

Machine Learning & Data Mining

CMS/CS/CNS/EE 155

Lecture 1:
Administrivia & Basics

Course Info

- Lecture (Tu/Th)
 - 2:30pm – 3:55pm in Beckman Institute Auditorium
- Recitation (Th)
 - Evening, location TBA
 - As needed
 - Usually 45-60 minutes
 - **First one on Thursday! (Introduction to Python)**

Staff



Matt
Levine



Alex
Cui



James
Deacon



Marcus
Dominguez-Kuhne



Alex
Guerra



Alice
Jin



Frank
Kou



Karthik
Nair



Jessica
Wang



Sherry
Wang



Erika
Yu



Albert
Zhai



Jim
Zhang



Eric
Zhao

Course Breakdown

- 6 Homeworks, ~60% of final grade
 - Due on Friday nights via Gradescope
 - **Homework 1 will be released tomorrow.**
 - Due following Wednesday @9pm
- Plan accordingly w/ CS144 & CS139!
- 3 Mini-projects, ~30% of final grade
- Final, ~10% of final grade

Regarding Homework 1

- If you have prior experience with CS 156
 - Should be pretty straightforward (4-5 hours)
- If you do not...
 - Might take a while (8-12 hours?)
 - But will mostly catch you up if you make it through
 - Should consider dropping class if too hard

Late Submission Policy

- Up to 48 free late hours
- Specify # late hours used when submitting

Course Etiquette

- Please ask questions during lecture!
 - I might defer some in interest of time
- If you arrive late, or need to leave early, please do so quietly.
- Updated collaboration policy

Course Website

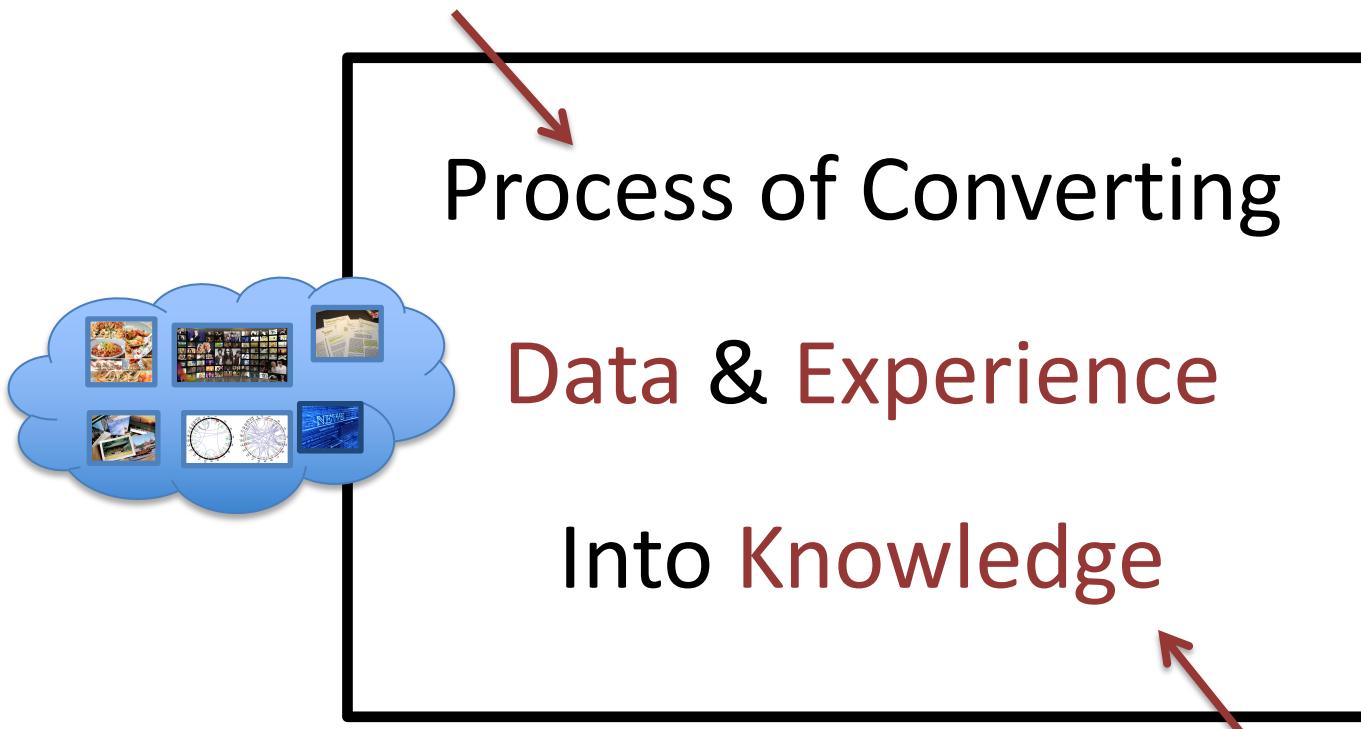
- <http://www.yisongyue.com/courses/cs155>
- Linked to from my website:
 - <http://www.yisongyue.com>
- Up-to-date office hours
- Lecture notes, additional reading, homework, etc.

Gradescope & Piazza

- Gradescope:
 - <https://www.gradescope.com/courses/77190>
 - Submission, Solutions, Grades
- Piazza
 - <https://piazza.com/class/k4c8ma7gjuy21q>
 - Course announcements
 - Q&A Forum (use it!)
- Lecture Videos
 - On YouTube (linked from course website)

Machine Learning & Data Mining

Computer Algorithm



Computer Model

Machine Learning vs Data Mining

- **ML focuses more on algorithms**
 - Typically more rigorous
 - Also on analysis (learning theory)
- **DM focuses more on knowledge extraction**
 - Typically uses ML algorithms
 - Knowledge should be human-understandable
- **Huge overlap**

Course Outline

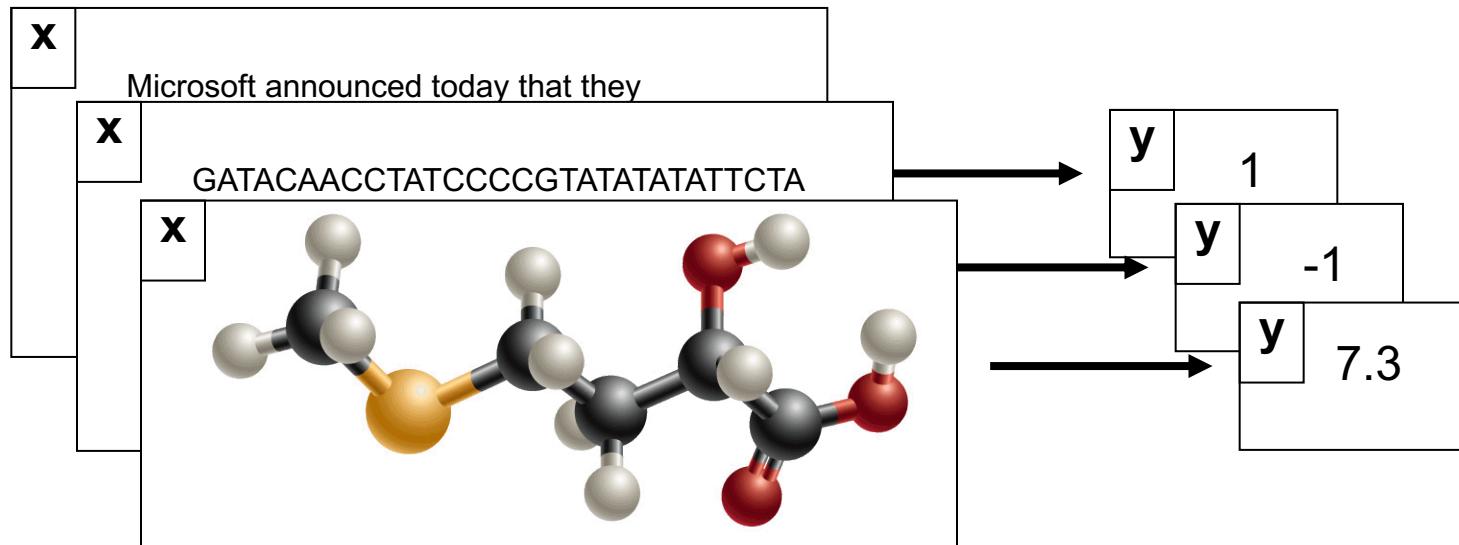
- Supervised Learning
 - 5 weeks
- Unsupervised Learning
 - 2 weeks
- Probabilistic Models
 - 2 weeks

