Caltech

Machine Learning & Data Mining

CMS/CS/CNS/EE 155

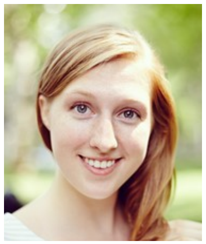
Lecture 1:

Administrivia & Basics

Course Info

- Lecture (Tu/Th)
  - 2:30pm – 3:55pm in Beckman Institute Auditorum
- Recitation (Th)
  - 7:00pm in 105 Annenberg
  - As needed
  - Usually 45-60 minutes
  - **First one on Thursday! (Introduction to Python)**

Staff


Natalie Bernat


Nishanth Bhaskara


Julia Deacon


Marcus Dominguez-Kuhne

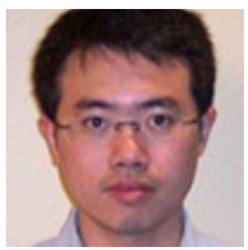

Rupesh Jeyaram


Cherie Jia

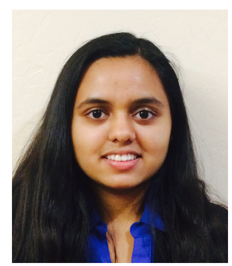

Karthik Karnik

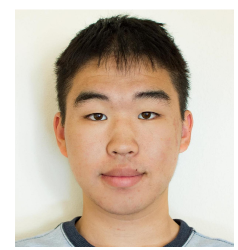

Meera Krishnamoorthy

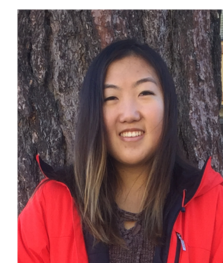

Karthik Nair


Jian Shi


Kapil Sinha


Vaishnavi Shrivastava


Albert Tseng


Michelle Zhao

Course Breakdown

- 6 Homeworks, ~60% of final grade
  - Due on Friday nights via Moodle & Gradescope
  - **Homework 1 will be released tonight.**
    - Due next Wednesday

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.

Moodle/Gradscope & Piazza

- Moodle & Gradescope:
  - https://courses.caltech.edu/course/view.php?id=3248
  - https://www.gradescope.com/courses/35620
  - Submission, Solutions, Grades
- Piazza
  - https://piazza.com/class/jqfs0b3935c7ho
  - Course announcements
  - Q&A Forum (use it!)
- Lecture Videos

  Not today sorry!
  - On YouTube (linked from course website)

Machine Learning & Data Mining

Computer Algorithm

Process of Converting

Data & Experience

Into Knowledge

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

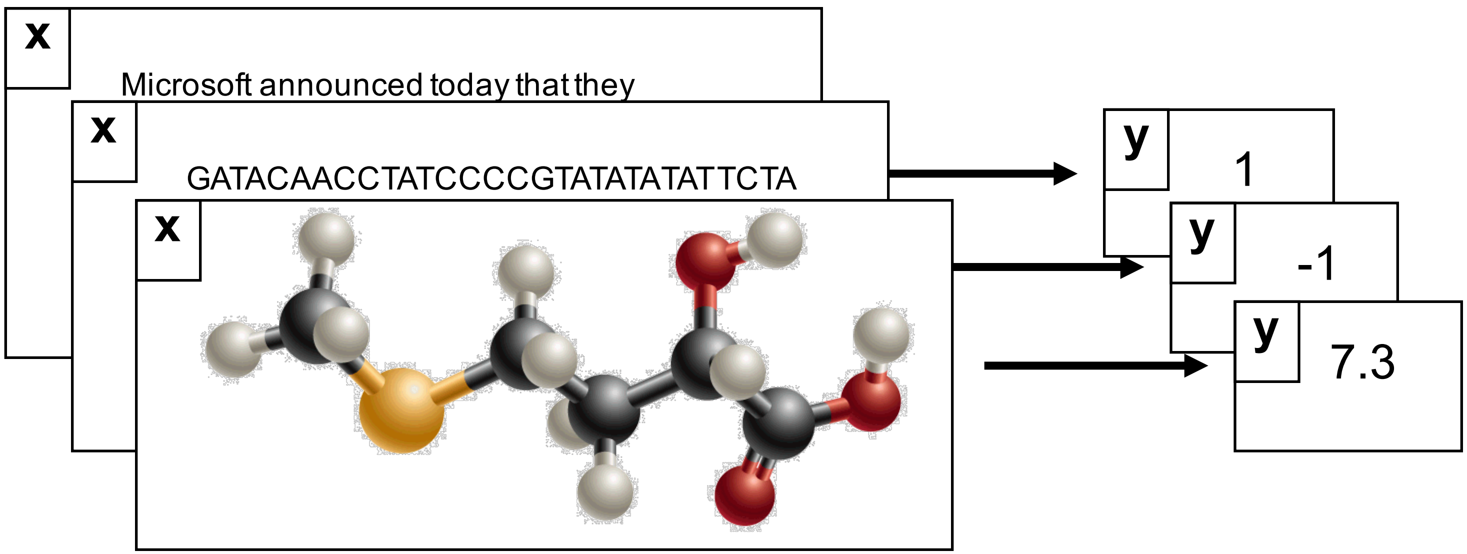
- Supervised Learning
  - 5 weeks
- Unsupervised Learning
  - 2 weeks
- Probabilistic Models
  - 2 weeks

