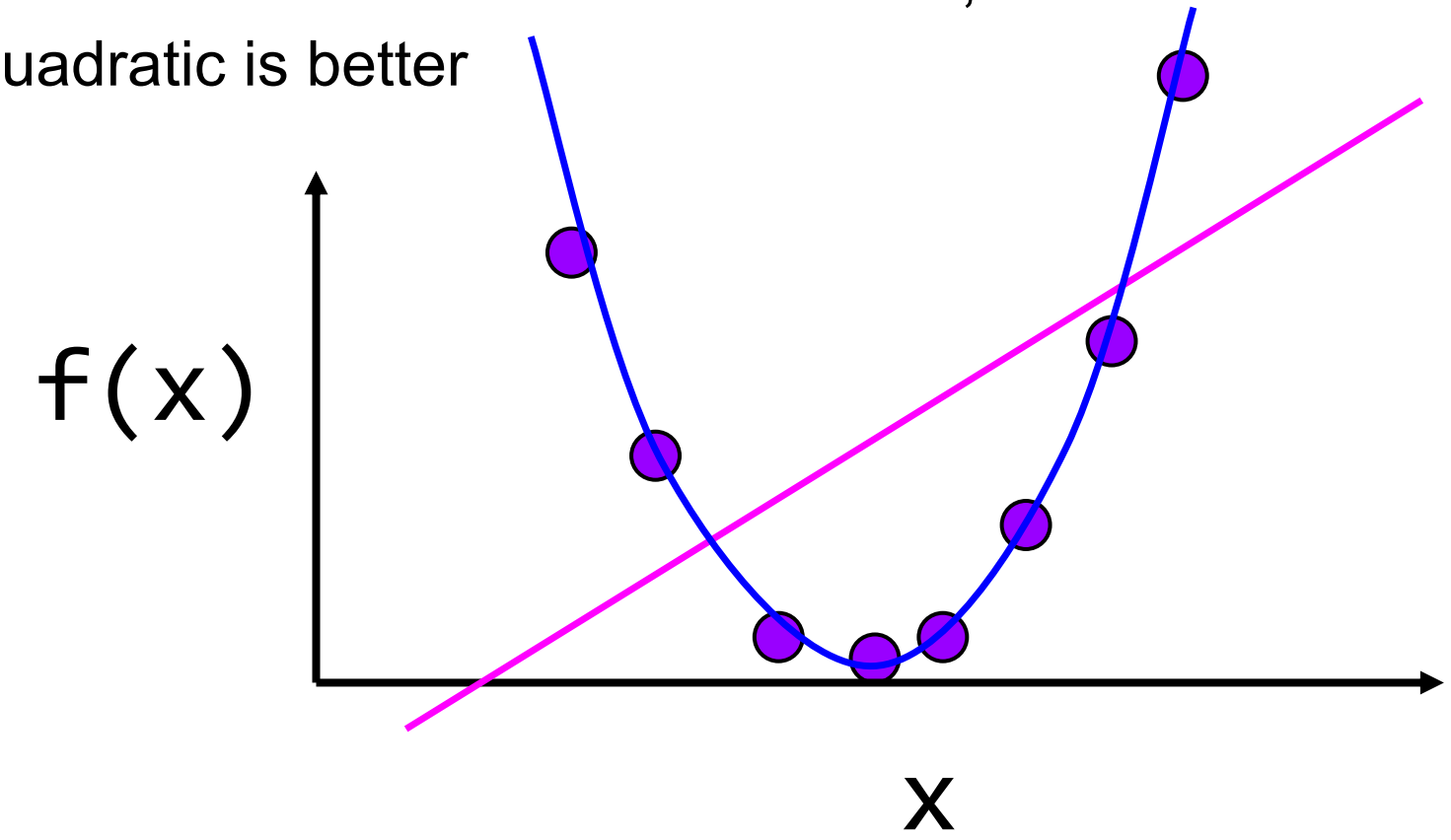
- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn't



Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

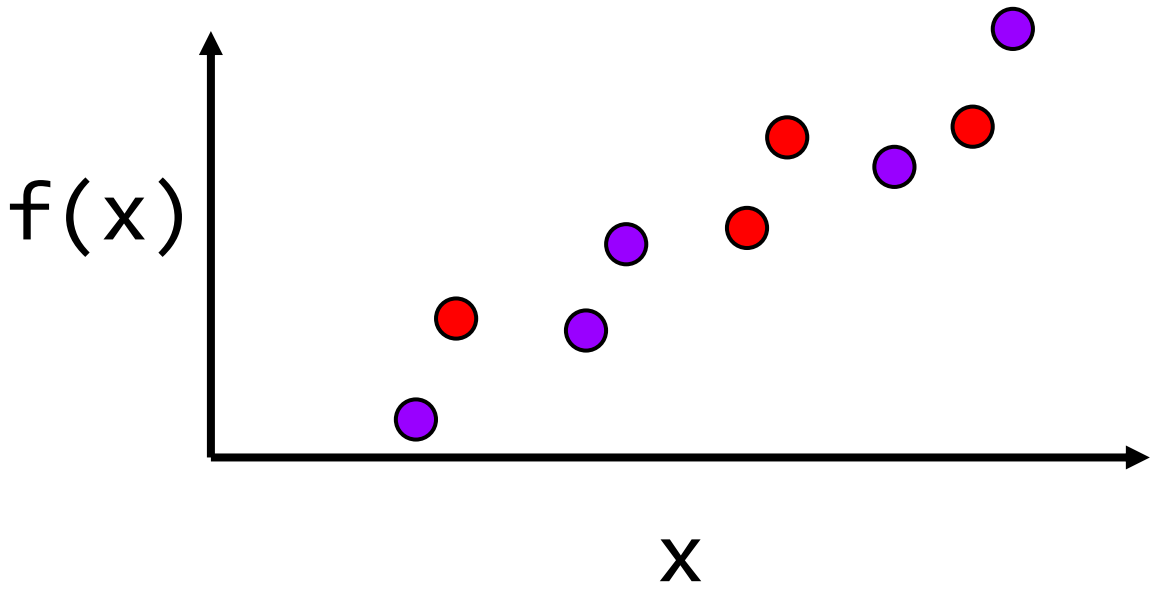
- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn't
  - Quadratic is better



Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

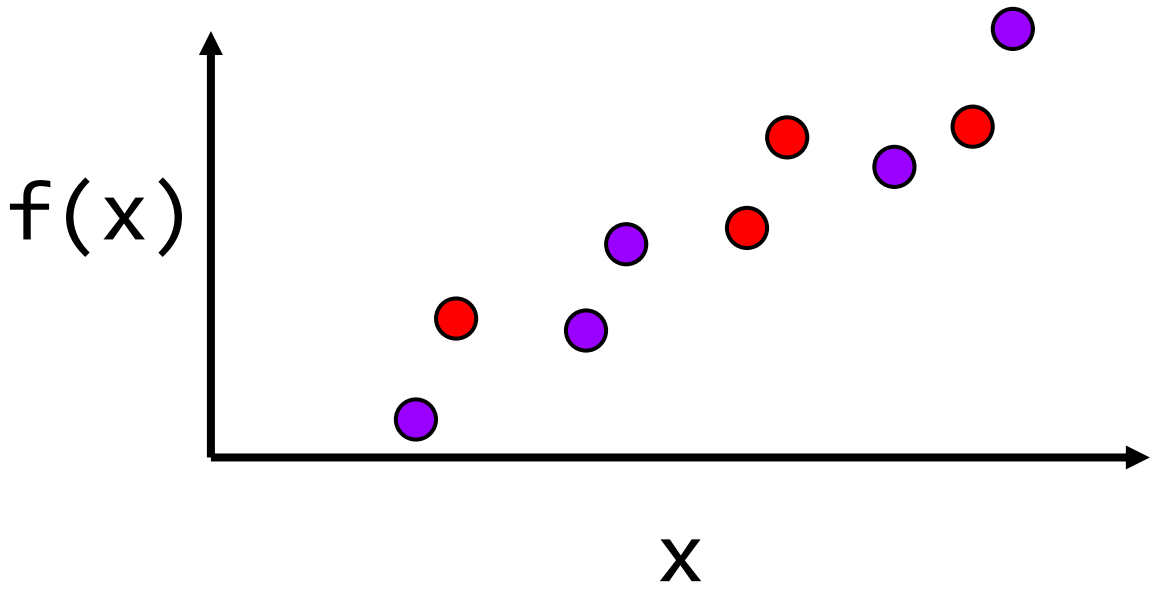
- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!

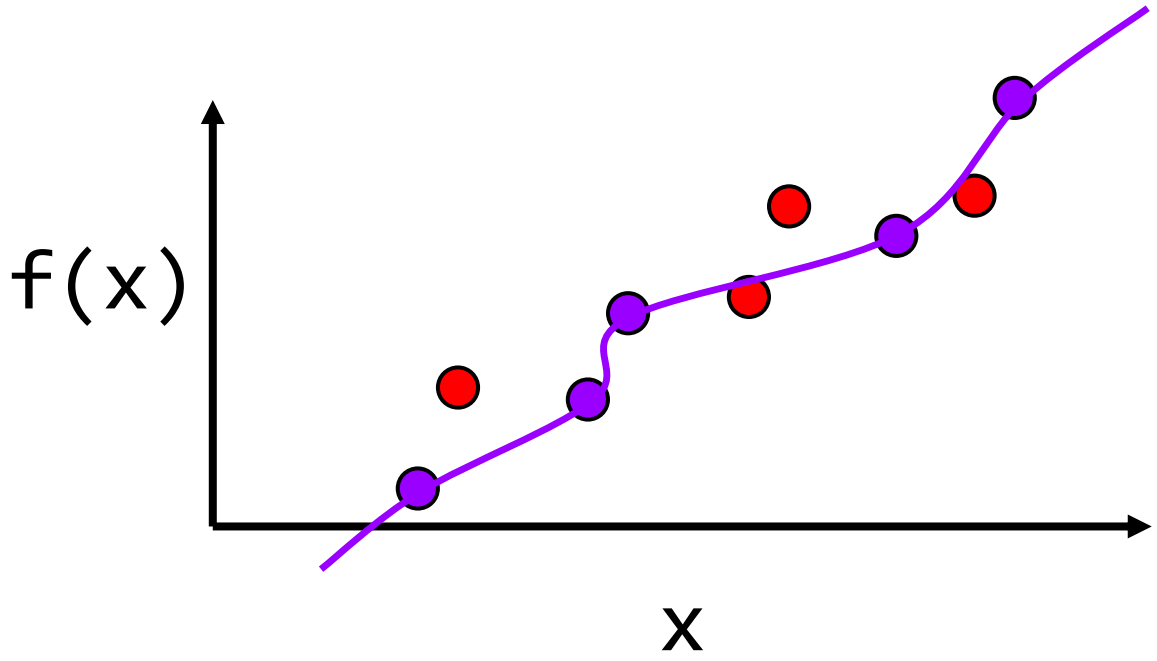


Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!