Supervised Learning

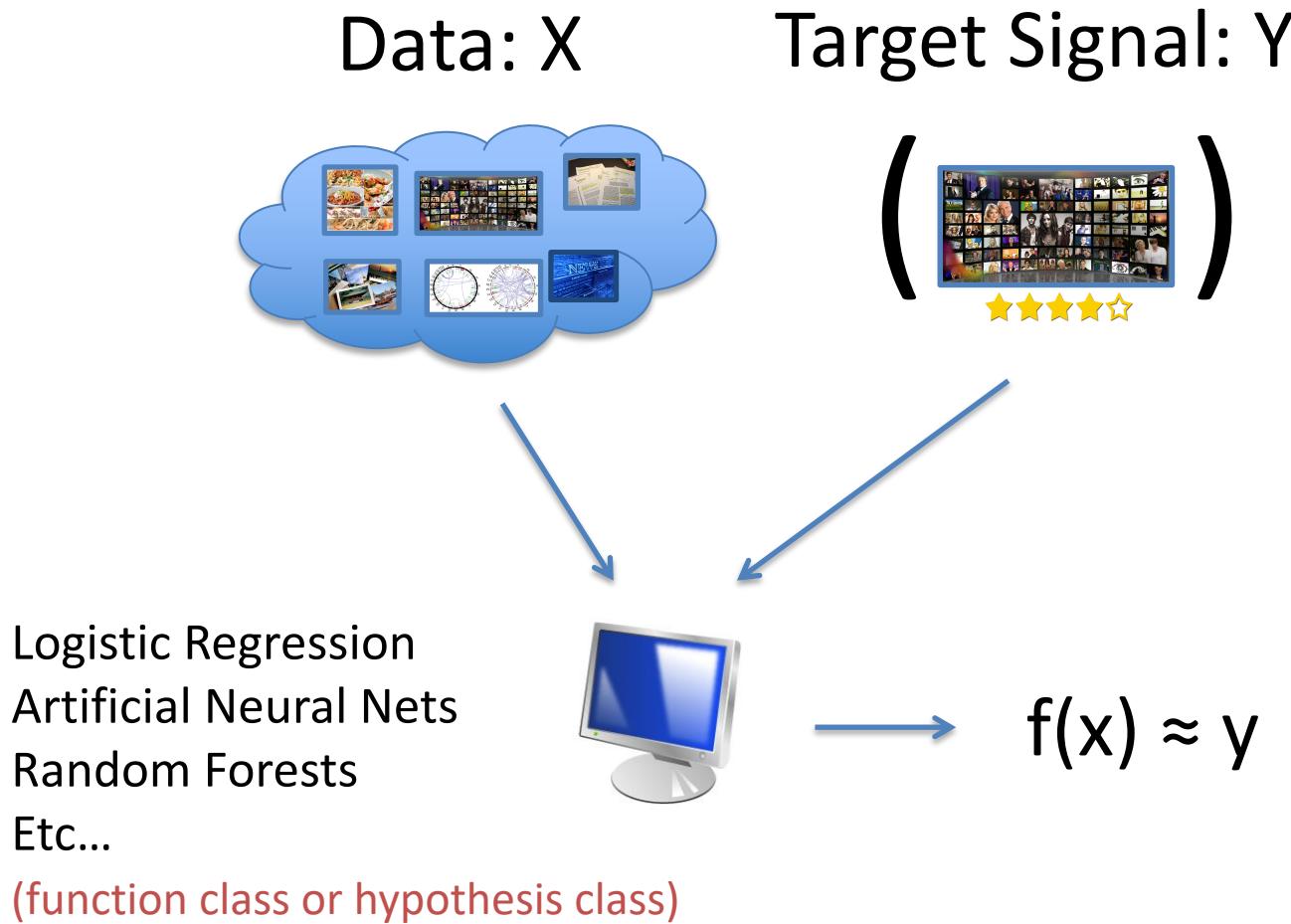
- Find function from input space X to output space Y

$$f : X \rightarrow Y \quad (\text{sometimes use } h)$$

such that the prediction error is low.

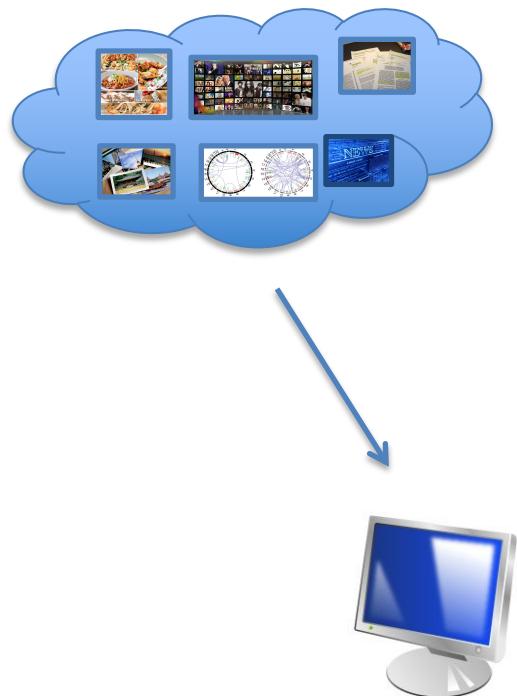


Supervised Learning



Aside: Unsupervised Learning

Data: X



No supervised target!

Learning goal is usually to find low-dimensional “summary” or reconstruction.

More on this later in course.

Example: Spam Filtering

- **Goal:** write a program to filter spam.

Viagra, Cialis,
Levitra

Reminder:
homework due
tomorrow.

Nigerian Prince
in Need of Help

SPAM!

NOT SPAM

SPAM!

Example: Spam Filtering

- **Goal:**

```
FUNCTION SpamFilter(string document)
{
    IF("Viagra" in document)
        RETURN TRUE
    ELSE IF("NIGERIAN PRINCE" in document)
        RETURN TRUE
    ELSE IF("Homework" in document)
        RETURN FALSE
    ELSE
        RETURN FALSE
    END IF
}
```

Viag
I

—
—
—
—

S

Prince
f Help

M!

Why is Spam Filtering Hard?

- Easy for humans to recognize
- Hard for humans to write down algorithm
- Lots of IF statements!

Machine Learning to the Rescue!

Training Set



SPAM!



SPAM!



NOT SPAM



NOT SPAM



SPAM!



SPAM!

:

Build a Generic Representation

Run a Generic Learning Algorithm

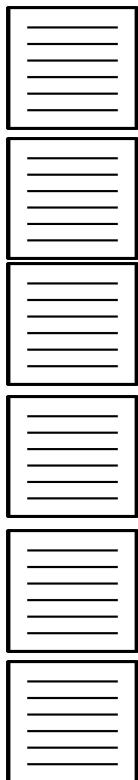
→ Classification Model

Labeled by Humans (“Supervision”)



Bag of Words Representation

Training Set



SPAM!

SPAM!

NOT SPAM

NOT SPAM

SPAM!

SPAM!

:

Bag of Words

(0,0,0,1,1,1)

(1,0,0,1,0,0)

(1,0,1,0,1,0)

(0,1,1,0,1,0)

(1,0,1,1,0,1)

(1,0,0,0,0,1)

“Feature Vector”

One feature for
each word in the
vocabulary

In practice 10k-1M

Linear Models

Let x denote the bag-of-words for an email

E.g., $x = (1, 1, 0, 0, 1, 1)$

“dot product” (linear algebra recitation)

Linear Classifier:

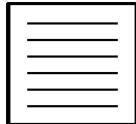
$$\begin{aligned} f(x | w, b) &= \text{sign}(w^T x - b) \\ &= \text{sign}(w_1 * x_1 + \dots + w_6 * x_6 - b) \end{aligned}$$

$$f(x|w,b) = \text{sign}(w^T x - b)$$

$$= \text{sign}(w_1 * x_1 + \dots w_6 * x_6 - b)$$

$w = (1,0,0,1,0,1)$
$b = 1.5$

Training Set



SPAM!



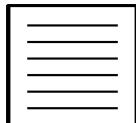
SPAM!



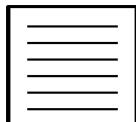
NOT SPAM



NOT SPAM



SPAM!



SPAM!