Supervised Learning

- Find function from input space *X* to output space *Y*

$$f : X \to Y$$ (sometimes use *h*)

such that the prediction error is low.



Supervised Learning

Data: X

Target Signal: Y

()

$f(x) \approx y$

Logistic Regression
Artificial Neural Nets
Random Forests
Etc…

(function class or hypothesis class)

Aside: Unsupervised Learning

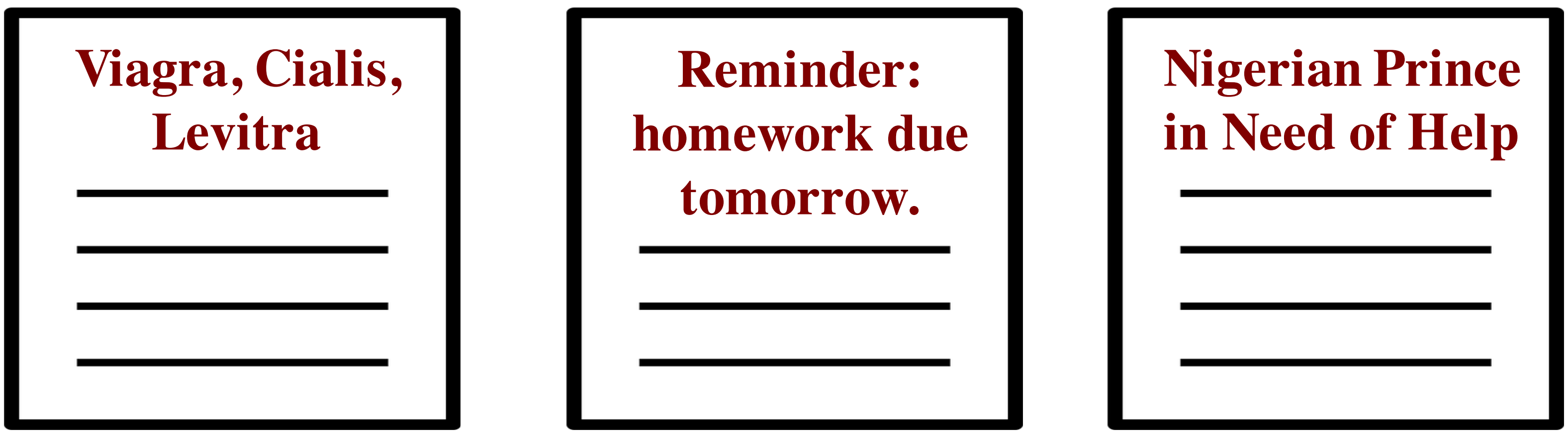
No supervised target!

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.



SPAM! | NOT SPAM | SPAM!

Example: Spam Filtering

- Goa

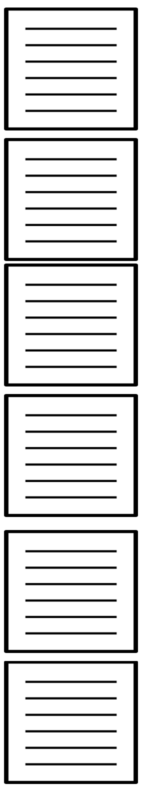
```
FUNCTION SpamFilter(string document)
{
        IF("Viagra" in document)
                RETURN TRUE
        ELSE IF("NIGERIAN PRINCE" in document)
                RETURN TRUE
        ELSE IF("Homework" in document)
                RETURN FALSE
        ELSE
                RETURN FALSE
        END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

}
```

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!

