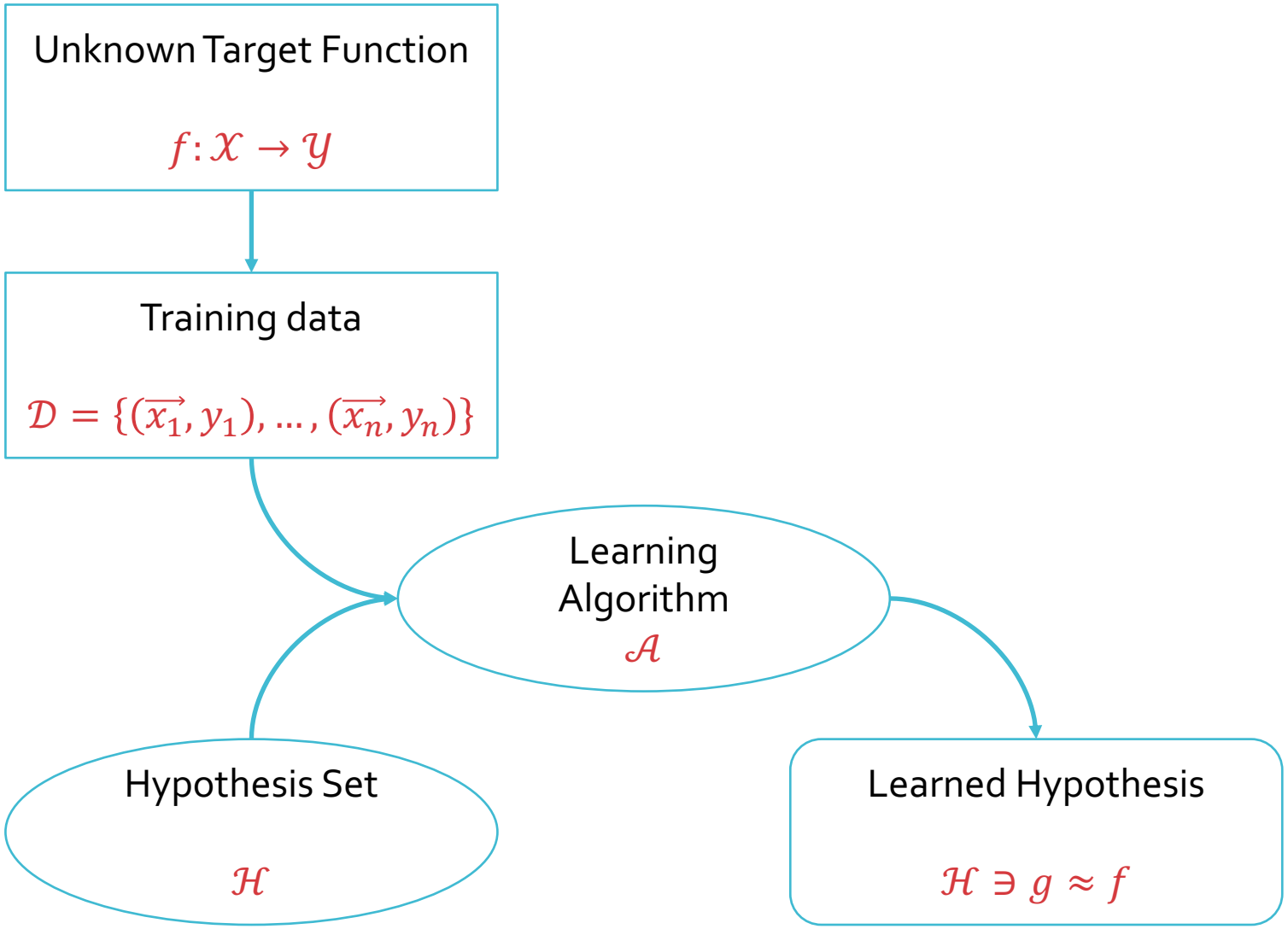
CSE 417T: Introduction to Machine Learning