:

Bag of Words

(0,0,0,1,1,1)

$f(x|w,b) = +1$

(1,0,0,1,0,0)

$f(x|w,b) = +1$

(1,0,1,0,1,0)

$f(x|w,b) = -1$

(0,1,1,0,1,0)

$f(x|w,b) = -1$

(1,0,1,1,0,1)

$f(x|w,b) = +1$

(1,0,0,0,0,1)

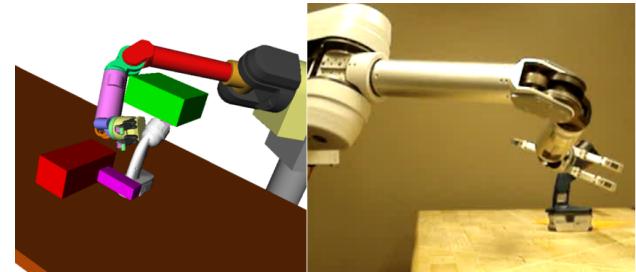
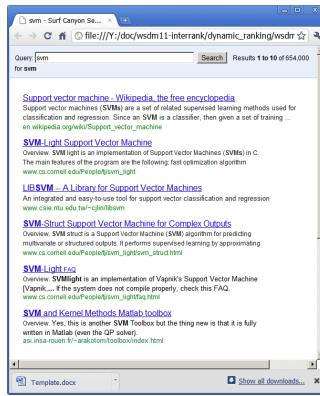
$f(x|w,b) = +1$

:

:

Linear Models

- Workhorse of Machine Learning



- By end of this lecture, you'll learn 75% how to build basic linear model.

Why Does Machine Learning Work?

- Repeated patterns in the data
 - Typically in the features
 - E.g., “Nigerian Prince” is indicative of spam
- Machine learning will find those patterns
 - Linear model over features
 - E.g., high weight on the words “Nigerian Prince”

Two Basic Supervised ML Problems

- **Classification** $f(x|w,b) = \text{sign}(w^T x - b)$
 - Predict which class an example belongs to
 - E.g., spam filtering example
- **Regression** $f(x|w,b) = w^T x - b$
 - Predict a real value or a probability
 - E.g., probability of being spam
- **Highly inter-related**
 - Train on Regression => Use for Classification

$$f(x|w,b) = w^T x - b$$

$$= w_1 * x_1 + \dots w_6 * x_6 - b$$

$w = (1,0,0,1,0,1)$
$b = 1.5$

Training Set



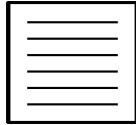
SPAM!



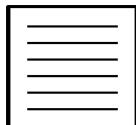
SPAM!



NOT SPAM



NOT SPAM



SPAM!



SPAM!

:

Bag of Words

(0,0,0,1,1,1)

$f(x|w,b) = +0.5$

(1,0,0,1,0,0)

$f(x|w,b) = +0.5$

(1,0,1,0,1,0)

$f(x|w,b) = -0.5$

(0,1,1,0,1,0)

$f(x|w,b) = -1.5$

(1,0,1,1,0,1)

$f(x|w,b) = +1.5$

(1,0,0,0,0,1)

$f(x|w,b) = +0.5$

:

:

Formal Definitions

- Training set: $S = \{(x_i, y_i)\}_{i=1}^N$ $x \in R^D$
 $y \in \{-1, +1\}$
- Model class: $f(x | w, b) = w^T x - b$ **Linear Models**
aka hypothesis class
- **Goal:** find (w, b) that predicts well on S .
 - How to quantify “well”?

Basic Supervised Learning Recipe

- Training Data:

$$S = \{(x_i, y_i)\}_{i=1}^N \quad \begin{aligned} x &\in R^D \\ y &\in \{-1, +1\} \end{aligned}$$

- Model Class:

$$f(x | w, b) = w^T x - b \quad \text{Linear Models}$$

- Loss Function:

$$L(a, b) = (a - b)^2 \quad \text{Squared Loss}$$

- Learning Objective:

$$\operatorname{argmin}_{w, b} \sum_{i=1}^N L(y_i, f(x_i | w, b))$$

Optimization Problem

Loss Function

- Measures penalty of mis-prediction:

- 0/1 Loss:

$$L(a, b) = 1_{[a \neq b]}$$

Classification

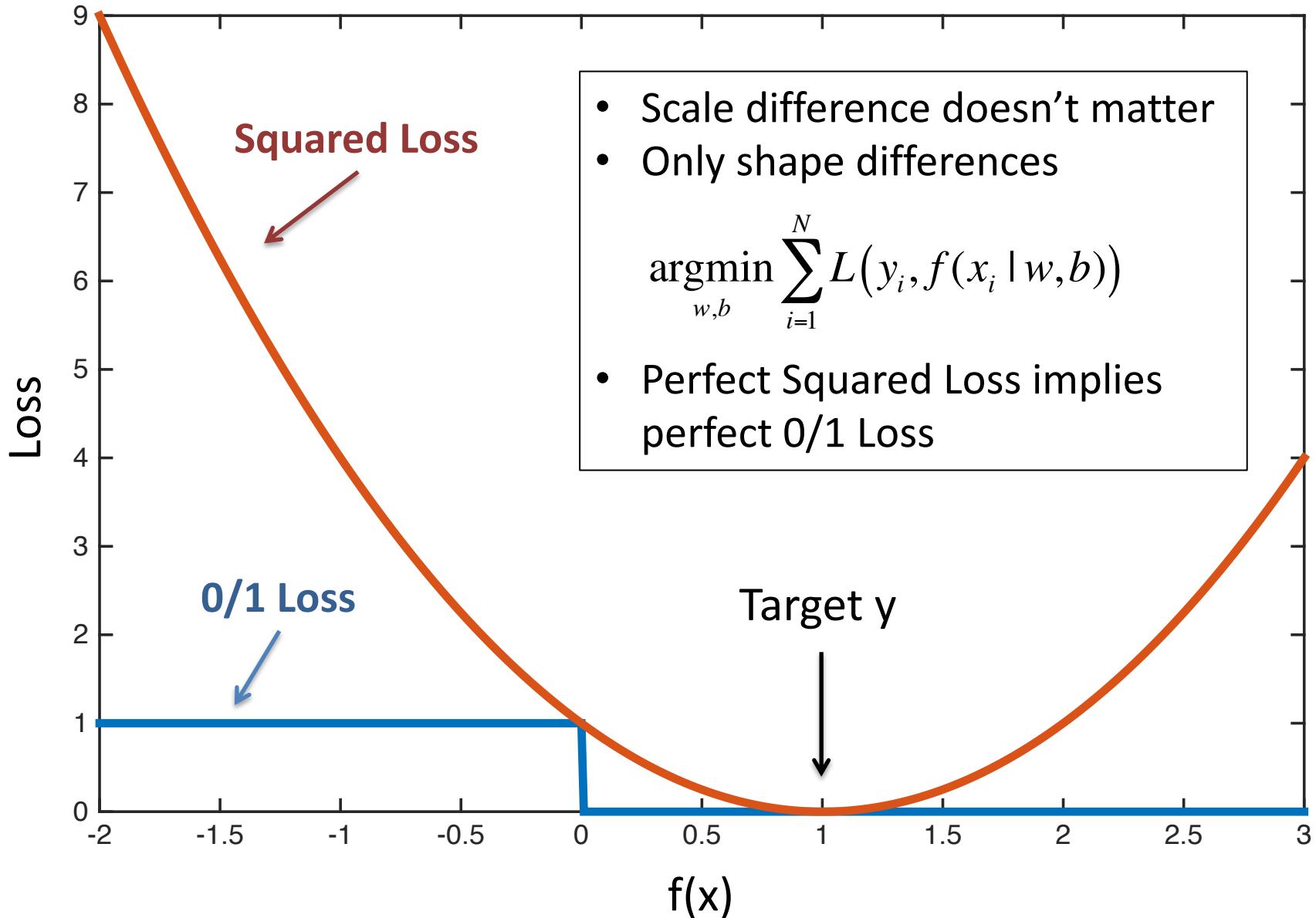
$$L(a, b) = 1_{[\text{sign}(a) \neq \text{sign}(b)]}$$

- Squared loss:

$$L(a, b) = (a - b)^2$$

Regression

- Substitute: $a=y$, $b=f(x)$



$$f(x|w,b) = w^T x - b$$

$$= w_1 * x_1 + \dots w_6 * x_6 - b$$

$w = (0.05, 0.05, -0.68, 0.68, -0.63, 0.68)$
$b = 0.27$

Training Set



SPAM!