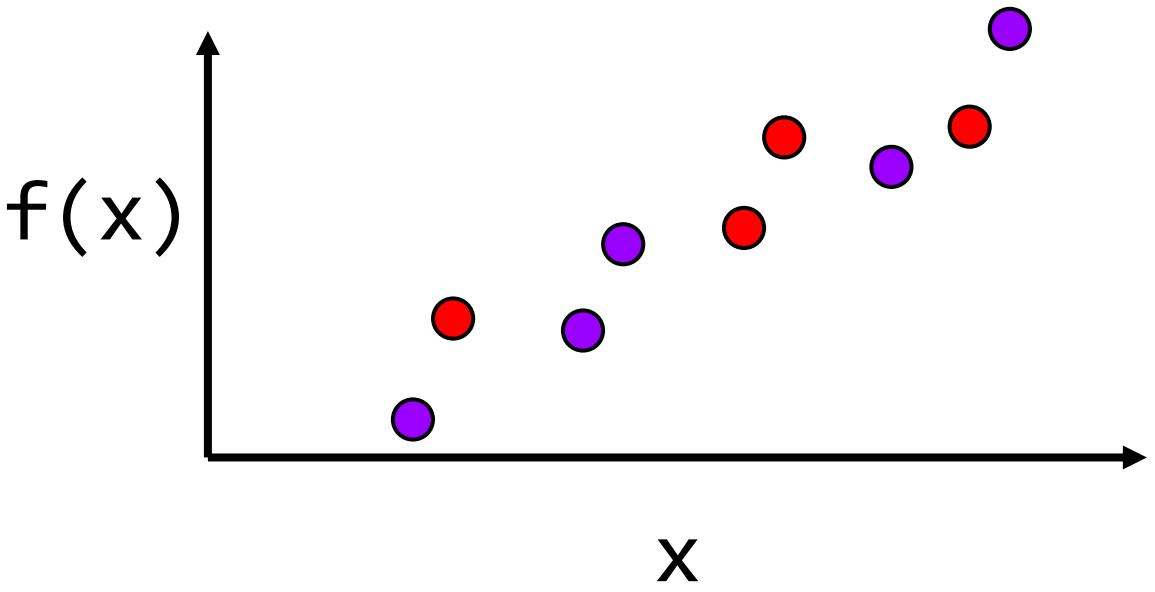


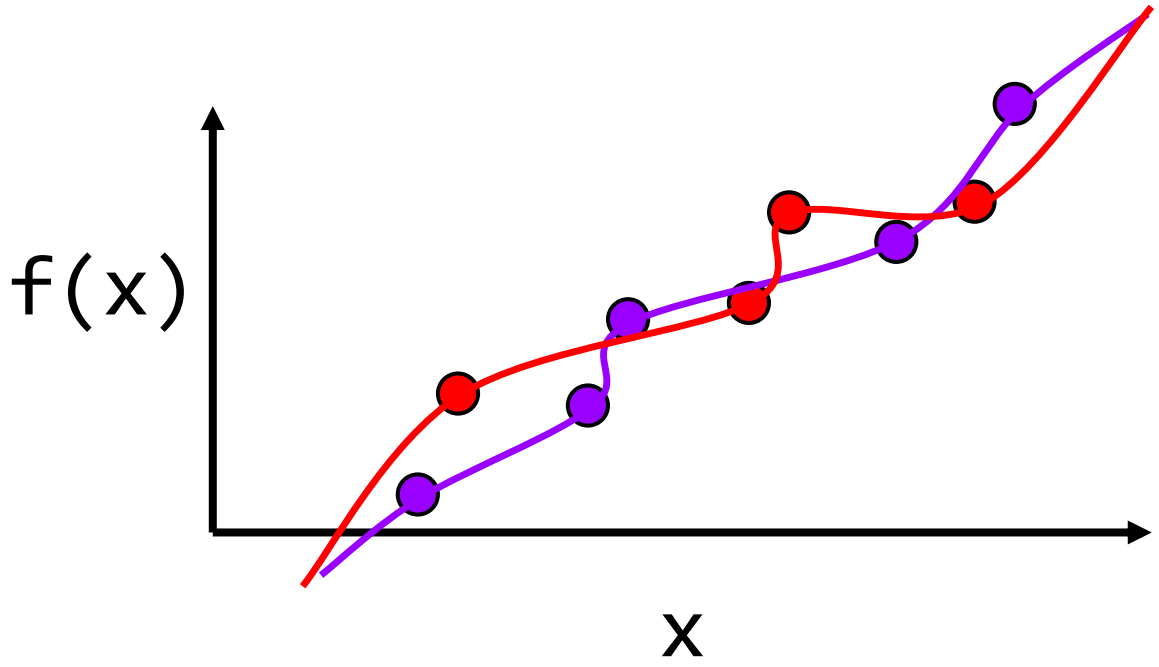


Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

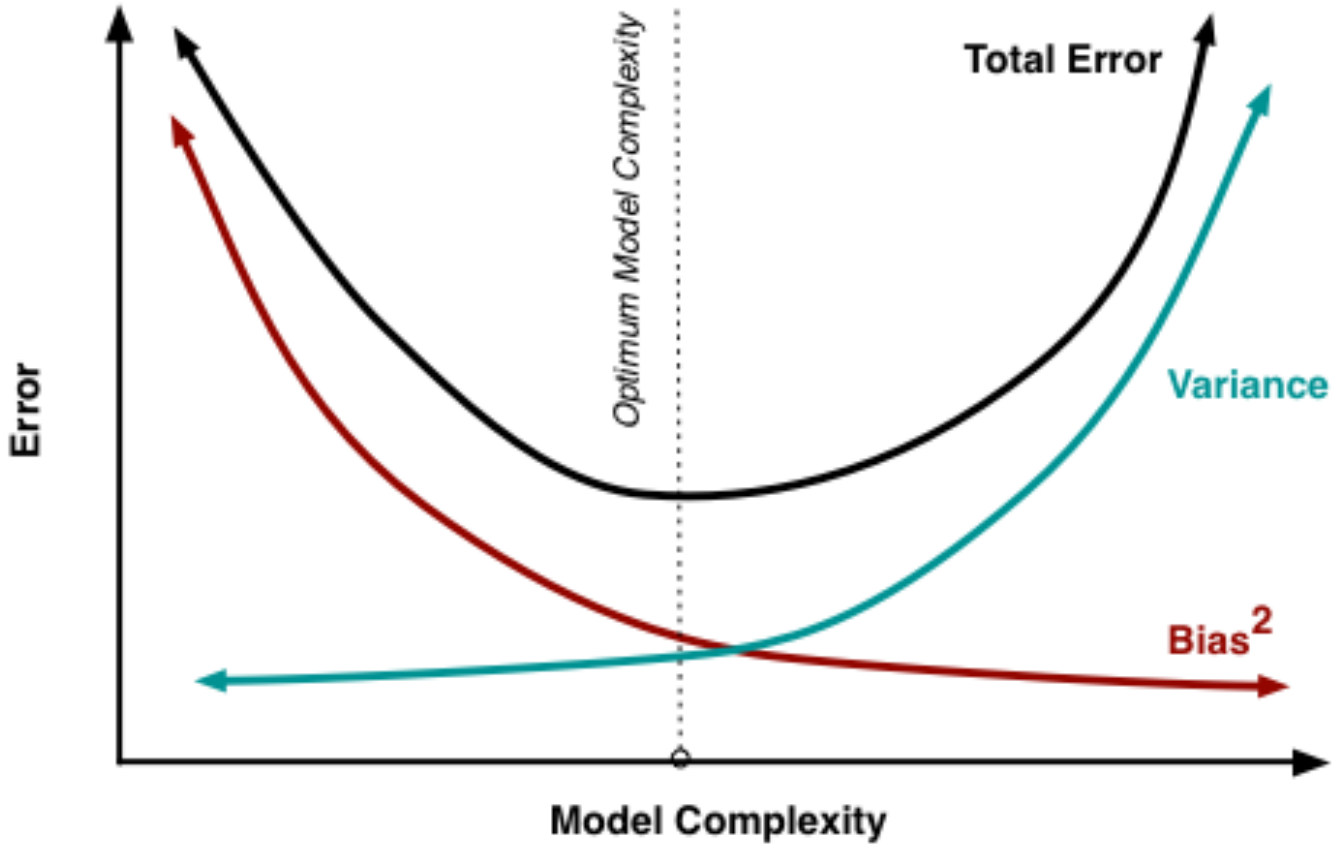
- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!
  - High variance hurts generalization, *overfitting*