Lecture 11: Review

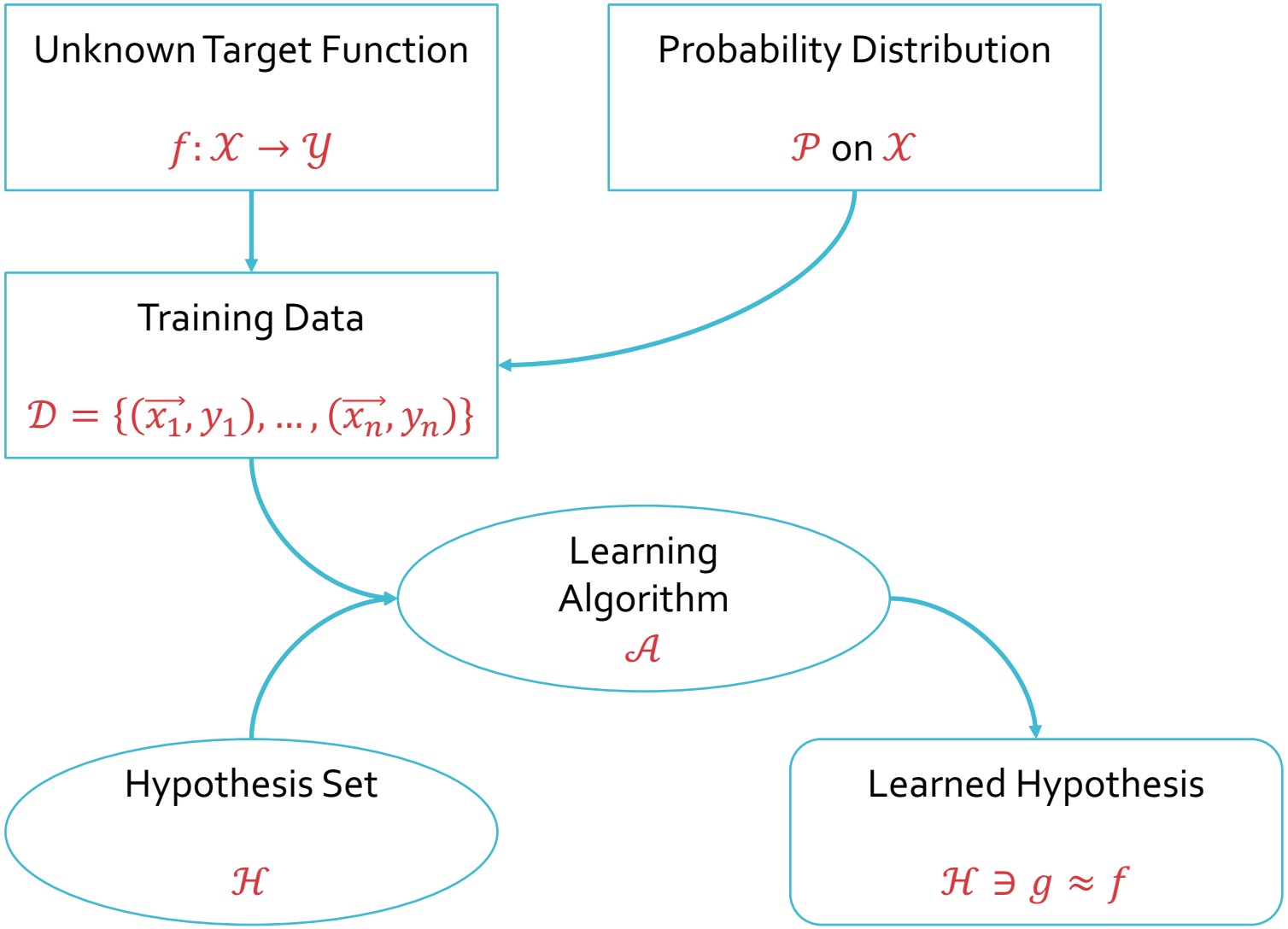
Henry Chai

10/02/18

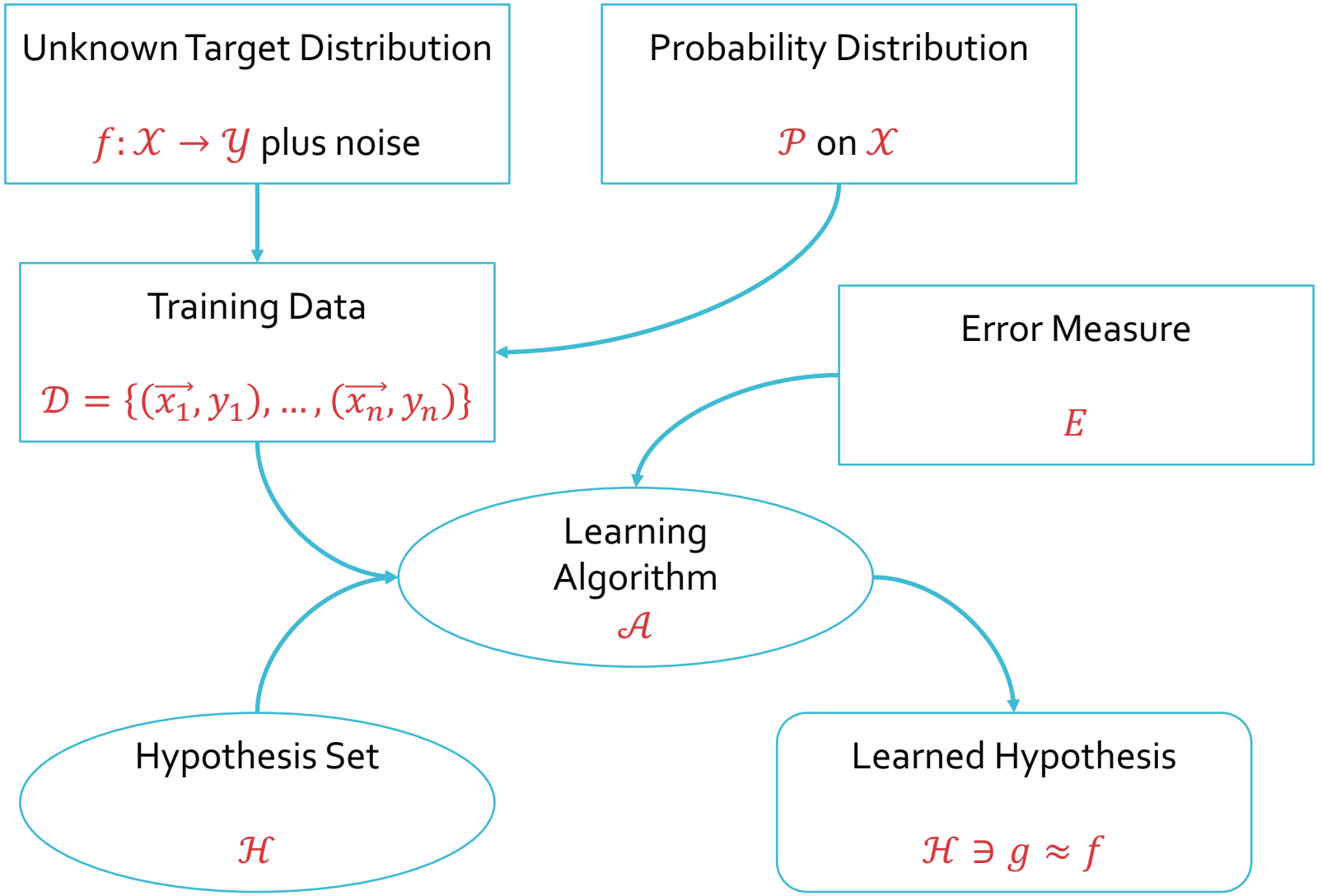
Formal Setup



Formal Setup



Formal Setup



Hoeffding's Inequality (Validation)

- For a given hypothesis, h
- $E_{in}(h)$ = in-sample error of h
- $E_{out}(h)$ = out-of-sample error of h

- $P\{|E_{in}(h) - E_{out}(h)| > \epsilon\} \leq 2e^{-2\epsilon^2 n}$

- As n increases, RHS decreases
- As ϵ decreases, RHS increases

Hoeffding's Inequality (Corrected)

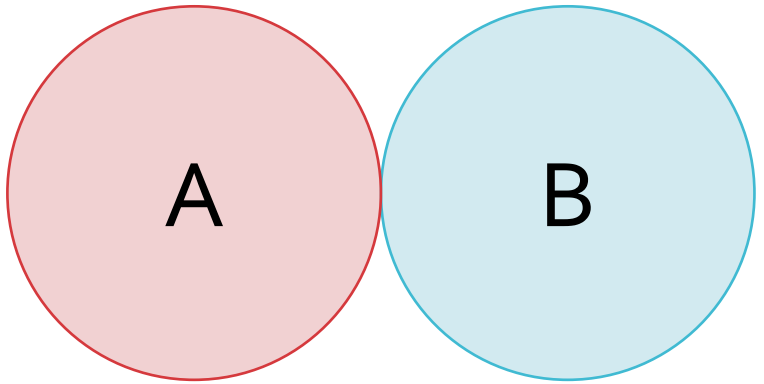
- For a given finite hypothesis set, $\mathcal{H} = \{h_1, \dots, h_m\}$
- $E_{in}(g)$ = in-sample error of best hypothesis in $\mathcal{H}$
- $E_{out}(g)$ = out-of-sample error of best hypothesis in $\mathcal{H}$