SPAM!



NOT SPAM



NOT SPAM



SPAM!



SPAM!

Bag of Words

(0,0,0,1,1,1)

$f(x|w,b) = +1$

(1,0,0,1,0,0)

$f(x|w,b) = +1$

(1,0,1,0,1,0)

$f(x|w,b) = -1$

(0,1,1,0,1,0)

$f(x|w,b) = -1$

(1,0,1,1,0,1)

$f(x|w,b) = +1$

(1,0,0,0,0,1)

$f(x|w,b) = +1$

Train using Squared Loss

Learning Algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^N L(y_i, f(x_i | w, b))$$

- Typically, requires optimization algorithm.
- Simplest: **Gradient Descent**

**Loop for T
iterations**

$$w_{t+1} \leftarrow w_t - \partial_w \sum_{i=1}^N L(y_i, f(x_i | w_t, b_t))$$

$$b_{t+1} \leftarrow b_t - \partial_b \sum_{i=1}^N L(y_i, f(x_i | w_t, b_t))$$

Gradient Review

$$\partial_w \sum_{i=1}^N L(y_i, f(x_i | w, b))$$

$$= \sum_{i=1}^N \partial_w L(y_i, f(x_i | w, b))$$

$$= \sum_{i=1}^N -2(y_i - f(x_i | w, b)) \partial_w f(x_i | w, b)$$

$$= \sum_{i=1}^N -2(y_i - w^T x + b) x$$

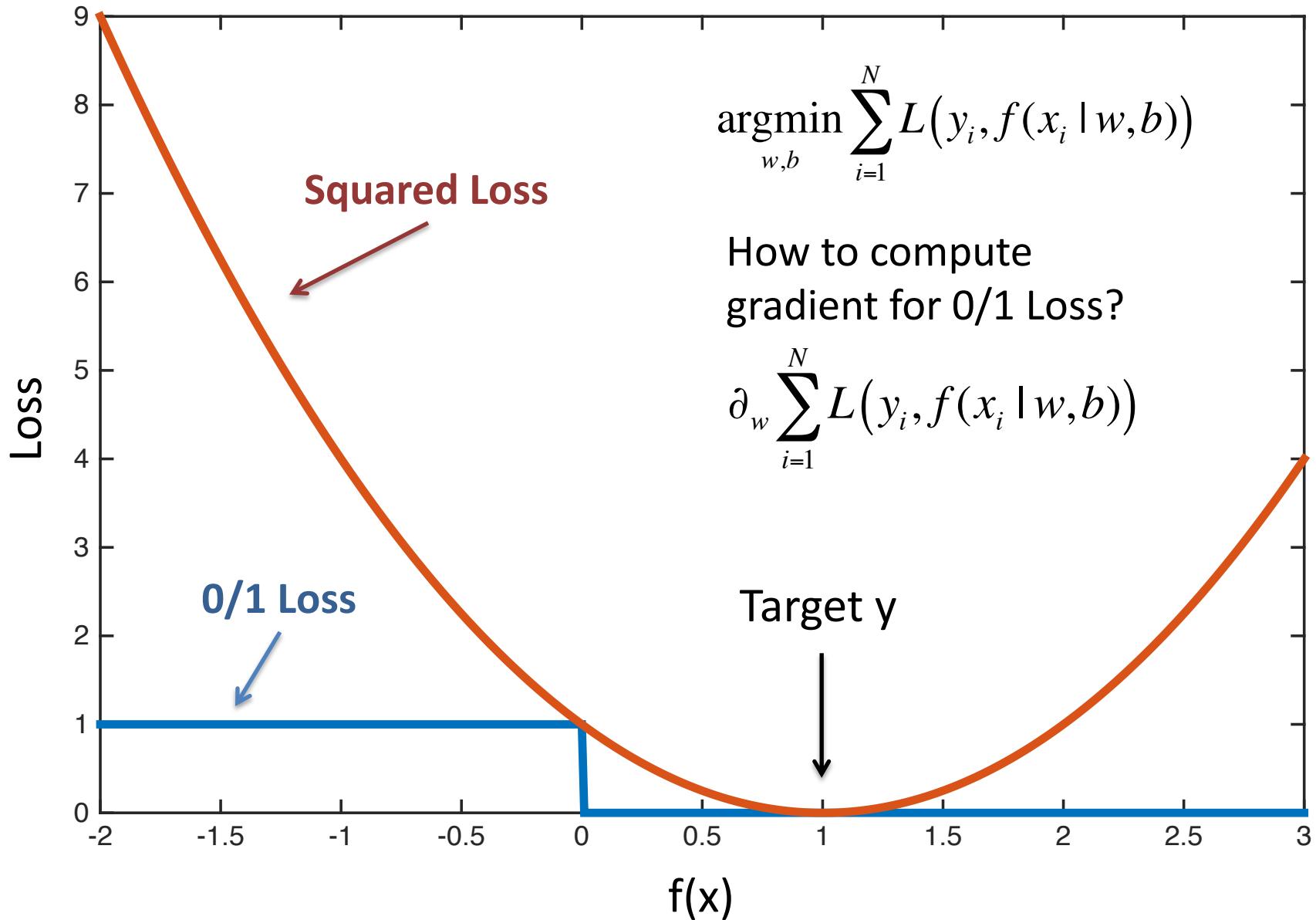
More Details
Next Lecture

Linearity of Differentiation

$$L(a, b) = (a - b)^2$$

Chain Rule

$$f(x | w, b) = w^T x - b$$



0/1 Loss is Intractable

- 0/1 Loss is flat or discontinuous everywhere
- VERY difficult to optimize
- **Solution:** Optimize smooth surrogate Loss
 - E.g., Squared Loss

Recap: Two Basic ML Problems

- **Classification** $f(x|w,b) = \text{sign}(w^T x - b)$
 - Predict which class an example belongs to
 - E.g., spam filtering example
- **Regression** $f(x|w,b) = w^T x - b$
 - Predict a real value or a probability
 - E.g., probability of being spam
- **Highly inter-related**
 - Train on Regression => Use for Classification

Recap: Supervised Learning Recipe

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^N$ $x \in R^D$
 $y \in \{-1, +1\}$
- Model Class: $f(x | w, b) = w^T x - b$ **Linear Models**
- Loss Function: $L(a, b) = (a - b)^2$ **Squared Loss**
- Learning Objective: $\operatorname{argmin}_{w, b} \sum_{i=1}^N L(y_i, f(x_i | w, b))$ **Optimization Problem**

Recap: Supervised Learning Recipe

- Training Data: $x \in R^D$, $y \in \{-1, +1\}$
 - Model: Linear Models
 - Loss Function: Squared Loss
 - Learning Objective:
Optimization Problem
- Congratulations!**
You now know the basic steps to training a model!
- But is your model any good?
- $$\operatorname{argmin}_{w,b} \sum_{i=1}^N L(y_i, f(x_i | w, b))$$

Optimization Problem

Example: Self-Driving Cars



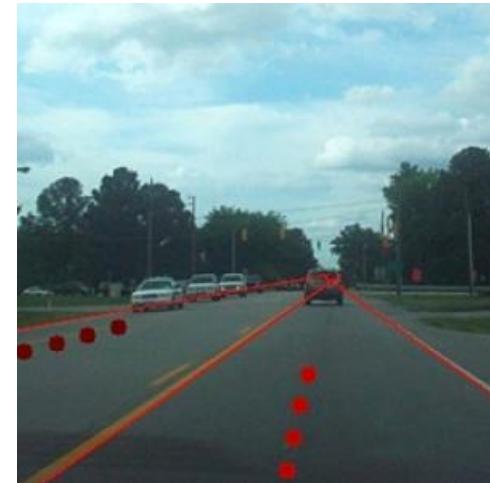
Basic Setup

- Mounted cameras
- Use image features
- Human demonstrations
- $f(x|w) = \text{steering angle}$
- Learn on training set



Overfitting

- Very accurate model
- But crashed on live test!
- Model w only cared about staying between two green patches



Test Error

- “True” distribution: $P(x,y)$ “All possible emails”
 - Unknown to us
- Train: $f(x) = y$
 - Using training data: $S = \{(x_i, y_i)\}_{i=1}^N$
 - Sampled identically and independently from $P(x,y)$
- Test Error:
$$L_P(f) = E_{(x,y) \sim P(x,y)} [L(y, f(x))]$$
 Prediction Loss on all possible emails
- Overfitting: Test Error >> Training Error