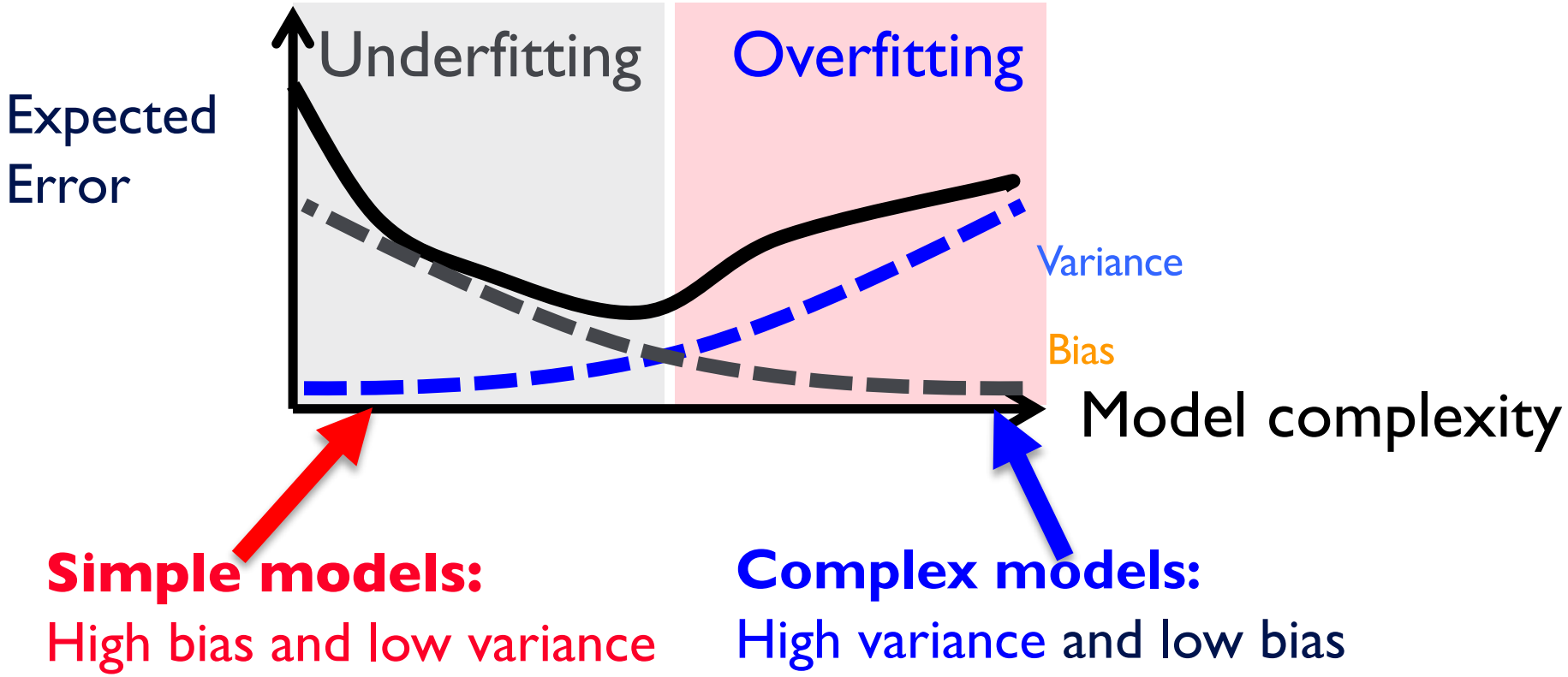
Bias / Variance tradeoff

Based on Joseph Redmon's slides,
UW Computer Vision class



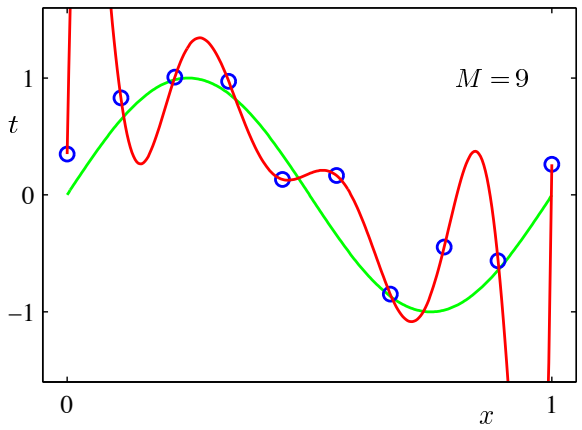
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

Underfitting and Overfitting



Adapted from slides by Dan Roth

Controlling Over-fitting: Regularization

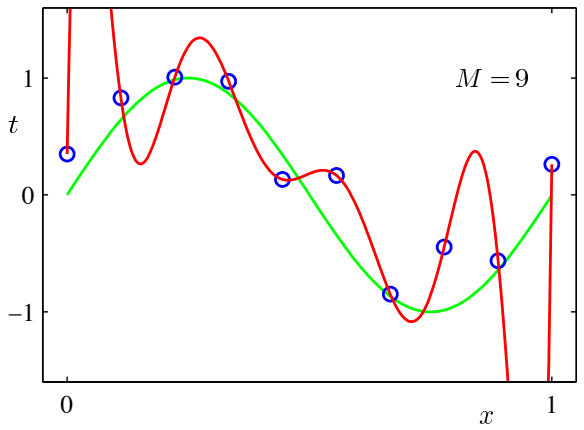


	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- As order of polynomial M increases, so do coefficient magnitudes
- Penalize large coefficients in error function:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

Controlling Over-fitting: Regularization

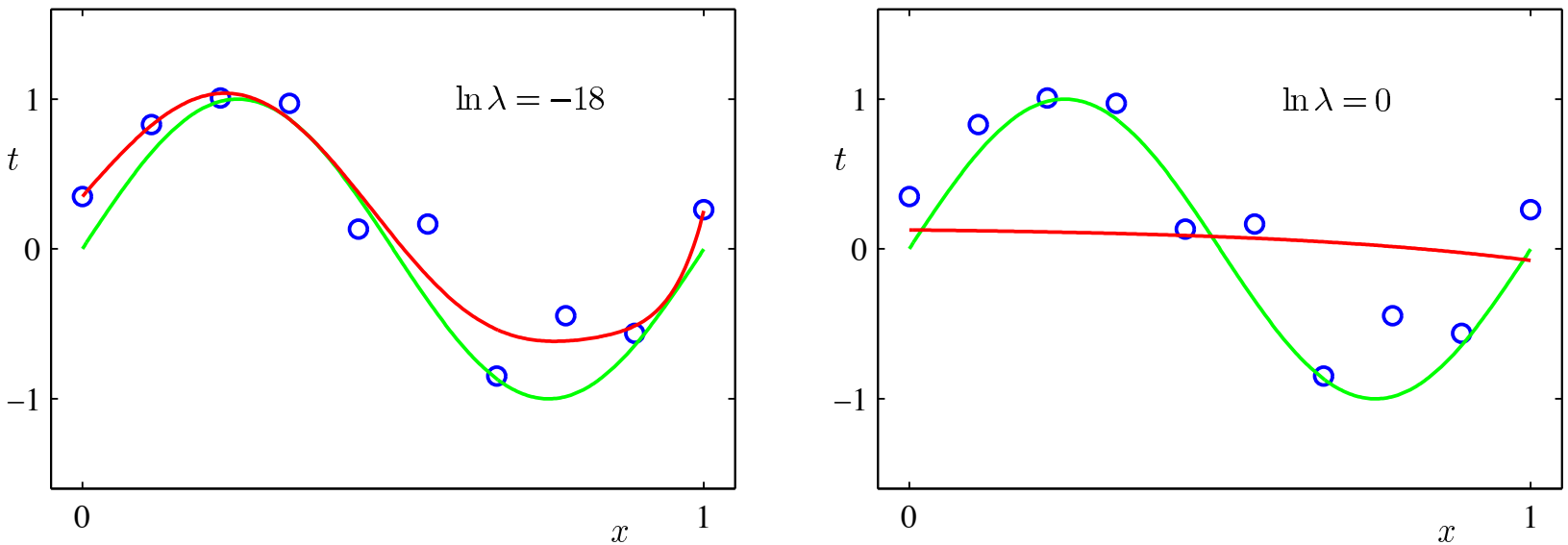


	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

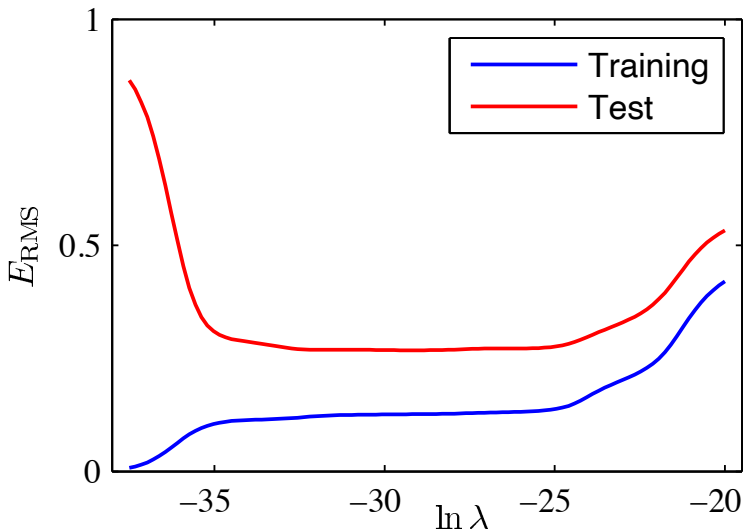
- As order of polynomial M increases, so do coefficient magnitudes
- Penalize large coefficients in error function:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

Controlling Over-fitting: Regularization



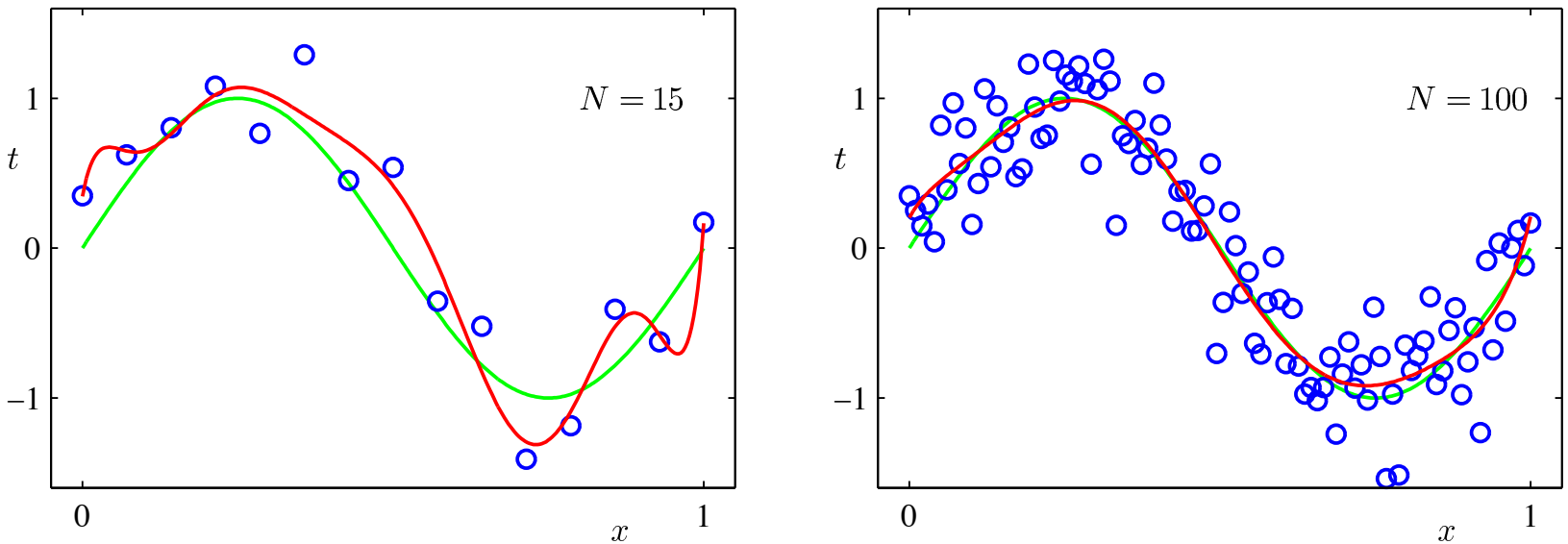
Controlling Over-fitting: Regularization