- $P\{|E_{in}(g) - E_{out}(g)| > \epsilon\} \leq 2(m)e^{-2\epsilon^2 n}$

- As n increases, RHS decreases
- As ϵ decreases, RHS increases
- As m increases, RHS increases

The Union Bound is bad

$$P\{A \cup B\} = P\{A\} + P\{B\} - P\{A \cap B\}$$



Dichotomy

- Given some finite sample of points $(\overrightarrow{x_1}, \dots, \overrightarrow{x_n})$ from the input space and single hypothesis $h \in \mathcal{H}$, applying h to each point in $(\overrightarrow{x_1}, \dots, \overrightarrow{x_n})$ results in a **dichotomy**
  - $\left(h(\overrightarrow{x_1}), \dots, h(\overrightarrow{x_n})\right)$ is a vector of n +1's and -1's
- Given $(\overrightarrow{x_1}, \dots, \overrightarrow{x_n})$, each hypothesis in $\mathcal{H}$ generates a dichotomy but not necessarily a unique dichotomy!
  - The set of dichotomies induced by $\mathcal{H}$ on $(\overrightarrow{x_1}, \dots, \overrightarrow{x_n})$ is $\mathcal{H}(\overrightarrow{x_1}, \dots, \overrightarrow{x_n}) = \left\{ \left(h(\overrightarrow{x_1}), \dots, h(\overrightarrow{x_n})\right) \mid h \in \mathcal{H} \right\}$

Growth Function

- The **growth function** of $\mathcal{H}$ is the largest number of dichotomies $\mathcal{H}$ can induce across all data sets of size n

- $m_{\mathcal{H}}(n) = \max_{(\overrightarrow{x_1},\ldots,\overrightarrow{x_n}) \in X} |\mathcal{H}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})|$

Growth Function (Shattering)

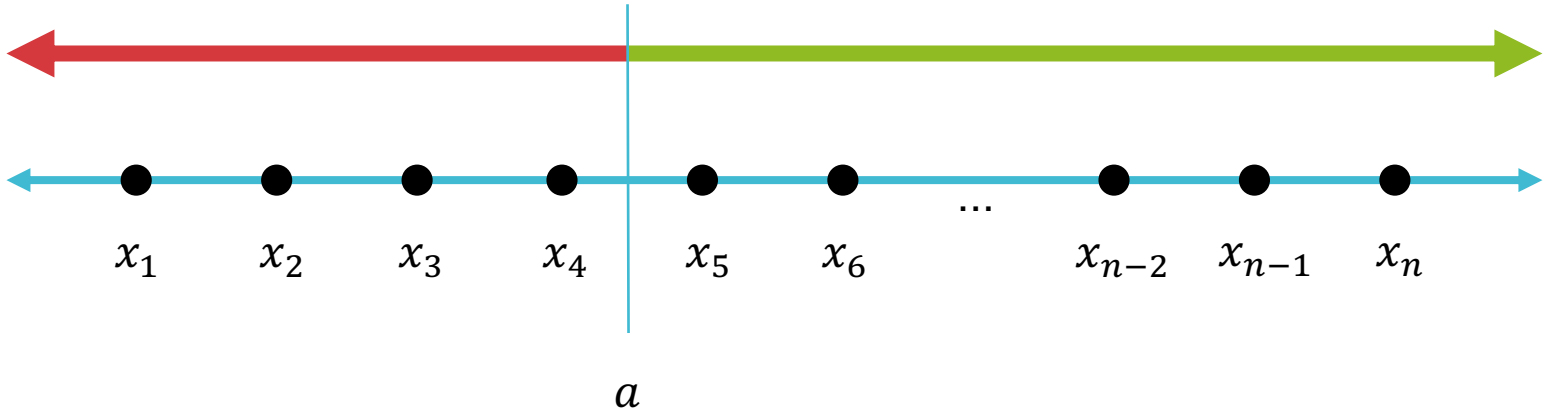
- Observe that $m_{\mathcal{H}}(n) \leq 2^n \; \forall \; \mathcal{H}$ and n
- Given $\mathcal{H}$, if $\exists \; (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) \in \mathcal{X}$ s.t. $|\mathcal{H}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})| = 2^n$, then $\mathcal{H}$ **shatters** $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})$
- If $\exists \; (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) \in \mathcal{X}$ that is shattered by $\mathcal{H}$, then $m_{\mathcal{H}}(n) = 2^n$

Growth Function (Break Points)

- If $m_{\mathcal{H}}(k) < 2^k$, then k is a break point for $\mathcal{H}$
- If there is at least one break point for $\mathcal{H}$, then $m_{\mathcal{H}}(n)$ is polynomial in n

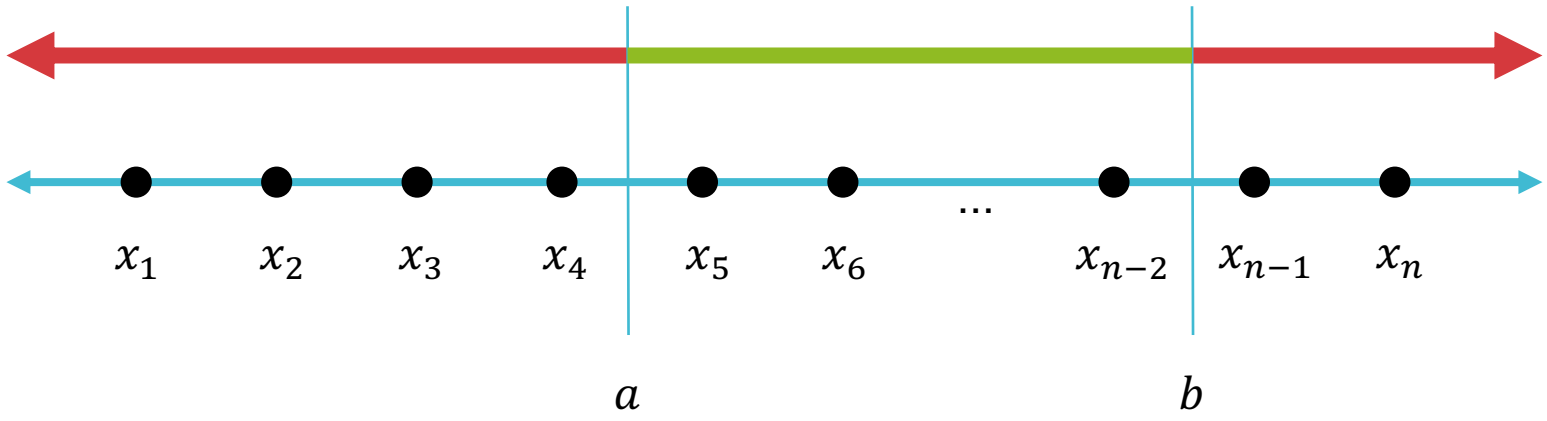
Growth Function: Example

- $\mathcal{X} = \mathbb{R}$ and $\mathcal{H}$ = Positive rays: $h(x) = sign(x - a)$
- $m_{\mathcal{H}}(n) = n + 1$