Test Error

- **Test Error:**

$$L_P(f) = E_{(x,y) \sim P(x,y)} [L(y, f(x))]$$

- **Treat f_S as random variable:** (randomness over S)

$$f_S = \operatorname{argmin}_{w,b} \sum_{(x_i, y_i) \in S} L(y_i, f(x_i | w, b))$$

- **Expected Test Error:**

$$E_S [L_P(f_S)] = E_S [E_{(x,y) \sim P(x,y)} [L(y, f_S(x))]]$$

Bias-Variance Decomposition

$$E_S[L_P(f_S)] = E_S[E_{(x,y) \sim P(x,y)}[L(y, f_S(x))]]$$

- For squared error:

$$E_S[L_P(f_S)] = E_{(x,y) \sim P(x,y)} \left[E_S[(f_S(x) - F(x))^2] + (F(x) - y)^2 \right]$$


$$F(x) = E_S[f_S(x)]$$

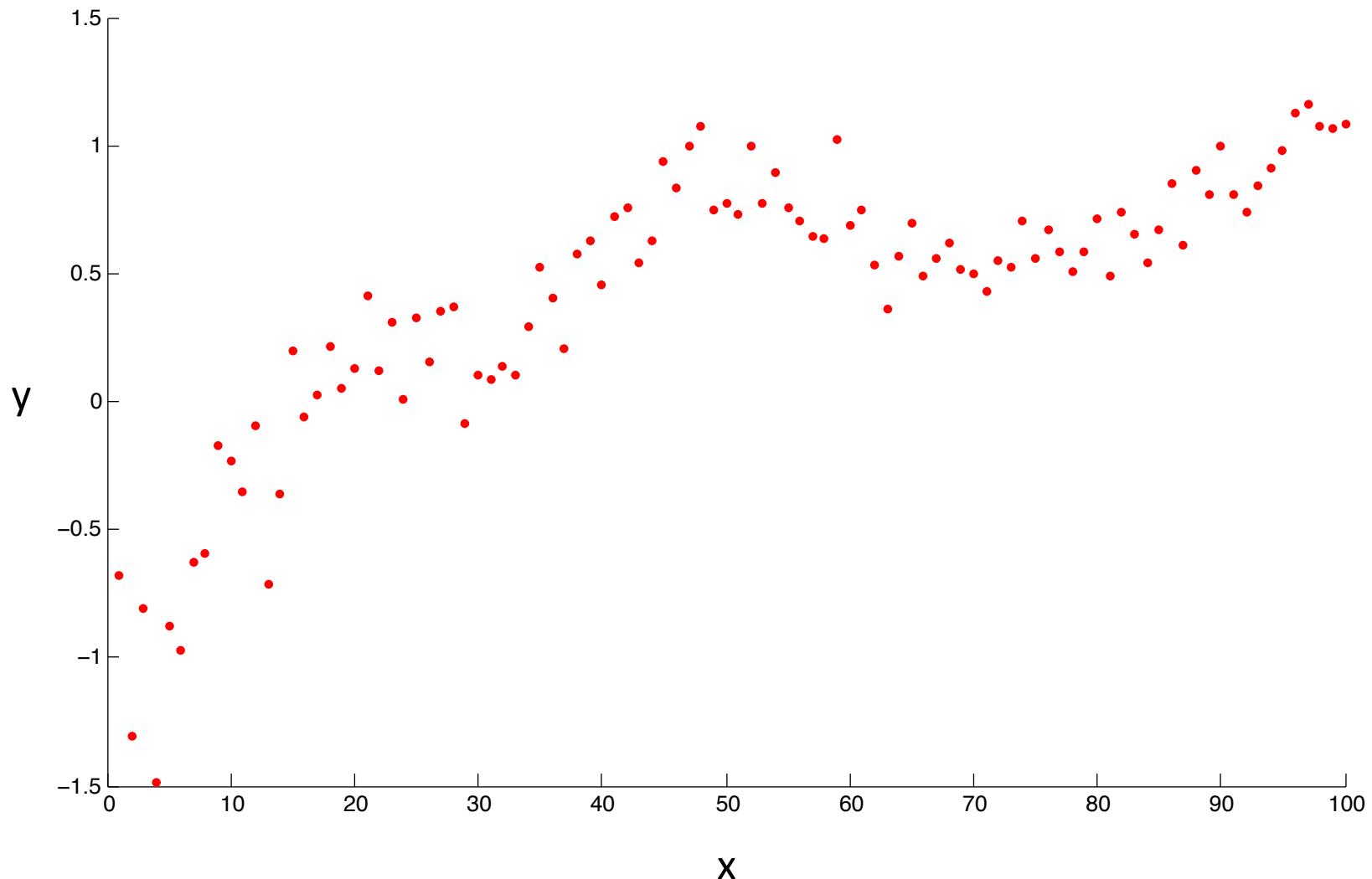
↑

“Average prediction”

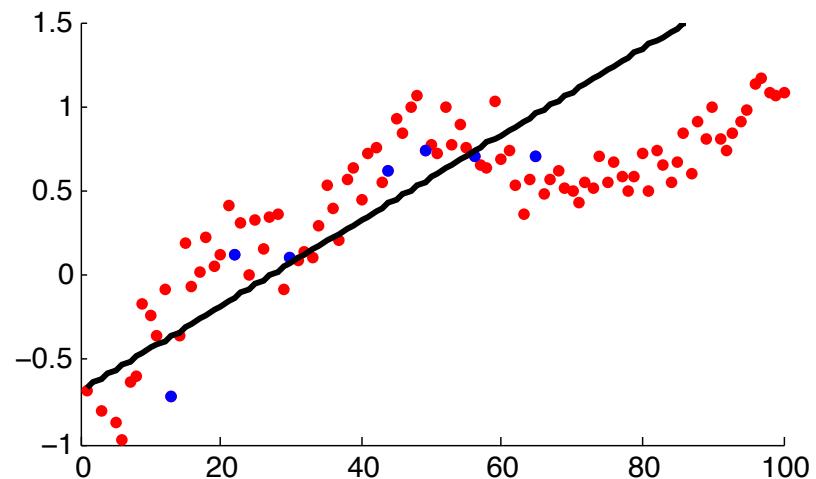
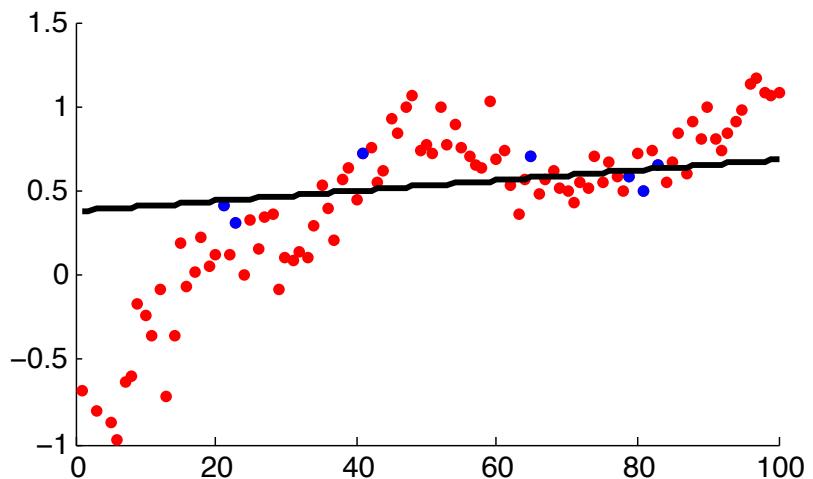
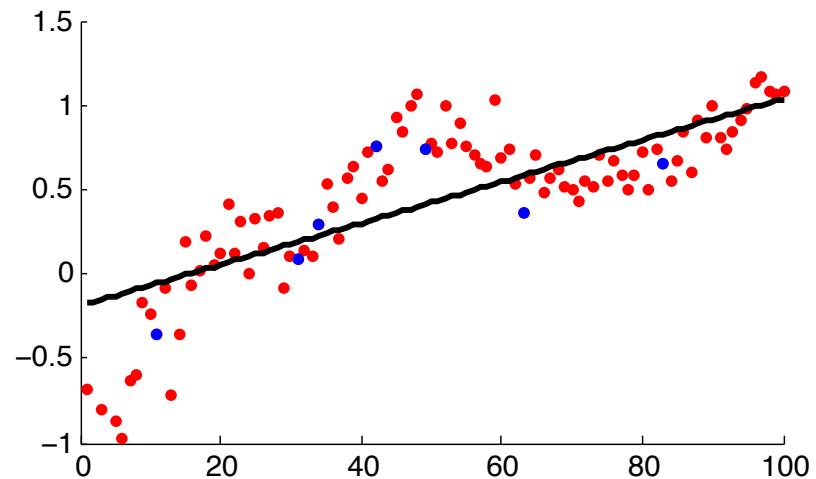
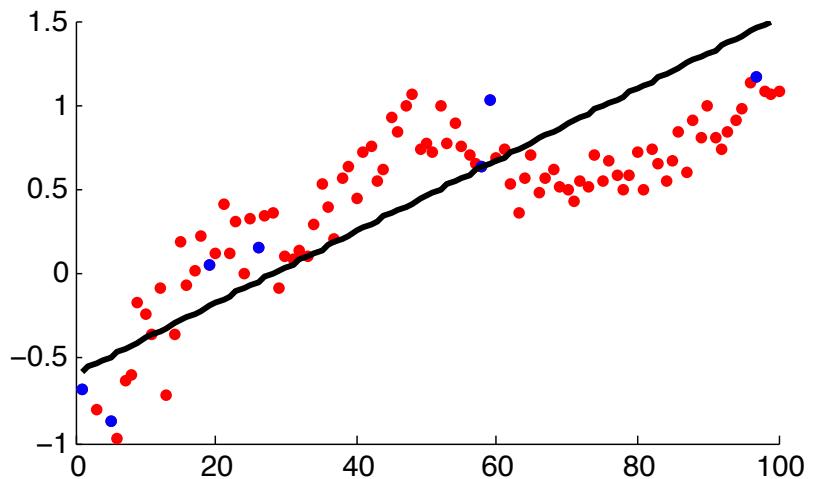
Variance Term