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

- Note the E_{RMS} for the training set. Perfect match of training set with the model is a result of over-fitting
- Training and test error show similar trend

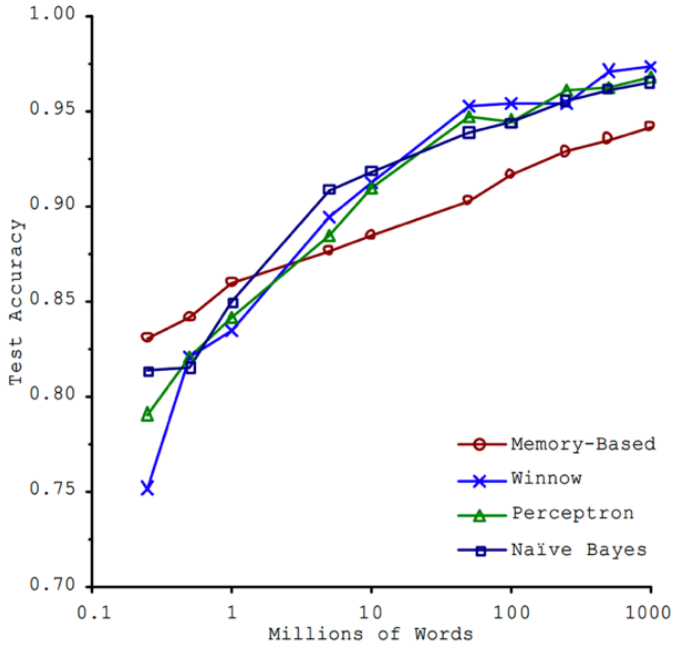
Over-fitting: Dataset size



- With more data, more complex model ($M = 9$) can be fit
- Rule of thumb: 10 datapoints for each parameter

Why Large-Scale ML?

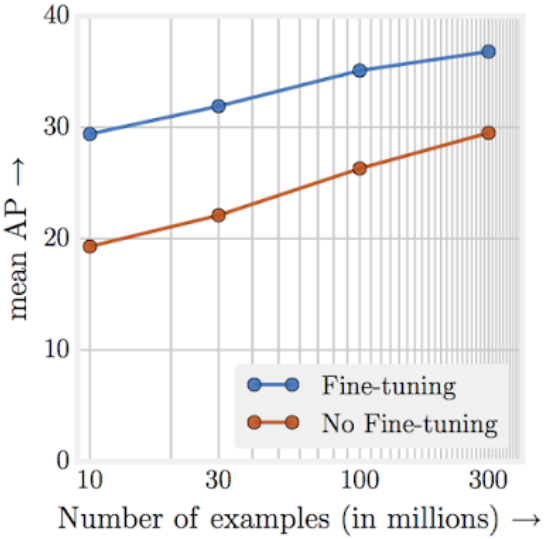
- **Brawn or Brains?**
  - In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML-based language translation
- **Findings:**
  - **Size of the dataset** used to train the model **mattered more** than the model itself



Banko, M. and Brill, E. (2001) , "Scaling to Very Very Large Corpora for Natural Language Disambiguation"

Why Large-Scale ML?

- **The Unreasonable Effectiveness of Data**
  - In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision
- **Findings:**
  - Performance increases logarithmically **based on volume of training data**
  - Complexity of modern ML models (i.e., deep neural nets) allows for even **further performance gains**



- **Large datasets + large ML models => amazing results!!**

"Revisiting Unreasonable Effectiveness of Data in Deep Learning Era": https://arxiv.org/abs/1707.02968

Why Worry About Non-Deep Models?

A few reasons why this is important:

- Classical tasks in NLP and Vision are getting commoditized (you take pretrained model and fine tune it), but there are many other unique ML tasks.
- Deep models are often too risky/finicky. Traditional models allow you to encode prior knowledge better and give you more control
- Personally, if I am working on a well understood problem I'd use deep learning, but if I am the first engineer to work on a problem/classifier I'd use techniques we'll discuss here.

Decision Trees

Preface: Decision Trees

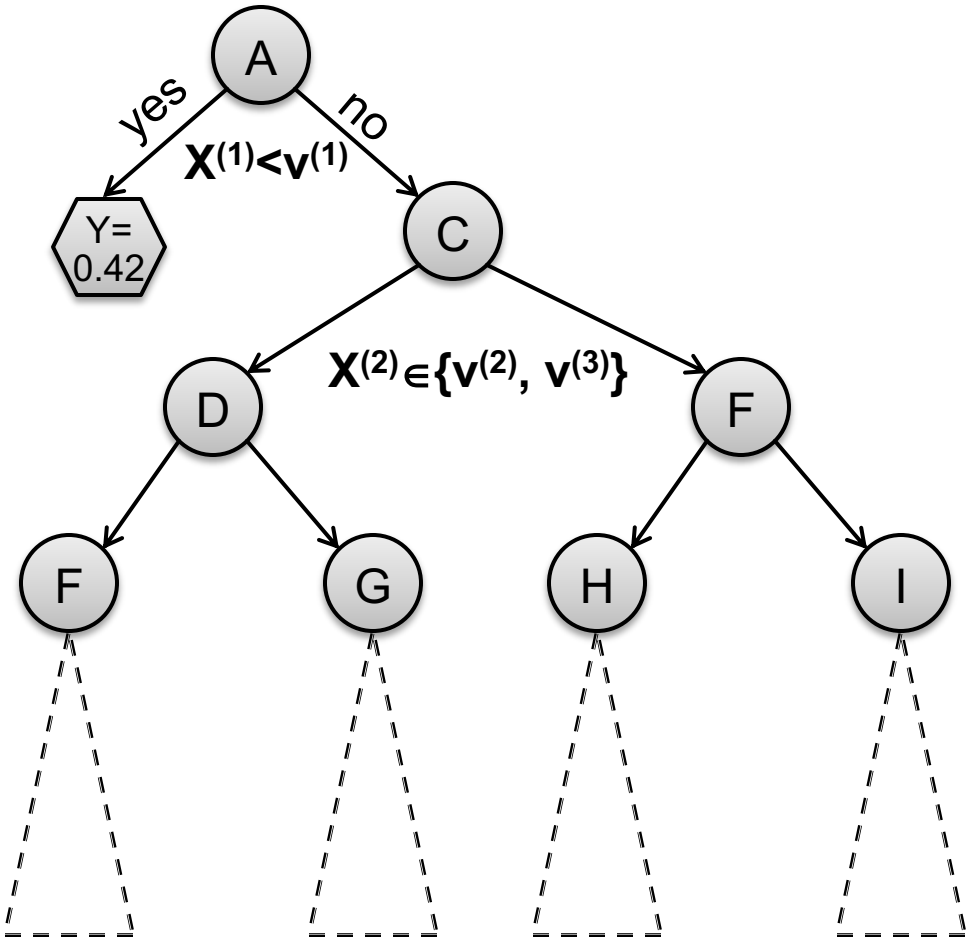
- **Decision trees are part of ML since 1980s**
  - Introduced by Leo Breiman in 1984
  - Notable algorithms: ID3, C4.5
- **More recent innovations include:**
  - Boosted decision trees (gradient boosted DT)
  - Random forest
- Even though DTs are old, hand-engineered and heuristic, they are a method of choice for tabular data and for Kaggle competitions. ☺

Decision Tree Learning

- **Given one attribute (e.g., lifespan), try to predict the value of new people's lifespans by means of some of the other available attribute**
- **Input attributes:**
  - **d** features/attributes: $\boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \ldots\ \boldsymbol{x^{(d)}}$
  - Each $\boldsymbol{x^{(j)}}$ has **domain** $\boldsymbol{O_j}$
    - **Categorical:** $\boldsymbol{O_j} = \{red, blue\}$
    - **Numerical:** $\boldsymbol{H_j} = (0, 10)$
  - $\boldsymbol{Y}$ is output variable with domain $\boldsymbol{O_Y}$:
    - **Categorical:** Classification, **Numerical:** Regression
- **Data D:**
  - $\boldsymbol{n}$ examples $(\boldsymbol{x_i}, \boldsymbol{y_i})$ where $\boldsymbol{x_i}$ is a $\boldsymbol{d}$-dim feature vector, $\boldsymbol{y_i} \in \boldsymbol{O_Y}$ is output variable
- **Task:**
  - Given an input data vector $\boldsymbol{x}$ predict output label $\boldsymbol{y}$