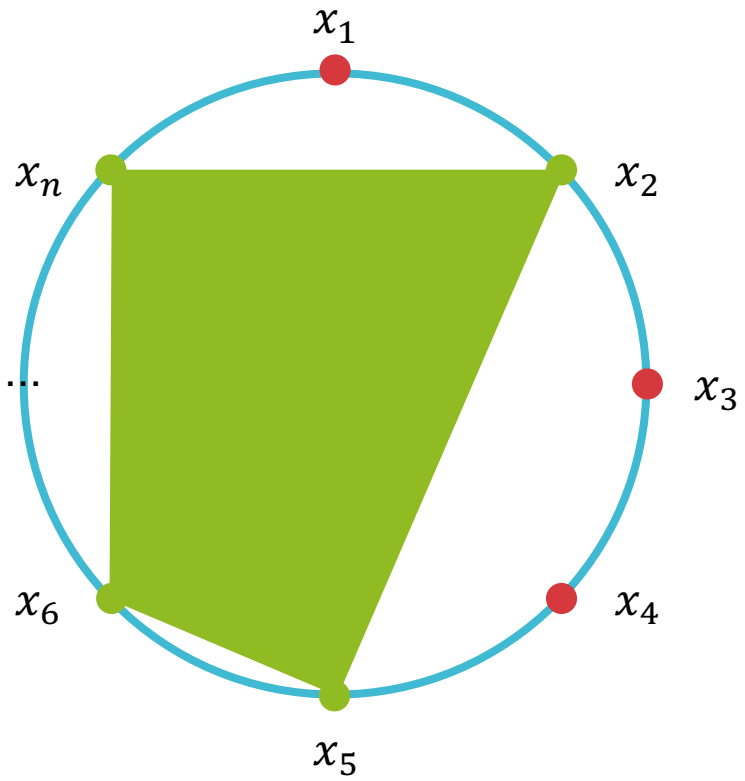
Growth Function: Example

- $\mathcal{X} = \mathbb{R}$ and $\mathcal{H} =$ Positive intervals
- $m_{\mathcal{H}}(n) = \binom{n+1}{2} + 1$



Growth Function: Example

- $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{H} =$ Convex sets
- $m_{\mathcal{H}}(n) = 2^n$



Vapornik-Chervonenkis (VC)-Bound

$$P\{|E_{in}(g) - E_{out}(g)| > \epsilon\} \leq 4m_{\mathcal{H}}(2n)e^{-\frac{1}{8}\epsilon^2 n}$$

Or…

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{n}\log\left(\frac{4m_{\mathcal{H}}(2n)}{\delta}\right)}$$

with probability at least $1 - \delta$

VC-Dimension

- $d_{VC}(\mathcal{H}) =$ the largest value of n s.t. $m_{\mathcal{H}}(n) = 2^n$
- The VC-dimension is the greatest number of points that can be shattered by $\mathcal{H}$
- If k^* is the smallest breakpoint for $\mathcal{H}$, then $d_{VC}(\mathcal{H}) = k^* - 1$
- $m_{\mathcal{H}}(n) \leq n^{d_{VC}} + 1$
- $E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{VC}\frac{\log(n)}{n}}\right)$

Sample Complexity

- How many samples do we need in our training data to say that the generalization error is less than ϵ with probability at least $1 - \delta$?

- Set $\sqrt{\frac{8}{n}\log\left(\frac{4\left(1+(2n)^{d_{VC}}\right)}{\delta}\right)} \leq \epsilon$

- Conclude that we need $n \geq \frac{8}{\epsilon^2}\log\left(\frac{4\left(1+(2n)^{d_{VC}}\right)}{\delta}\right)$

- As δ decreases, RHS increases
- As ϵ decreases, RHS increases
- As d_{VC} decreases, RHS decreases

Penalty for Model Complexity

- Given n samples, how good can we say our learned hypothesis will do with confidence at least $1 - \delta$?

- Conclude that $E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{n} \log\left(\frac{4\left((2n)^{d_{VC}}+1\right)}{\delta}\right)}$

Approximation Generalization Tradeoff

How well does g generalize?

$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{VC}\frac{\log(n)}{n}}\right)$$

How well does g approximate f?

Approximation Generalization Tradeoff

Increases as d_{VC} increases

$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{VC}\frac{\log(n)}{n}}\right)$$

Decreases as d_{VC} increases

Bias-Variance Tradeoff

How variable is g?

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g_{\mathcal{D}})] = \mathbb{E}_{\vec{x}}\left[\mathbb{E}_{\mathcal{D}}[g_{\mathcal{D}}(\vec{x})^2 - \bar{g}(\vec{x})^2] + \left(\bar{g}(\vec{x}) - f(\vec{x})\right)^2\right]$$