Bias Term

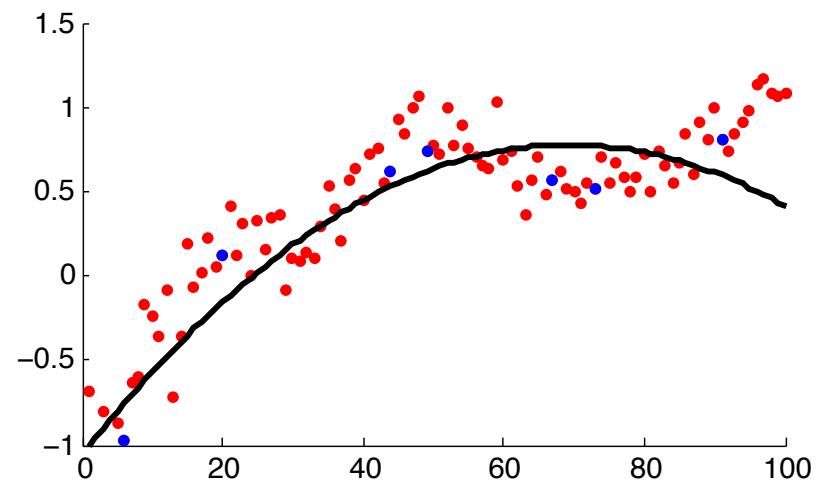
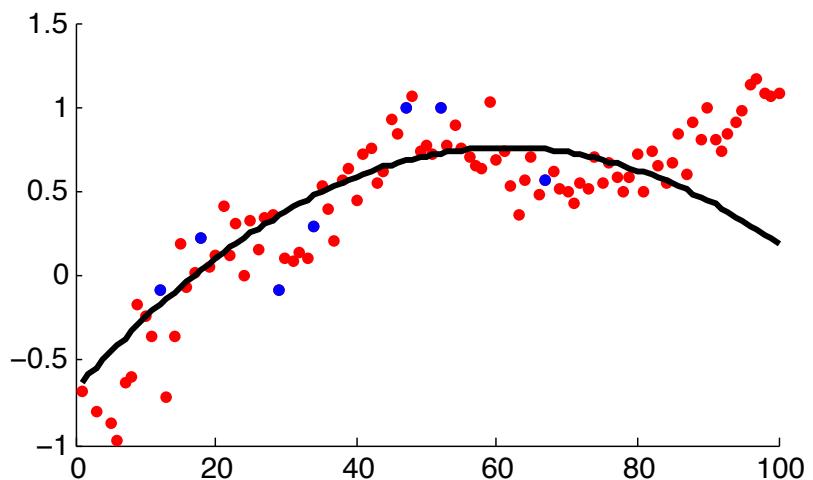
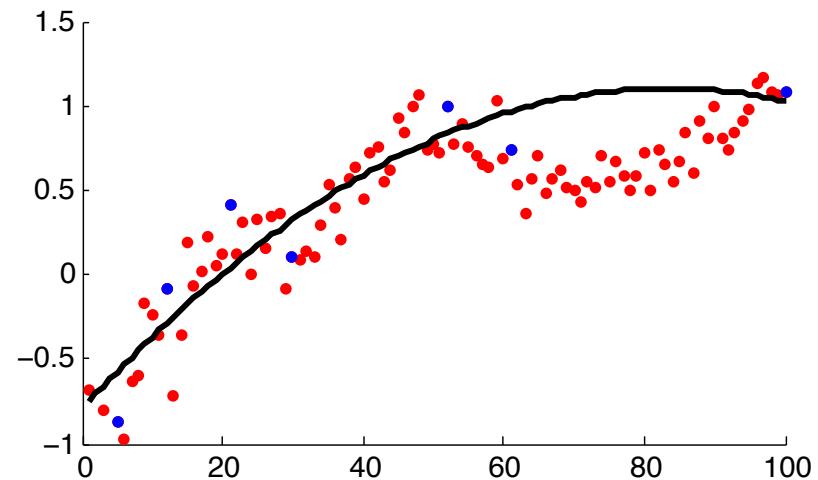
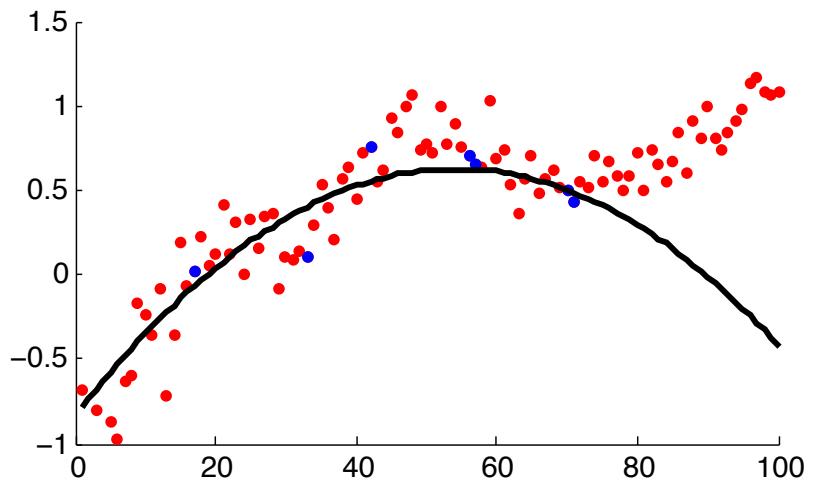
Example P(x,y)