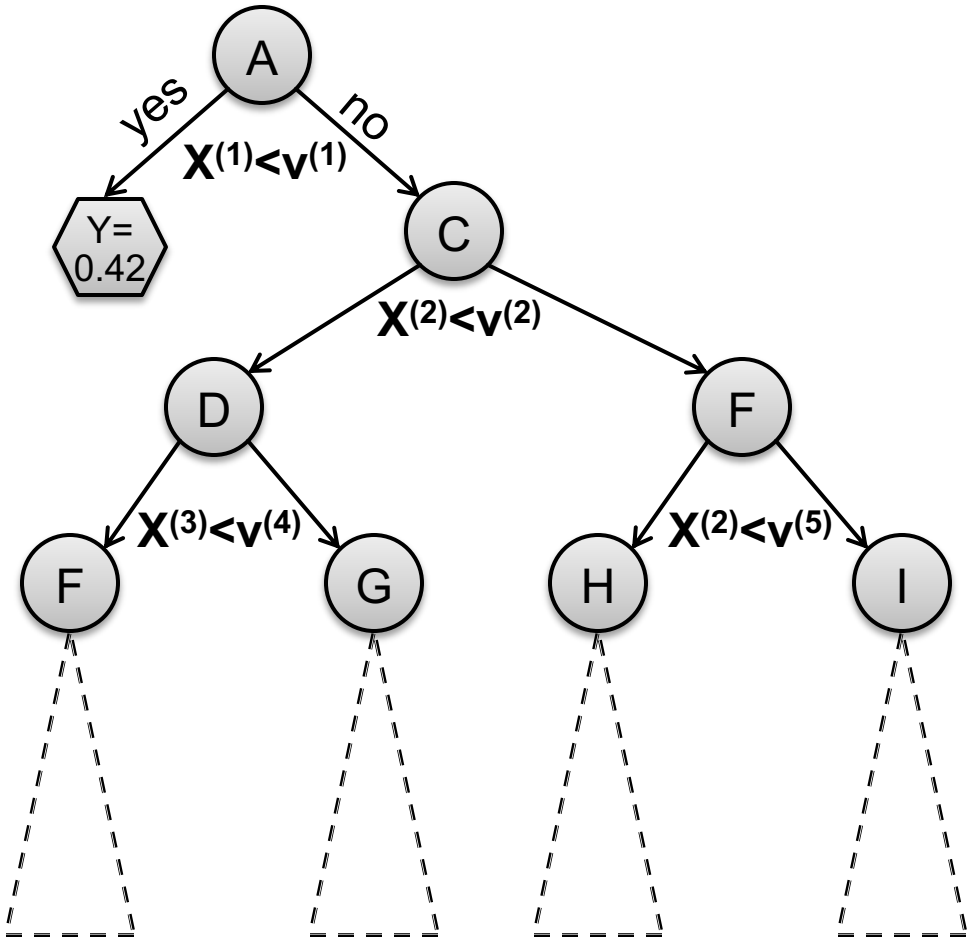
Decision Trees

- A **Decision Tree** is a tree-structured plan of a set of attributes to test in order to predict the output



Decision Trees

- **Decision trees:**
  - Split the data at each internal node
  - Each leaf node makes a prediction
- **Lecture today:**
  - Binary splits: $X^{(j)} < v$
  - Numerical attributes
  - Regression

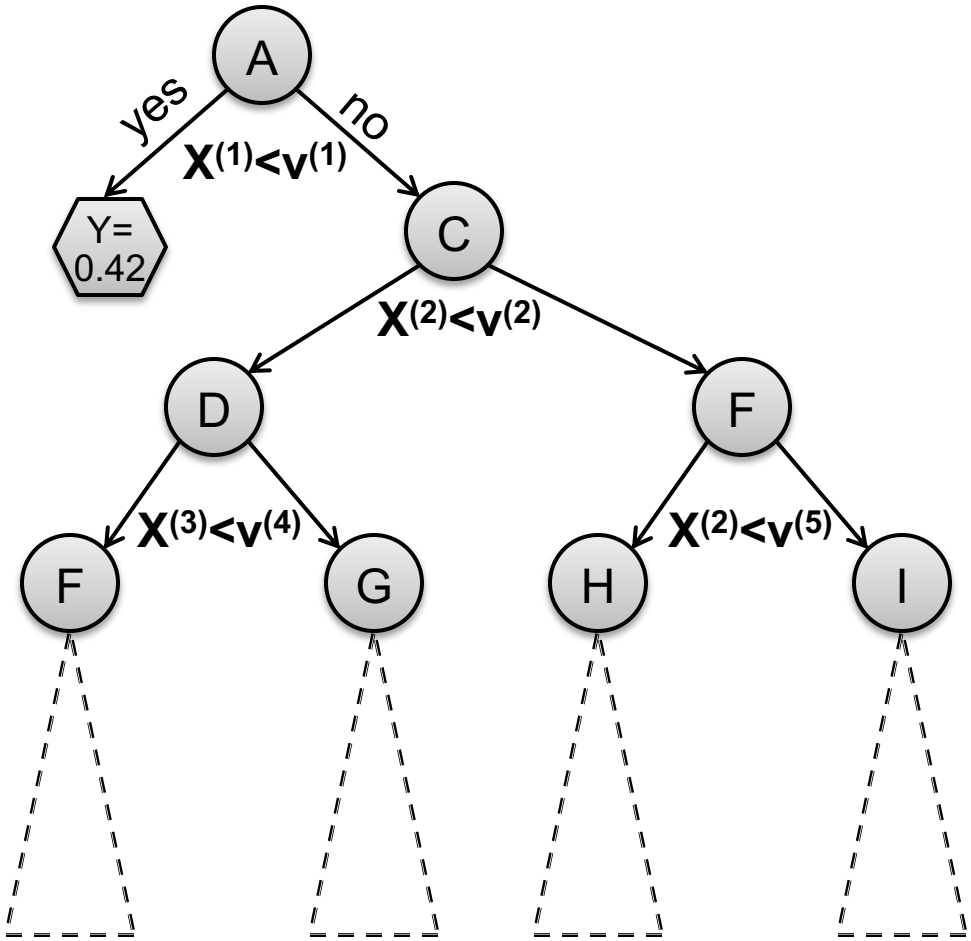


How to make predictions?

- **Input:** Example x_i
- **Output:** Predicted $\hat{y}_i$

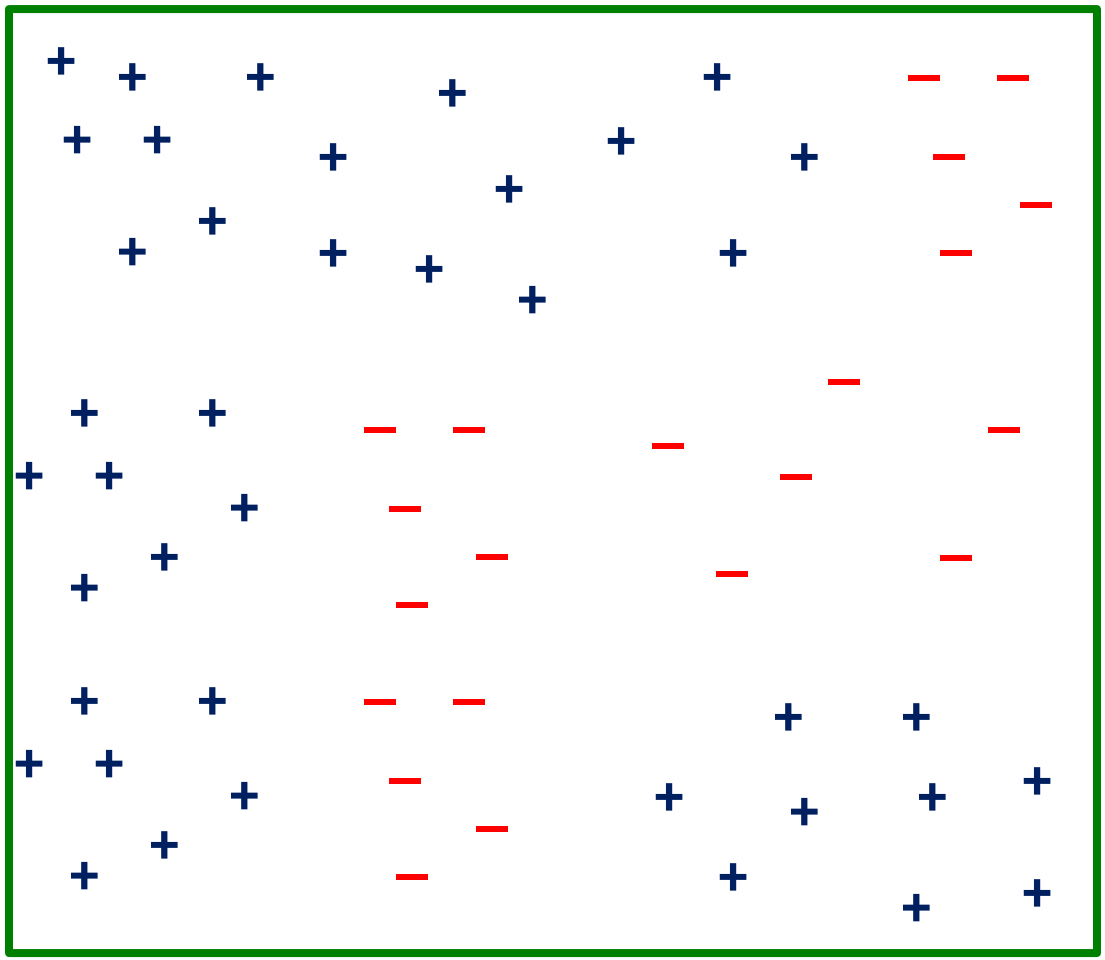
- "Drop" x_i down the tree until it hits a leaf node