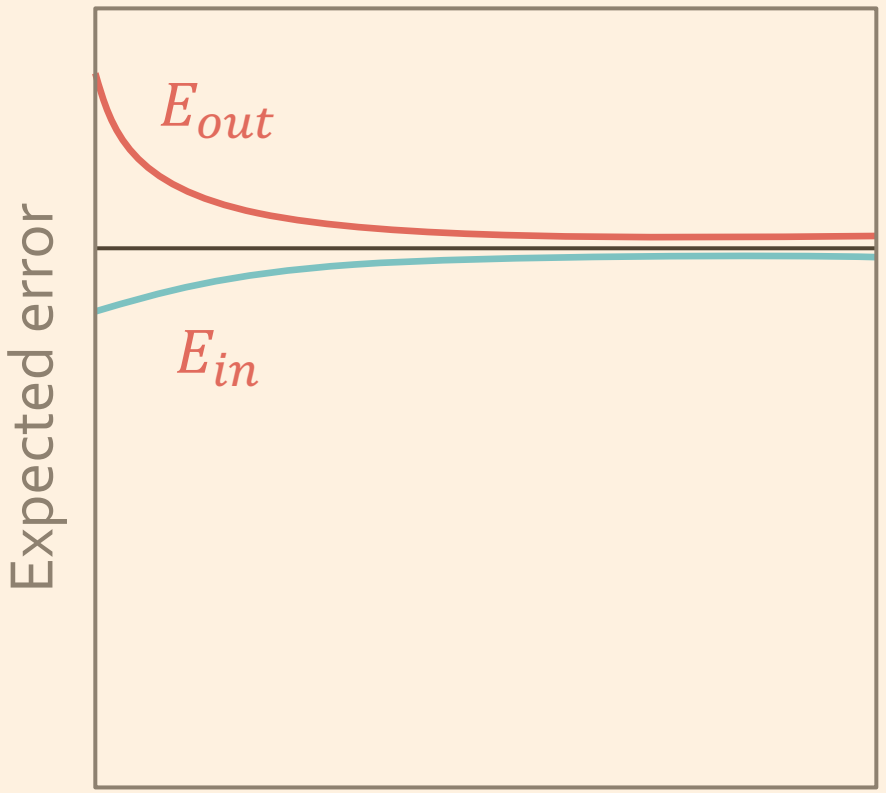
How well, on average, does g approximate f?

Bias-Variance Tradeoff

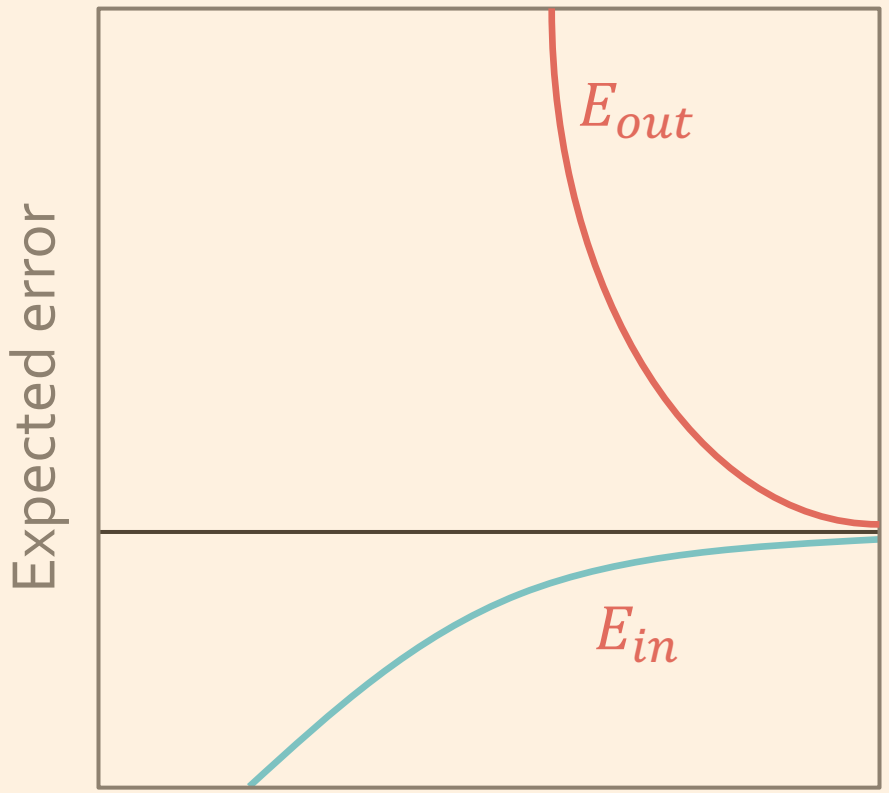
Increases as $\mathcal{H}$ becomes more complex

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g_{\mathcal{D}})] = \mathbb{E}_{\vec{x}}\left[\mathbb{E}_{\mathcal{D}}[g_{\mathcal{D}}(\vec{x})^2 - \bar{g}(\vec{x})^2] + \left(\bar{g}(\vec{x}) - f(\vec{x})\right)^2\right]$$

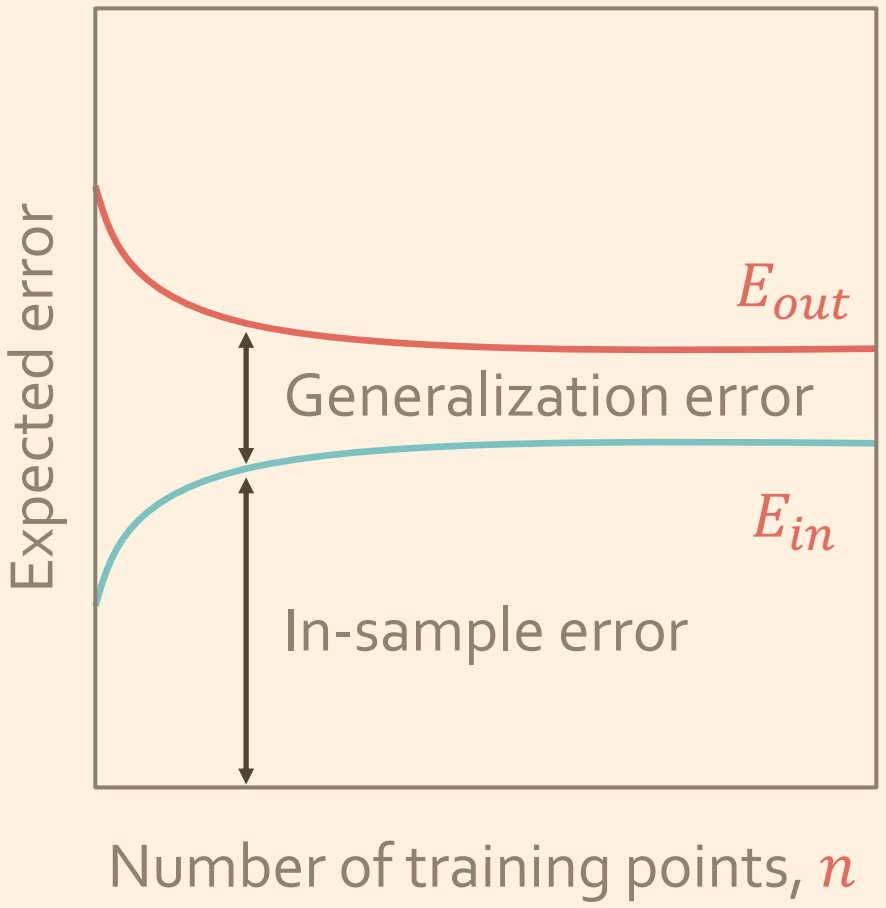
Decreases as $\mathcal{H}$ becomes more complex