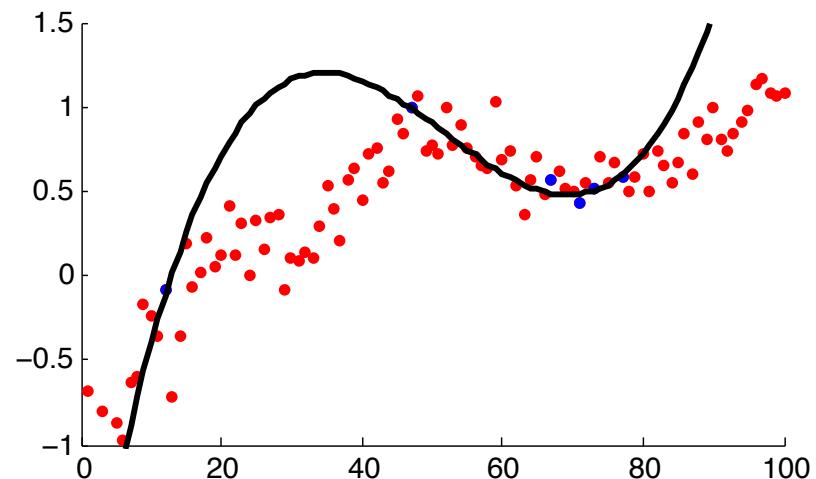
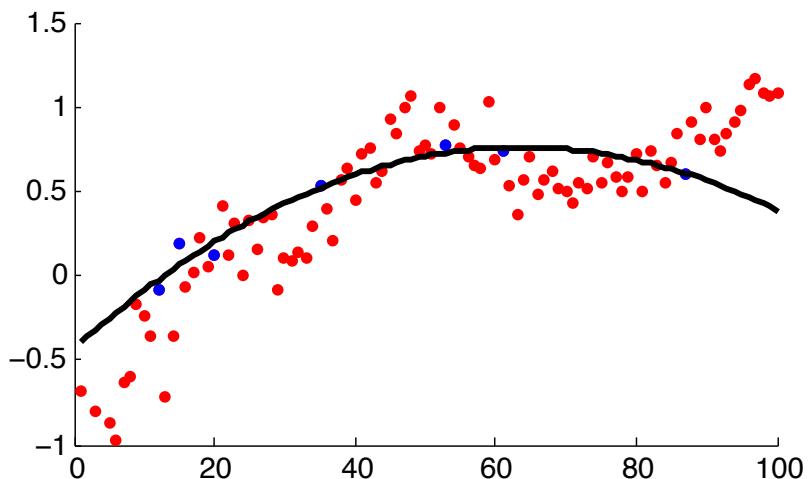
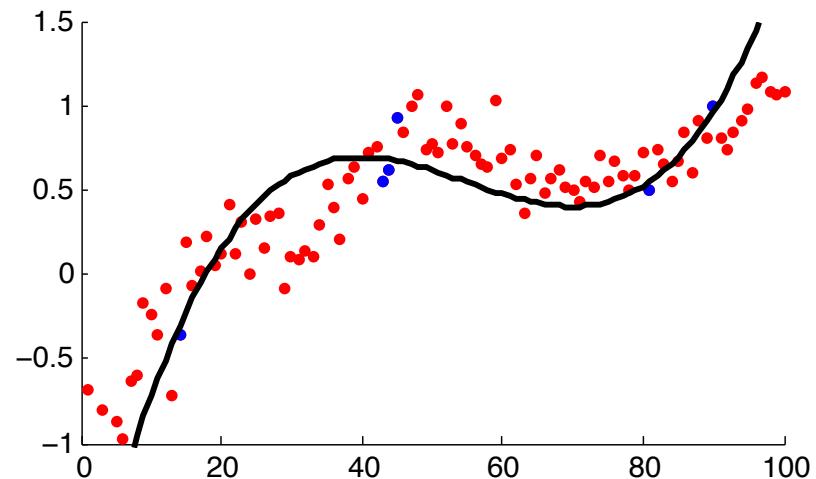
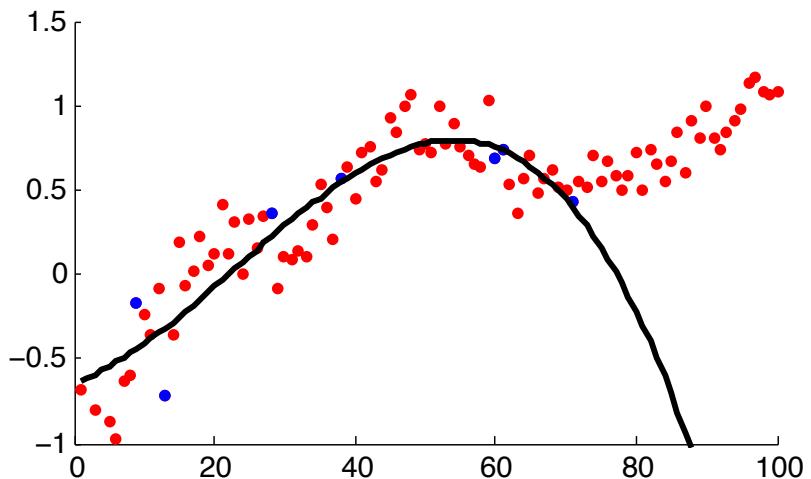
$f_S(x)$ Linear



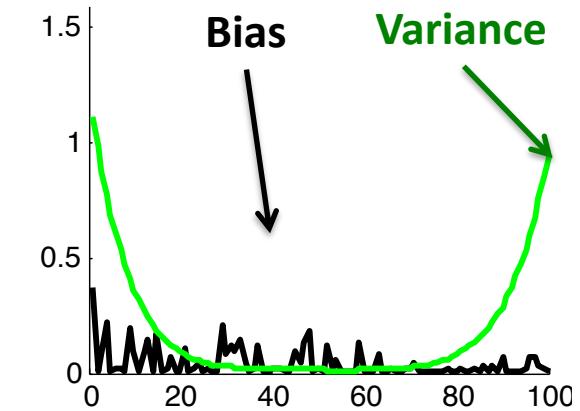
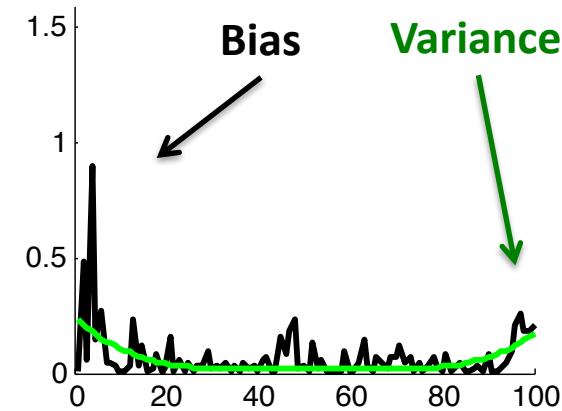
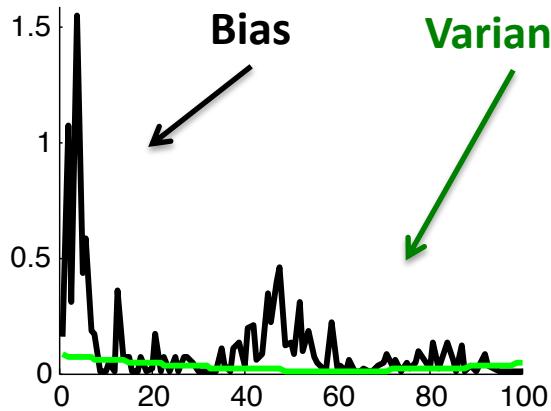
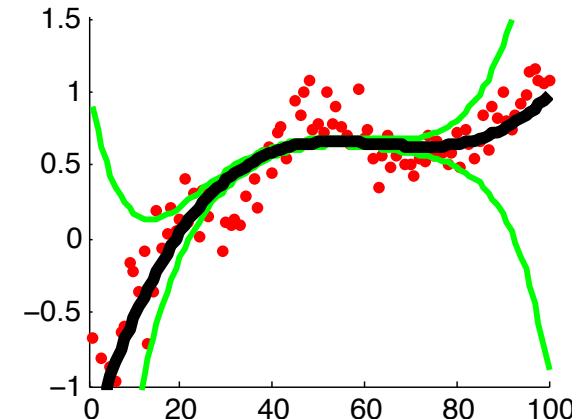
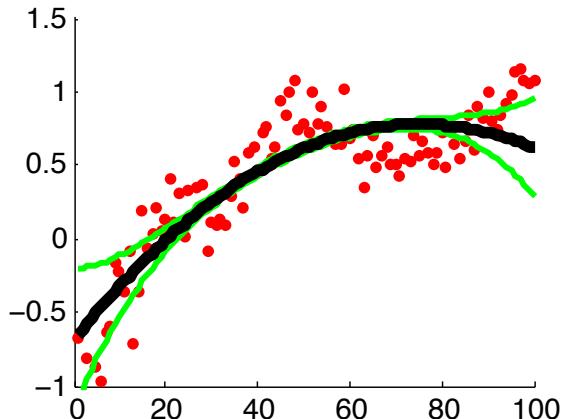
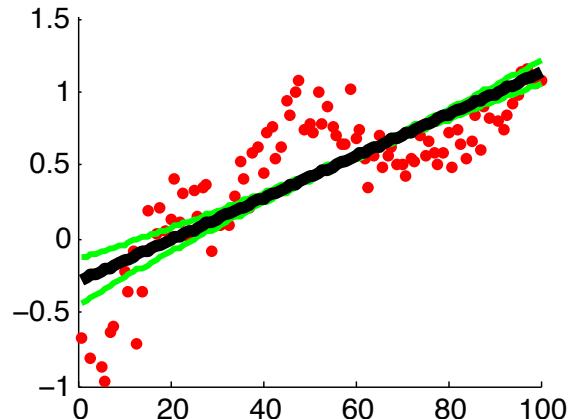
$f_S(x)$ Quadratic