⋮

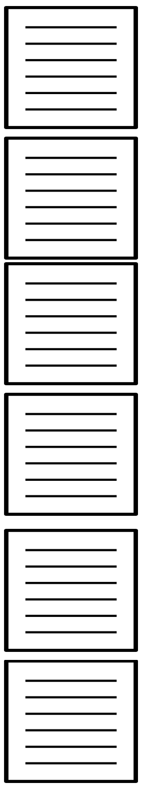
Build a Generic Representation

Run a Generic Learning Algorithm

➔ Classification Model

Labeled by Humans (“Supervision”)

Bag of Words Representation



Training Set	Bag of Words
SPAM!	(0,0,0,1,1,1)
SPAM!	(1,0,0,1,0,0)
NOT SPAM	(1,0,1,0,1,0)
NOT SPAM	(0,1,1,0,1,0)
SPAM!	(1,0,1,1,0,1)
SPAM!	(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:

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

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

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

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

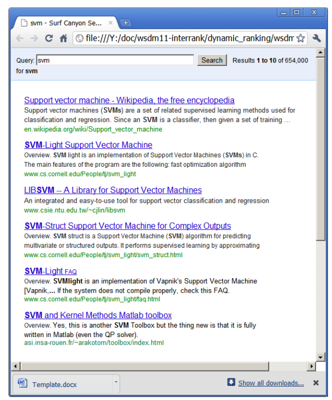
Training Set

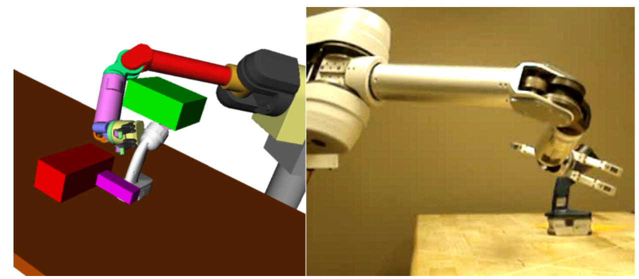
Bag of Words

SPAM!	(0,0,0,1,1,1)	$f(x\|w,b) = +1$
SPAM!	(1,0,0,1,0,0)	$f(x\|w,b) = +1$
NOT SPAM	(1,0,1,0,1,0)	$f(x\|w,b) = -1$
NOT SPAM	(0,1,1,0,1,0)	$f(x\|w,b) = -1$
SPAM!	(1,0,1,1,0,1)	$f(x\|w,b) = +1$
SPAM!	(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

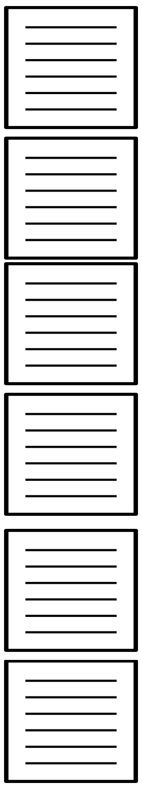
- **Classification** $$f(x \mid w, b) = \mathrm{sign}(w^T x - b)$$
  - Predict which class an example belongs to
  - E.g., spam filtering example
- **Regression** $$f(x \mid w, b) = w^T x - b$$
  - Predict a real value or a probability
  - E.g., probability of being spam
- **Highly inter-related**
 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

$$f(x|w,b) = w^T x - b$$
$$= w_1 * x_1 + \ldots w_6 * x_6 - b$$

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

Training Set

Bag of Words



SPAM!	(0,0,0,1,1,1)	$f(x\|w,b) = +0.5$
SPAM!	(1,0,0,1,0,0)	$f(x\|w,b) = +0.5$
NOT SPAM	(1,0,1,0,1,0)	$f(x\|w,b) = -0.5$
NOT SPAM	(0,1,1,0,1,0)	$f(x\|w,b) = -1.5$
SPAM!	(1,0,1,1,0,1)	$f(x\|w,b) = +1.5$
SPAM!	(1,0,0,0,0,1)	$f(x\|w,b) = +0.5$
⋮	⋮	⋮

Formal Definitions

- Training set: $S = \left\{(x_i, y_i)\right\}_{i=1}^{N}$ $x \in R^D$ $y \in \left\{-1, +1\right\}$

- Model class: $f(x \mid w, b) = w^T x - b$ **Linear Models**

aka hypothesis class

- **Goal:** find (w,b) that predicts well on S.
  – How to quantify "well"?