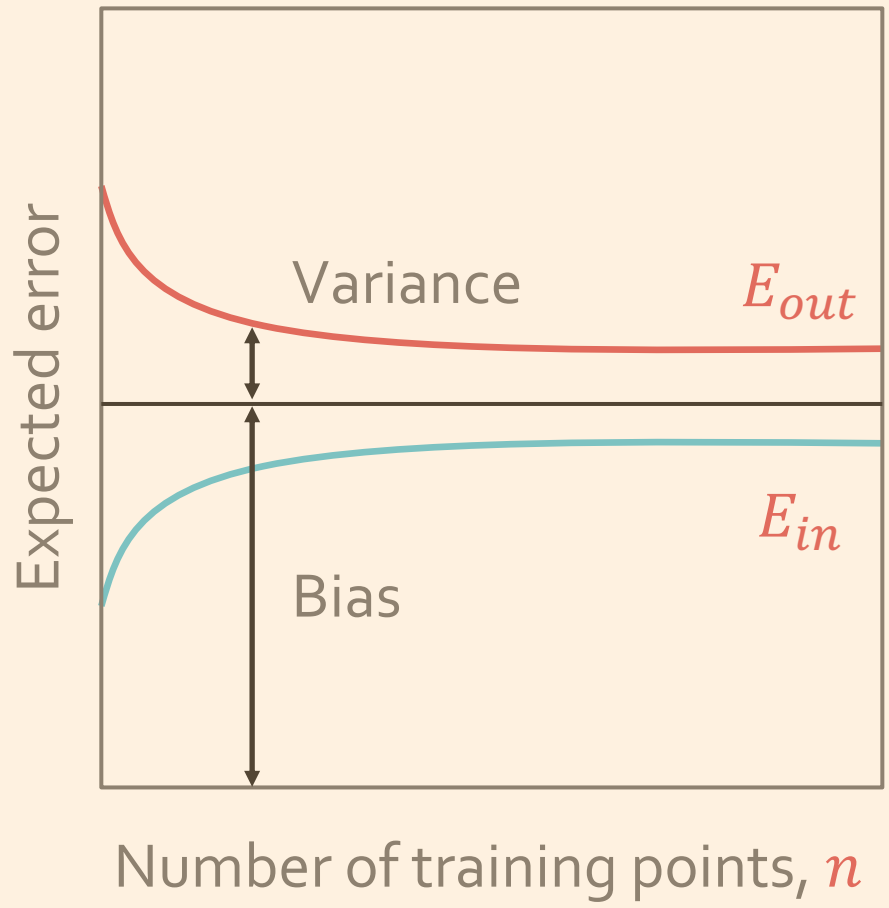
Simple model



Complex model



VC analysis



Bias-Variance analysis

Test Sets

- Instead of bounding $E_{out}(g)$ using $E_{in}(g)$, estimate $E_{out}(g)$ using the error on some test dataset $\mathcal{D}'$, $E_{test}(g)$

- If the $\mathcal{D}'$ is not involved in the training process, then we are validating g using $\mathcal{D}'$
  - Therefore, Hoeffding's bound applies with $m = |\mathcal{H}| = 1$
  - $P\{|E_{test}(g) - E_{out}(g)| > \epsilon\} \leq 2e^{-2\epsilon^2 n'}$ where $n' = |\mathcal{D}'|$

- As n' increases, $2e^{-2\epsilon^2 n'}$ decreases
- As n' increases, $E_{test}(g)$ increases

3 Learning Problems

Problem	Domain
Classification	$\mathcal{Y} = \{-1, +1\}$
Predicting Probabilities	$\mathcal{Y} = [0, 1]$
Regression	$\mathcal{Y} = \mathbb{R}$

Linear Models

$$h(\vec{x}) = \text{some function of } \vec{w}^T \vec{x}$$

$$\vec{x} = [1\ x_1\ x_2\ \dots\ x_d]$$

3 Learning Solutions

Problem	Model
Linear Classification	$h(\vec{x}) = sign(\vec{w}^T \vec{x})$
Logistic Regression	$h(\vec{x}) = \theta(\vec{w}^T \vec{x})$
Linear Regression	$h(\vec{x}) = \vec{w}^T \vec{x}$

Linear Classification

- Perceptron
  - Given some input $\vec{x} = (x_0 = 1, x_1, \ldots, x_d)$:

$$h(\vec{x}) = sign\left(\sum_{i=0}^{d} w_i x_i\right)$$

Perceptron Learning Algorithm

- PLA finds a linear separator in finite time, **if the data is linearly separable**

- Given: training data $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_n}, y_n)\}$
- Initialize $\vec{w}$ to all zeros or (small) random numbers
- While ∃ some misclassified training example i.e.

  $(\overrightarrow{x_i}, y_i) \in \mathcal{D}$ s.t. $h(\overrightarrow{x_i}) = sign(\vec{w}^{\mathrm{T}}\overrightarrow{x_i}) \neq y_i$

  - Randomly pick a misclassified training example, $(\vec{x}, y)$
  - Update $\vec{w}$: $\vec{w} = \vec{w} + y\vec{x}$

Perceptron Learning Algorithm

- Suppose $(\vec{x}, y) \in \mathcal{D}$ is a misclassified training example and $y = +1$
  - $\vec{w}^{\mathrm{T}}\vec{x}$ is negative
  - After updating $\vec{w} = \vec{w} + y\vec{x}$, $(\vec{w} + y\vec{x})^{\mathrm{T}}\vec{x} = \vec{w}^{\mathrm{T}}\vec{x} + y\vec{x}^{\mathrm{T}}\vec{x}$ is less negative than $\vec{w}^{\mathrm{T}}\vec{x}$
    - Because $y > 0$ and $\vec{x}^{\mathrm{T}}\vec{x} > 0$
- A similar argument holds if $y = -1$

Linear Regression: Squared Error

$$E_{in}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{w}^T\vec{x_i} - y_i)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x_i}^T\vec{w} - y_i\right)^2$$

$$= \frac{1}{n}\|X\vec{w} - \vec{y}\|^2 \text{ where } \|\vec{z}\| = \sqrt{\sum_{i=1}^{d} z_i^2} = \sqrt{\vec{z}^T\vec{z}}$$

$$= \frac{1}{n}(X\vec{w} - \vec{y})^{\mathrm{T}}(X\vec{w} - \vec{y})$$

Minimizing Error

- Find the gradient
- Set it equal to zero
- Solve
- (Check that the solution is a minimum)