- Predict the value stored in the leaf that x_i hits



Decision Trees: feature space

- **Alternative view:**

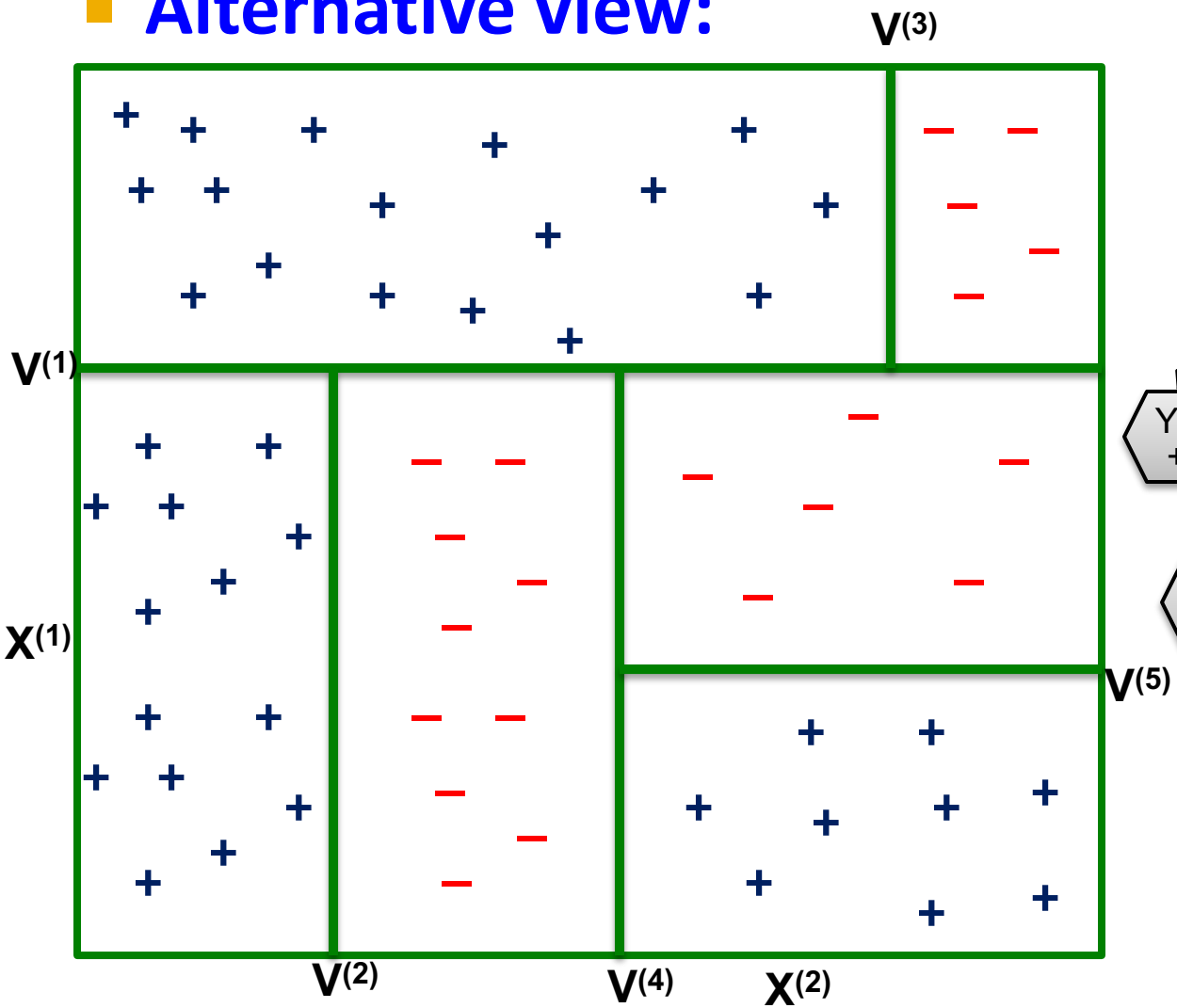


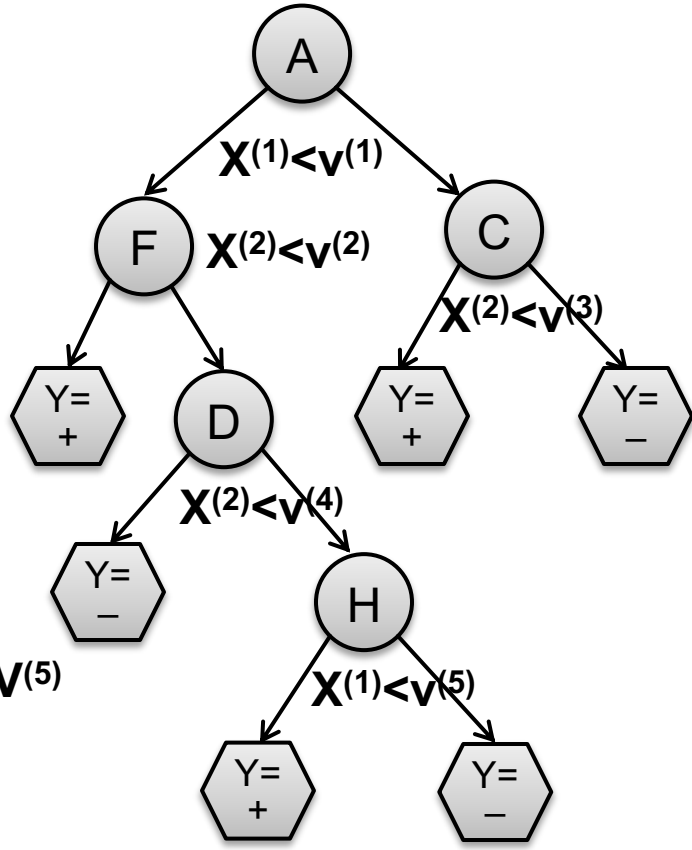
Data is 2-dim: $x = x^{(1)}, x^{(2)}$

Class label: $y = \{+, -\}$

Decision Trees: feature space

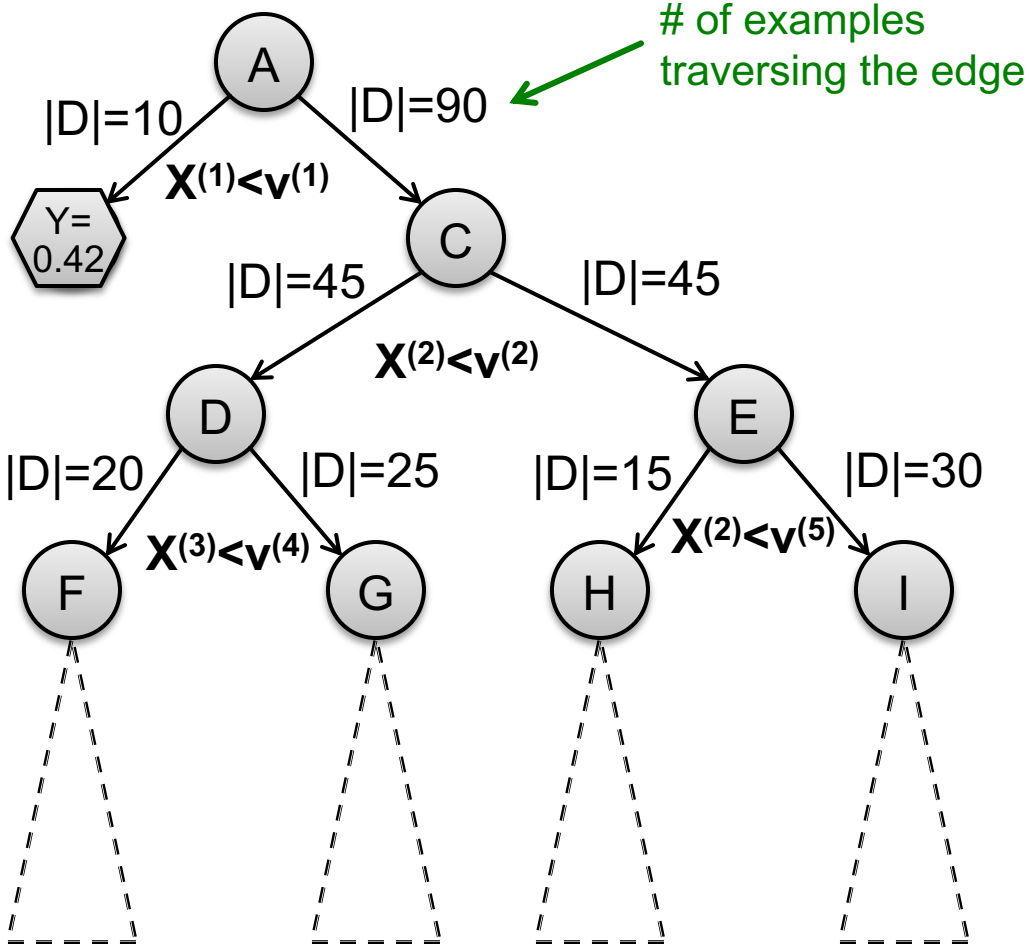
- **Alternative view:**





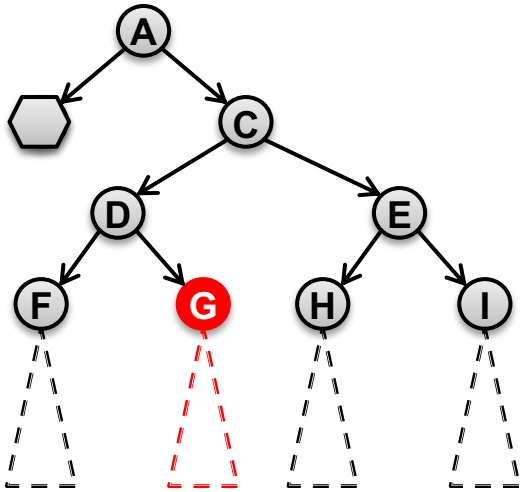
How to construct a tree?

- **Training dataset D^*, $|D^*| = 100$ examples**



How to construct a tree?

- Imagine we are currently at some node ***G***
  - Let D_G be the data that reaches ***G***
- **There is a decision we have to make: Do we continue building the tree?**
  - **If yes**, which variable and which value do we use for a **split**?
    - Continue building the tree recursively
  - **If not**, how do we make a prediction?
    - We need to build a **"predictor node"**



3 steps in constructing a tree

Algorithm 1 **BuildSubtree**

Require: Node n, Data $D \subseteq D^*$

1: $(n \rightarrow \text{split}, D_L, D_R) = \text{FindBestSplit}(D)$ (1)
2: **if** $\text{StoppingCriteria}(D_L)$ **then** (2)
3: $n \rightarrow \text{left\_prediction} = \text{FindPrediction}(D_L)$ (3)
4: **else**
5: **BuildSubtree** $(n \rightarrow \text{left}, D_L)$
6: **if** $\text{StoppingCriteria}(D_R)$ **then**
7: $n \rightarrow \text{right\_prediction} = \text{FindPrediction}(D_R)$
8: **else**
9: **BuildSubtree** $(n \rightarrow \text{right}, D_R)$

- **Requires at least a single pass over the data!**

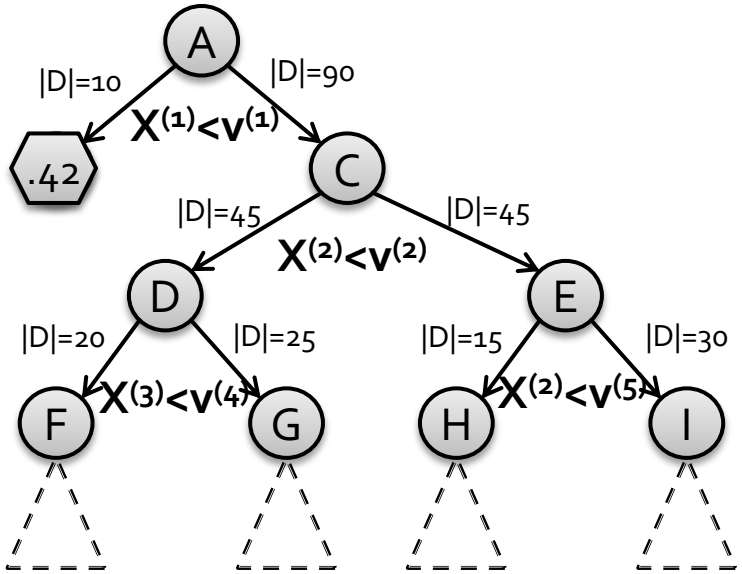
How to construct a tree?