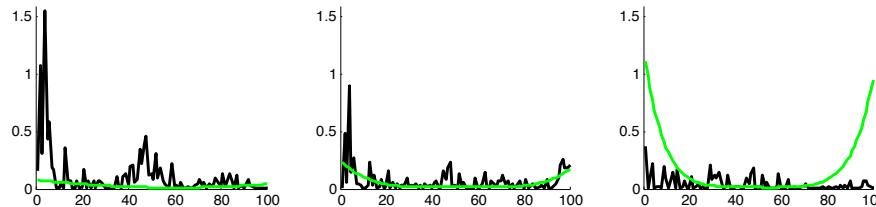
$f_s(x)$ Cubic



Bias-Variance Trade-off

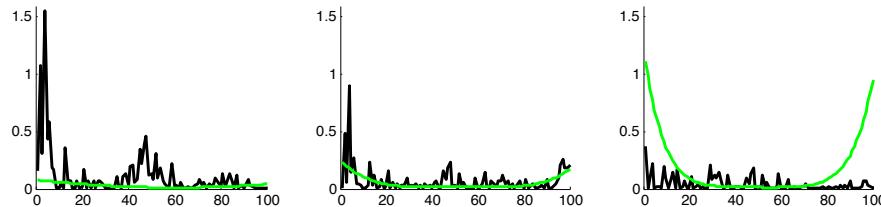


Overfitting vs Underfitting



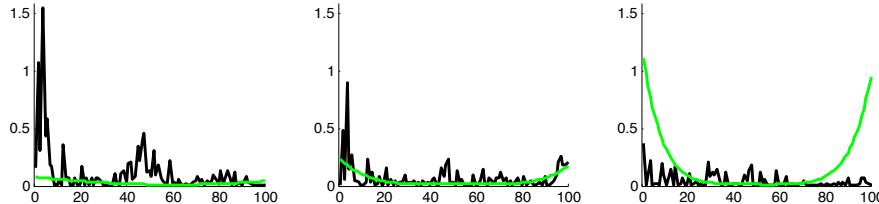
- High variance implies **overfitting**
 - Model class unstable
 - Variance increases with model complexity
 - Variance reduces with more training data.
- High bias implies **underfitting**
 - Even with no variance, model class has high error
 - Bias decreases with model complexity
 - Independent of training data size

Model Selection



- Finite training data
- Complex model classes overfit
- Simple model classes underfit
- **Goal:** choose model class with the best test error

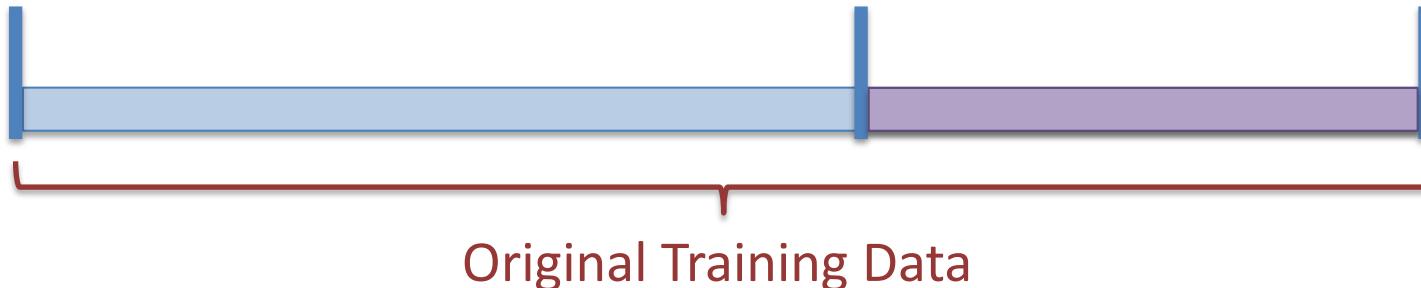
Model Selection



- Finite training data
- But we can't measure test error directly!
- (Don't have the whole distribution.)

error

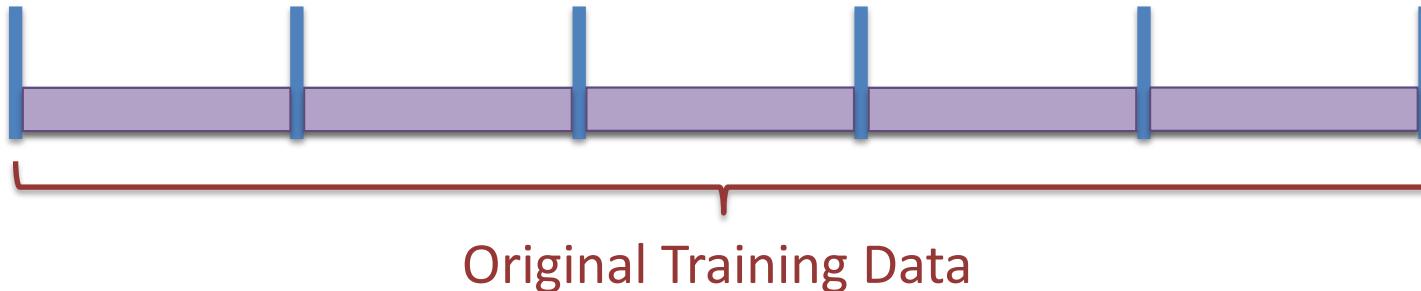
Use a Validation Set!



- Split data to **Training Set** and **Validation Set**
- Train model on **Training Set**
- Evaluate on **Validation Set** → Keep training and evaluation separate!
- What's wrong with this?
 - If dataset small, validation set small!

5-Fold Cross Validation

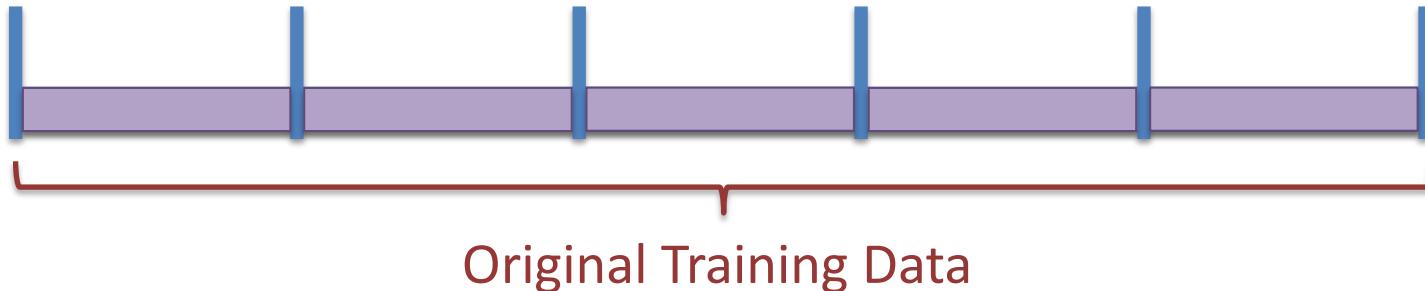
(one validation option)



- Split data into 5 equal partitions
- Train on 4 partitions
- Evaluate on 1 partition
- Allows re-using training data as test data
- Allows using all data as validation

5-Fold Cross Validation

(one validation option)



- Split data into 5 equal partitions
- Training & Validation set sampled independently from same distribution
- Allows using all data as validation

Complete Pipeline (Supervised Learning)

$$S = \{(x_i, y_i)\}_{i=1}^N$$

Training Data

$$f(x | w, b) = w^T x - b$$

Model Class(es)

$$L(a, b) = (a - b)^2$$

Loss Function



$$\operatorname{argmin}_{w,b} \sum_{i=1}^N L(y_i, f(x_i | w, b))$$

Cross Validation & Model Selection



Profit!

Next Lecture

- Perceptron
- Stochastic Gradient Descent
- Recitation on Thursday
 - Introduction to Python