Minimizing Error

$$E_{in}(\vec{w}) = \frac{1}{n}(X\vec{w} - \vec{y})^{\mathrm{T}}(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}\left((X\vec{w})^{T} - \vec{y}^{\mathrm{T}}\right)(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}(\vec{w}^{T}X^{T}X\vec{w} - 2\vec{w}^{T}X\vec{y} + \vec{y}^{T}\vec{y})$$

$$\rightarrow \nabla_{\vec{w}}E_{in}(\vec{w}^{*}) = \frac{1}{n}(2X^{T}X\vec{w}^{*} - 2X^{T}\vec{y}) = 0$$

$$\rightarrow 2X^{T}X\vec{w}^{*} - 2X^{T}\vec{y} = 0$$

$$\rightarrow X^{T}X\vec{w}^{*} = X^{T}\vec{y}$$

$$\rightarrow \vec{w}^{*} = (X^{T}X)^{-1}X^{T}\vec{y}$$

Checking

$$\nabla_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X\vec{w} - 2X^T\vec{y})$$

$$H_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X)$$

$\rightarrow H_{\vec{w}} E_{in}(\vec{w})$ is (almost always) positive definite

$\rightarrow \vec{w}^* = (X^T X)^{-1} X^T \vec{y}$ is a unique global minimum

Linear Regression Algorithm

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_n}, y_n)\}$

1. Construct X and $\vec{y}$
2. Compute the pseudo-inverse of $X = X^{\dagger} = (X^T X)^{-1} X^T$
3. Compute $\vec{w}^* = \mathrm{X}^{\dagger}\vec{y}$

- Output: $\vec{w}^*$

Linear Regression for Classification

- Key observation: $\{-1, +1\} \subset \mathbb{R}$
- Use linear regression to find $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$
- $\vec{w}^*$ minimizes $E_{in}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (\vec{w}^T \vec{x_i} - y_i)^2$
- In general, $sign\left(\vec{w}^{*T} \vec{x_i}\right) \approx y_i \; \forall \; i$
- Use $\vec{w}^*$ for linear classification: $g(\vec{x}) = sign\left(\vec{w}^{*T} \vec{x}\right)$

The Pocket Algorithm

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_n}, y_n)\},\ T$

1. Initialize $\vec{w}$ to all zeros and $E_{best} = \infty$
2. For $t = 1, 2, \dots, T$
   a. Randomly pick a misclassified training example, $(\vec{x}, y)$
   b. Update $\vec{w}$: $\vec{w} = \vec{w} + y\vec{x}$
   c. If $E_{in}(\vec{w}) < E_{best}$
      I. $E_{best} = E_{in}(\vec{w})$
      II. $\vec{w}^* = \vec{w}$

- Output: $\vec{w}^*$

Logistic Regression

- Training data **does not** consist of probabilities
- Observations are still **binary**: $y_i = \pm 1$
- Goal is to learn $f(\vec{x}) = P\{y = +1|\vec{x}\}$
- $h(\vec{x}) = \theta(\vec{w}^T\vec{x}) = \frac{1}{1+e^{-\vec{w}^T\vec{x}}} = \left(1 + e^{-\vec{w}^T\vec{x}}\right)^{-1} \in [0,1]$
- Note that $1 - \theta(\vec{w}^T\vec{x}) = \theta(-\vec{w}^T\vec{x})$

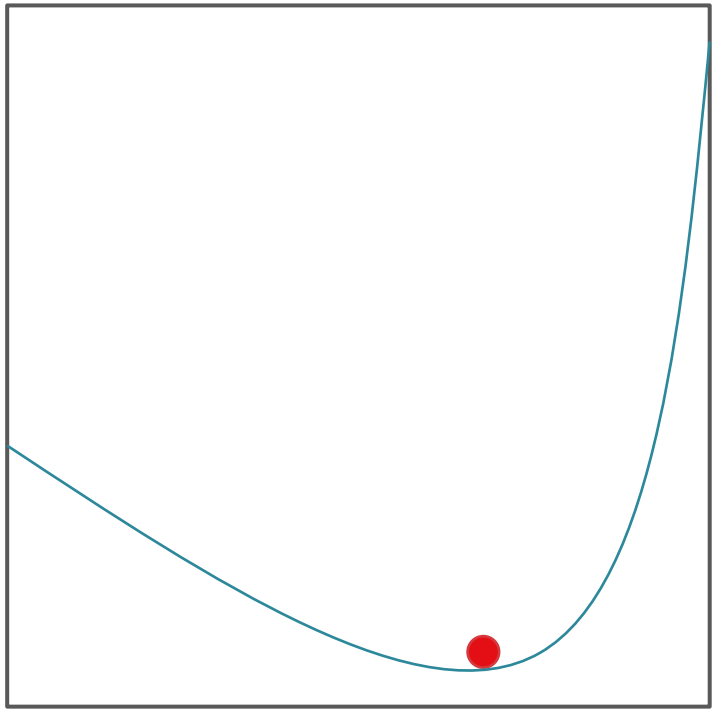
Cross-entropy Error

- Some hypothesis h is good if:
  - the probability of the training data $\mathcal{D}$ given h is high

- $E_{in}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} \ln\left(1 + e^{-y_i \vec{w}^T \vec{x_i}}\right)$

Gradient Descent: Intuition

- Iterative method for minimizing functions
- Requires the gradient to exist everywhere
- Particularly useful for minimizing convex functions, like the cross-entropy error



Gradient Descent: Intuition

- Suppose the current location is $\vec{w}_{(t)}$
- Move some distance, η, in the "most downhill" direction possible, $\hat{v}$
- $\vec{w}_{(t+1)} = \vec{w}_{(t)} + \eta\hat{v}$

$\hat{v}$

- Fix η and choose $\hat{v}$ to minimize ΔE_{in} after making the update $\vec{w}_{(t+1)} = \vec{w}_{(t)} + \eta\hat{v}$

- $\Delta E_{in}(\hat{v}) = E_{in}\left(\vec{w}_{(t)} + \eta\hat{v}\right) - E_{in}\left(\vec{w}_{(t)}\right)$