(1) How to split? Pick attribute & value that optimizes some criterion

- **Regression: Purity**
  - **Find split** $(X^{(i)}, v)$ that creates D, D_L, D_R: parent, left, right child datasets and **maximizes**:

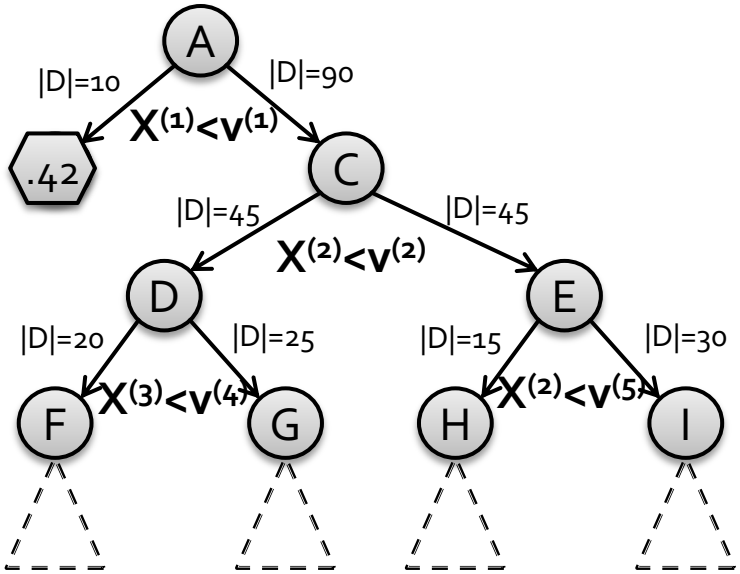
$$|D| \cdot Var(D) - \left(|D_L| \cdot Var(D_L) + |D_R| \cdot Var(D_R)\right)$$

  - $Var(D) = \frac{1}{|D|} \sum_{i \in D} (y_i - \bar{y})^2$ … variance of y_i in D



How to construct a tree?

(1) How to split? Pick attribute & value that optimizes some criterion



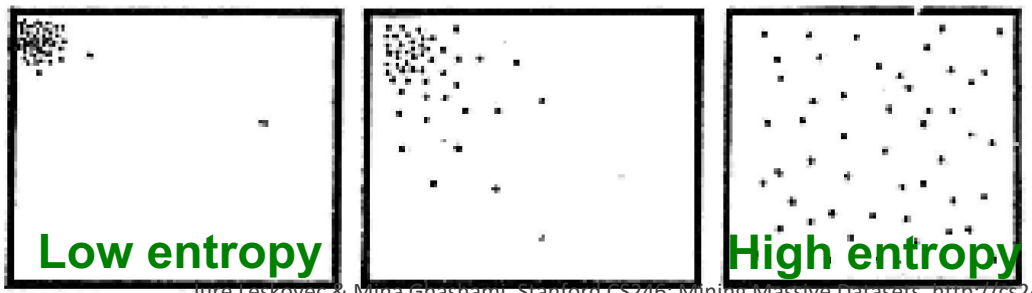
- **Classification:**
  Information Gain
  - **Measures how much a given attribute X tells us about the class Y**
  - $IG(Y \mid X)$: We must transmit Y over a binary link. How many bits on average would it save us if both ends of the line knew X?

$$IG(Y|X) = H(Y) - H(Y \mid X)$$

Why Information Gain? Entropy

Entropy:
Consider random variable $\boldsymbol{X} = \{\boldsymbol{X_1}, \ldots, \boldsymbol{X_m}\}$
What's the smallest possible number of bits, on average, per symbol, that we need to transmit a stream of symbols drawn from $\boldsymbol{X}$'s distribution?
The entropy of $\boldsymbol{X}$: $\boldsymbol{H(X) = -\sum_{j=1}^{m} p(X_j) \log p(X_j)}$

- **"High Entropy":** $\boldsymbol{X}$ is from a uniform (boring) distribution
  - A histogram of the frequency distribution of values of $\boldsymbol{X}$ is **flat**
- **"Low Entropy":** $\boldsymbol{X}$ is from a varied (peaks/valleys) distrib.
  - A histogram of the frequency distribution of values of $\boldsymbol{X}$ would have many lows and one or two highs



Why Information Gain? Entropy

- **Suppose I want to predict Y and I have input X**
  - X = College Major
  - Y = Likes "Casablanca"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

- **From this data we estimate**
  - $P(Y = Yes) = 0.5$
  - $H(Y) = -½\log_2(½) - ½\log_2(½) = \mathbf{1}$
  - $P(X = CS) = 0.25$
  - $P(X = History) = 0.25$
  - $P(X = Math) = 0.5$
  - $H(X) = \sum_{j=1}^{m} p(X_j)\, log\, p(X_j) = 1.5$

Why Information Gain? Entropy

- **Suppose I want to predict Y and I have input X**
  - X = College Major
  - Y = Likes "Casablanca"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

- **Def: Specific Conditional Entropy**
  - $H(Y \mid X = v)$ = The entropy of Y among only those records in which X has value v
  - **Example:**
    - $H(Y|X = Math) = 1$
    - $H(Y|X = History) = 0$
    - $H(Y|X = CS) = 0$

Why Information Gain?

- **Suppose I want to predict Y and I have input X**
  - X = College Major
  - Y = Likes "Casablanca"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

- **Def: Conditional Entropy**
  - $H(Y \mid X)$ = The average specific conditional entropy of Y
    - = if you choose a record at random what will be the conditional entropy of Y, conditioned on that row's value of X
    - = Expected number of bits to transmit Y if both sides knew the value of X
    - = $\sum_j P(X = v_j) H(Y|X = v_j)$

Why Information Gain?

- **Suppose I want to predict Y and I have input X**
  - $H(Y \mid X)$ = The average specific conditional entropy of Y

$$= \sum_j P(X = v_j) H(Y|X = v_j)$$

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

  - **Example:**

V_j	$P(X=v_j)$	$H(Y\|X=v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

  - **So:** H(Y|X)=0.5*1+0.25*0+0.25*0 = **0.5**

Why Information Gain?

- **Suppose I want to predict Y and I have input X**
  - **Def: Information Gain**
    - $IG(Y|X)$ = I must transmit Y. **How many bits on average would it save me if both ends of the line knew X?**

$$IG(Y|X) = H(Y) - H(Y \mid X)$$

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

  - **Example:**
    - H(Y) = 1
    - H(Y|X) = 0.5
    - Thus **IG(Y|X) = 1 – 0.5 = 0.5**

What is Information Gain used for?

- **Suppose you are trying to predict whether someone is going to live past 80 years**
- From historical data you might find:
  - $IG(LongLife \mid HairColor) = 0.01$
  - $IG(LongLife \mid Smoker) = 0.4$
  - $IG(LongLife \mid Gender) = 0.25$
  - $IG(LongLife \mid LastDigitOfSSN) = 0.00001$
- **IG tells us how much information about Y is contained in X**
  - **So attribute X that has high $IG(Y|X)$ is a good split!**

3 steps in constructing a tree

Algorithm 1 **BuildSubtree**

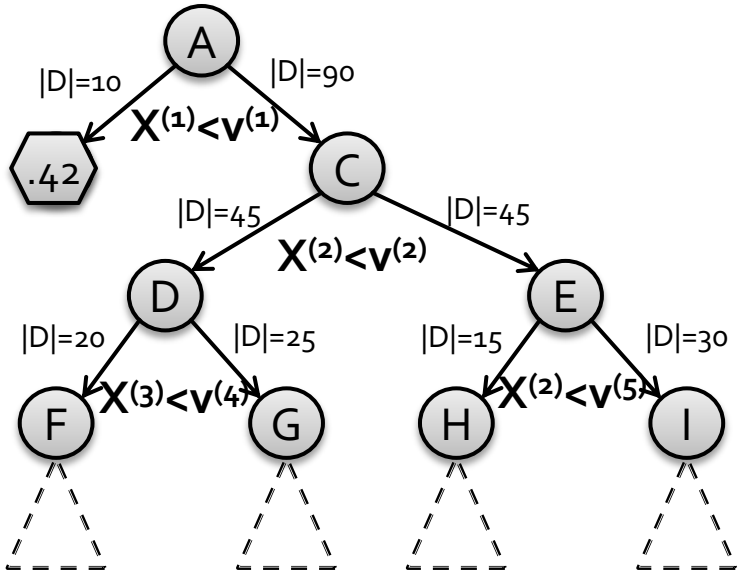
Require: Node n, Data $D \subseteq D^*$

1: $(n \rightarrow \text{split}, D_L, D_R) = \text{FindBestSplit}(D)$ (1)
2: **if** $\text{StoppingCriteria}(D_L)$ **then** (2)
3: $n \rightarrow \text{left\_prediction} = \text{FindPrediction}(D_L)$ (3)
4: **else**
5: **BuildSubtree** $(n \rightarrow \text{left}, D_L)$
6: **if** $\text{StoppingCriteria}(D_R)$ **then**
7: $n \rightarrow \text{right\_prediction} = \text{FindPrediction}(D_R)$
8: **else**
9: **BuildSubtree** $(n \rightarrow \text{right}, D_R)$

When to stop?

(2) When to stop?