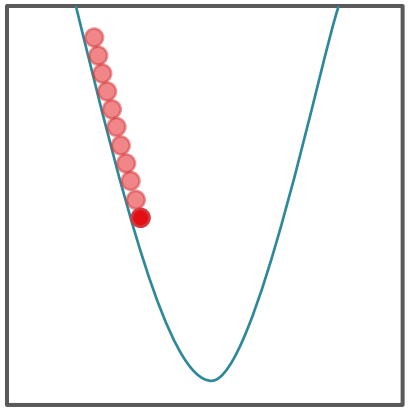
$$\approx \left(E_{in}\left(\vec{w}_{(t)}\right) + \eta\hat{v}^T\nabla_{\vec{w}}E_{in}\left(\vec{w}_{(t)}\right)\right) - E_{in}\left(\vec{w}_{(t)}\right)$$

$$\approx \eta\hat{v}^T\nabla_{\vec{w}}E_{in}\left(\vec{w}_{(t)}\right)$$

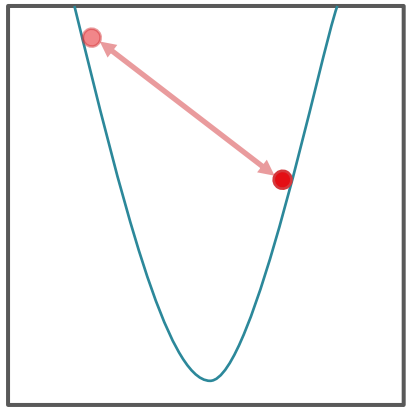
- $\Delta E_{in}(\hat{v}^*) = -\eta\left\|\nabla_{\vec{w}}E_{in}\left(\vec{w}_{(t)}\right)\right\|$

- $\hat{v}^* = -\frac{\nabla_{\vec{w}}E_{in}\left(\vec{w}_{(t)}\right)}{\left\|\nabla_{\vec{w}}E_{in}\left(\vec{w}_{(t)}\right)\right\|}$

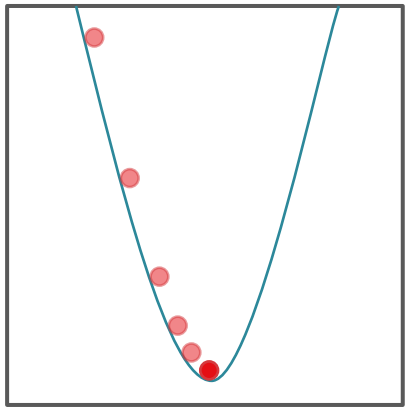
η_t



Small η



Large η



Variable η_t

- Set $\eta_t = \eta_0 \|\nabla_{\vec{w}} E_{in}(\vec{w}_t)\|$

- η_t decreases as t increases, because $\|\nabla_{\vec{w}} E_{in}(\vec{w}_t)\|$ decreases as $E_{in}(\vec{w}_t)$ approaches its minimum

Gradient Descent

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_n}, y_n)\}, \eta_0$

1. Initialize $\vec{w}_0$ to all zeros and set $t = 0$
2. While termination condition is not satisfied
   a. Compute $\nabla_{\vec{w}} E_{in}(\vec{w}_{(t)})$
   b. Update $\vec{w}$: $\vec{w}_{(t+1)} = \vec{w}_{(t)} - \eta_0 \nabla_{\vec{w}} E_{in}(\vec{w}_{(t)})$
   c. Increment t: $t = t + 1$

- Output: $\vec{w}_t \rightarrow g(\vec{x}) = P\{y = +1|\vec{x}\} = \frac{1}{1+e^{-\vec{w}_t^T \vec{x}}}$

Stochastic Gradient Descent (SGD)

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_n}, y_n)\}, \eta_0$

1. Initialize $\vec{w}_0$ to all zeros and set $t = 0$
2. While termination condition is not satisfied
   a. Pick a random data point in $\mathcal{D}$, $(\vec{x}, y)$
   b. Compute $\nabla_{\vec{w}} e\left(\vec{w}_{(t)}, \vec{x}, y\right)$
   c. Update $\vec{w}$: $\vec{w}_{(t+1)} = \vec{w}_{(t)} - \eta_0 \nabla_{\vec{w}} e\left(\vec{w}_{(t)}, \vec{x}, y\right)$
   d. Increment t: $t = t + 1$

- Output: $\vec{w}_t \rightarrow g(\vec{x}) = P\{y = +1|\vec{x}\} = \frac{1}{1+e^{-\vec{w}_t^T \vec{x}}}$

Logistic Regression for Classification

- Use logistic regression to find $\vec{w}_t$
- Use $\vec{w}_t$ for classification: if $P\{y = +1|\vec{x}\} = \theta(\vec{w}_t^T \vec{x}) \geq \frac{1}{2}$ then classify $\vec{x}$ as $+1$; otherwise, classify $\vec{x}$ as -1
- $g(\vec{x}) = sign\left(\frac{1}{1+e^{-\vec{w}_t^T \vec{x}}} - \frac{1}{2}\right)$

Logistic Regression for Classification

- Use logistic regression to find $\vec{w}_t$

- Use $\vec{w}_t$ for classification: if $P\{y = +1|\vec{x}\} = \theta(\vec{w}_t^T \vec{x}) \geq b$ then classify $\vec{x}$ as $+1$; otherwise, classify $\vec{x}$ as -1

- $g(\vec{x}) = sign\left(\frac{1}{1+e^{-\vec{w}_t^T \vec{x}}} - b\right)$

Error: Classification

- Fingerprint recognition:
  - Inputs are fingerprints
  - Outputs: +1 means "you", -1 means "not you"

- For personalized coupons:

		$f(\vec{x})$	
		+1	-1
$h(\vec{x})$	+1	0	1
	-1	100	0

For unlocking phones:

		$f(\vec{x})$	
		+1	-1
$h(\vec{x})$	+1	0	1000
	-1	1	0

Nonlinear Models

- Decide on a transformation $\Phi: \mathcal{X} \rightarrow \mathcal{Z}$
- Convert $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_n}, y_n)\}$ to $\widetilde{\mathcal{D}} = \{(\Phi(\overrightarrow{x_1}) = \overrightarrow{z_1}, y_1), \ldots, (\Phi(\overrightarrow{x_n}) = \overrightarrow{z_n}, y_n)\}$
- Fit a linear model using $\widetilde{\mathcal{D}}$, $\tilde{g}(\vec{z})$
- Return the corresponding predictor in the original space: $g(\vec{x}) = \tilde{g}\big(\Phi(\vec{x})\big)$

Tradeoffs

	Low-Dimensional Transformations	High-Dimensional Transformations
E_{in}	High	Low
Generalization	Good	Bad