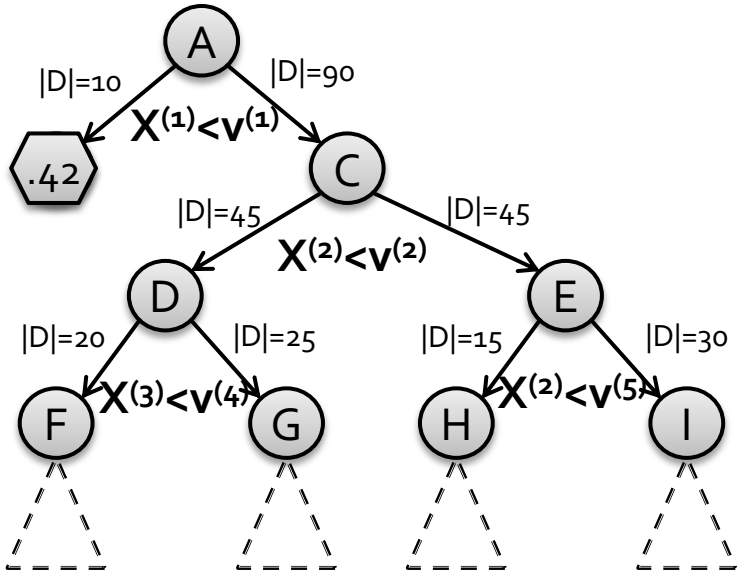
- Many different heuristic options
- **Two ideas:**
  - **(1) When the leaf is "pure"**
    - The target variable does not vary too much: $\boldsymbol{Var(y) < \varepsilon}$
  - **(2) When # of examples in the leaf is too small**
    - For example, $\boldsymbol{|D| \leq 100}$



How to predict?

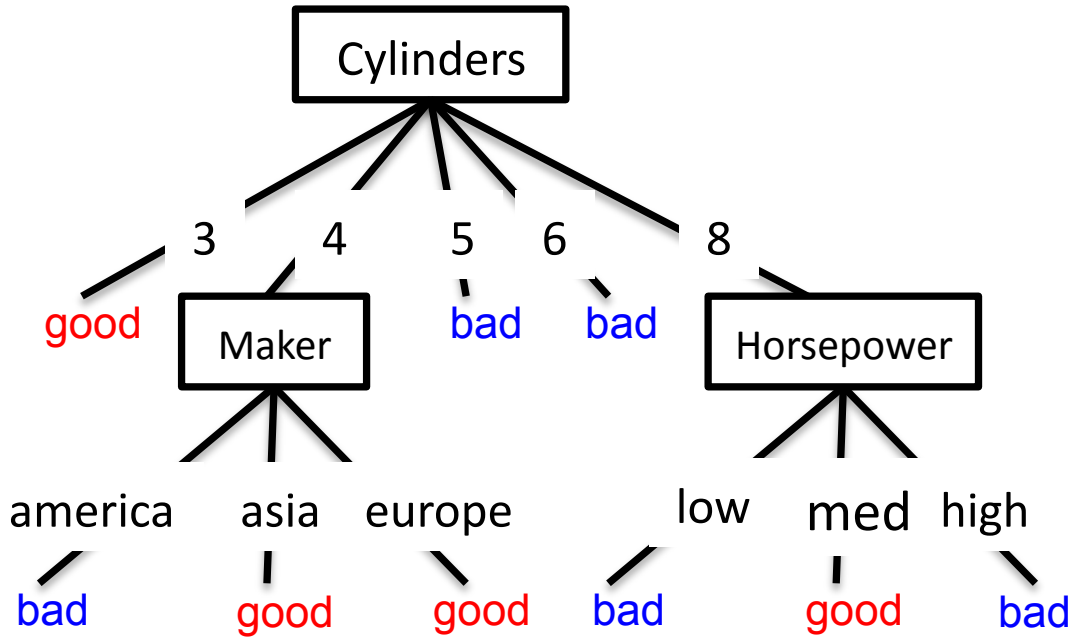
(3) How to predict?

- **Many options**
  - **Regression:**
    - Predict average y_i of the examples in the leaf
    - Build a linear regression model on the examples in the leaf
  - **Classification:**
    - Predict most common y_i of the examples in the leaf



Predicting Miles Per Gallon

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe



Two questions:
- Binary splits?
- Continuous (real-value) feature?

Slides adapted from David Sontag

Real-Valued inputs

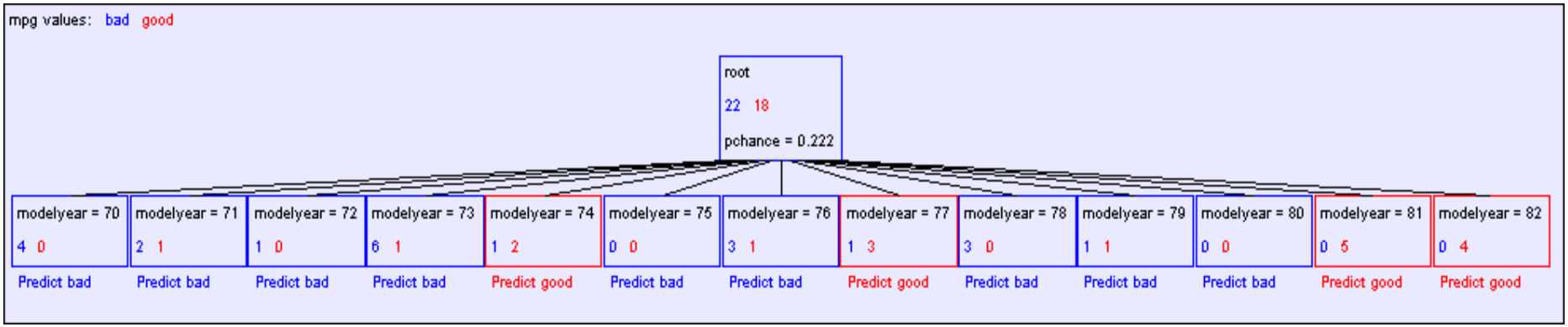
What should we do if some of the inputs are real-valued?

Infinite number of possible split values!!!

mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	america
bad	8	455	225	4425	10	70	america
good	4	107	86	2464	15.5	76	europe
bad	5	131	103	2830	15.9	78	europe

Slides adapted from David Sontag

“One branch for each numeric value” idea:

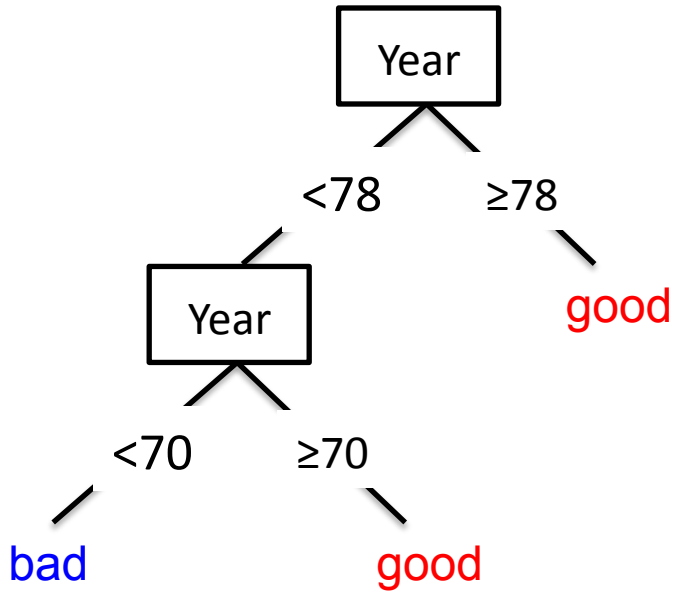


Hopeless: hypothesis with such a high branching factor will shatter *any* dataset and overfit

Slides adapted from David Sontag

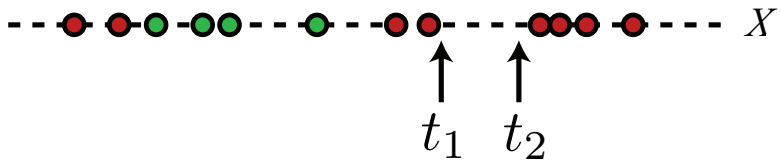
Threshold splits

- **Binary tree:** split on attribute X at value t
  - One branch: X < t
  - Other branch: X ≥ t
- **Requires small change**
  - Allow repeated splits on same variable **along a path**

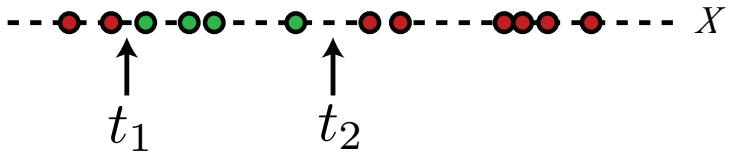


The set of possible thresholds

- Binary tree, split on attribute X
  - One branch: X < t
  - Other branch: X ≥ t
- Search through possible values of *t*
  - Seems hard!!!
- But only a finite number of *t*'s are important:



  - Sort data according to X into $\{x_1,\ldots,x_m\}$
  - Consider split points of the form $x_i + (x_{i+1} - x_i)/2$
  - Morever, only splits between examples of different classes matter!



(Figures from Stuart Russell)

Picking the best threshold

- Suppose *X* is real valued with threshold *t*
- Want **IG(Y | X:t)**, the information gain for Y when testing if *X* is greater than or less than *t*
- Define:
  - $H(Y|X:t) = p(X < t)\, H(Y|X < t) + p(X >= t)\, H(Y|X >= t)$
  - $IG(Y|X:t) = H(Y) - H(Y|X:t)$
  - $IG^*(Y|X) = \max_t IG(Y|X:t)$
- Use: $IG^*(Y|X)$ for continuous variables

Slides adapted from David Sontag

Decision Trees: Conclusion

Decision Trees

- **Characteristics**
  - **Classification & Regression**
    - Multiple (~10) classes
  - **Real valued and categorical features**
  - **Few (hundreds) of features**
  - Usually **dense features**
  - **Complicated decision boundaries**
    - Early stopping to avoid overfitting!
- **Example applications**
  - User profile classification
  - Landing page bounce prediction

Decision Trees

- **Decision trees are the single most popular data mining tool:**
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
  - It's possible to mitigate overfitting (i.e., with ensemble methods)
  - **They do classification as well as regression!**