Basic Supervised Learning Recipe

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

Optimization Problem

Loss Function

- Measures penalty of mis-prediction:

- 0/1 Loss:

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

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

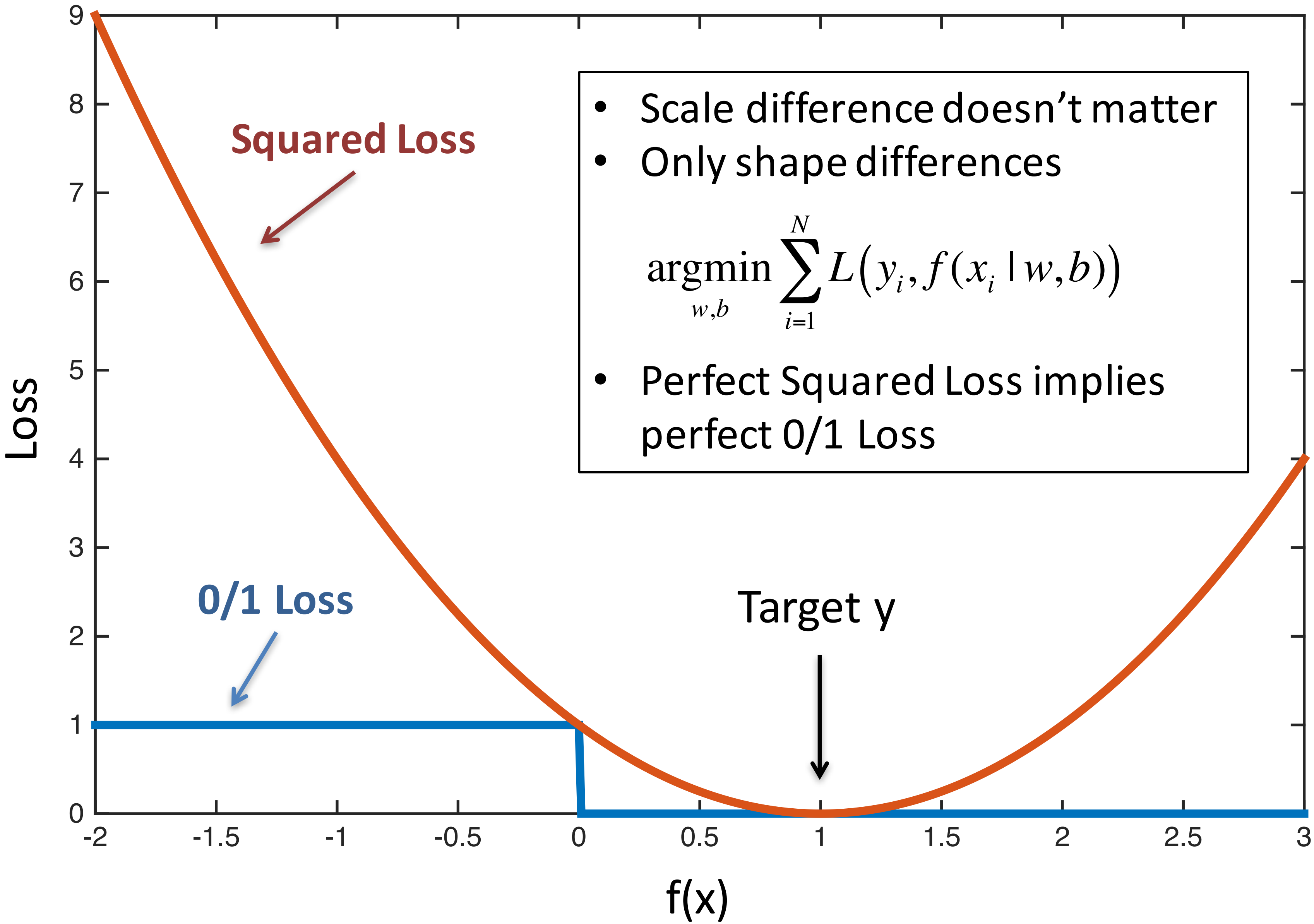
Classification

- Squared loss:

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

Regression

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


Squared Loss
0/1 Loss
Target y
Loss
f(x)
• Scale difference doesn't matter
• Only shape differences
$$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$$
• Perfect Squared Loss implies perfect 0/1 Loss
0
1
2
3
4
5
6
7
8
9
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

$$f(x|w,b) = w^T x - b$$
$$= w_1 * x_1 + \ldots w_6 * x_6 - b$$

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

Training Set	Bag of Words	
SPAM!	(0,0,0,1,1,1)	$f(x\|w,b) = +1$
SPAM!	(1,0,0,1,0,0)	$f(x\|w,b) = +1$
NOT SPAM	(1,0,1,0,1,0)	$f(x\|w,b) = -1$
NOT SPAM	(0,1,1,0,1,0)	$f(x\|w,b) = -1$
SPAM!	(1,0,1,1,0,1)	$f(x\|w,b) = +1$
SPAM!	(1,0,0,0,0,1)	$f(x\|w,b) = +1$

Train using Squared Loss

Learning Algorithm

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

- Typically, requires optimization algorithm.

- Simplest: **Gradient Descent**

Loop for T iterations

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

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

Gradient Review

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

More Details Next Lecture

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

Linearity of Differentiation

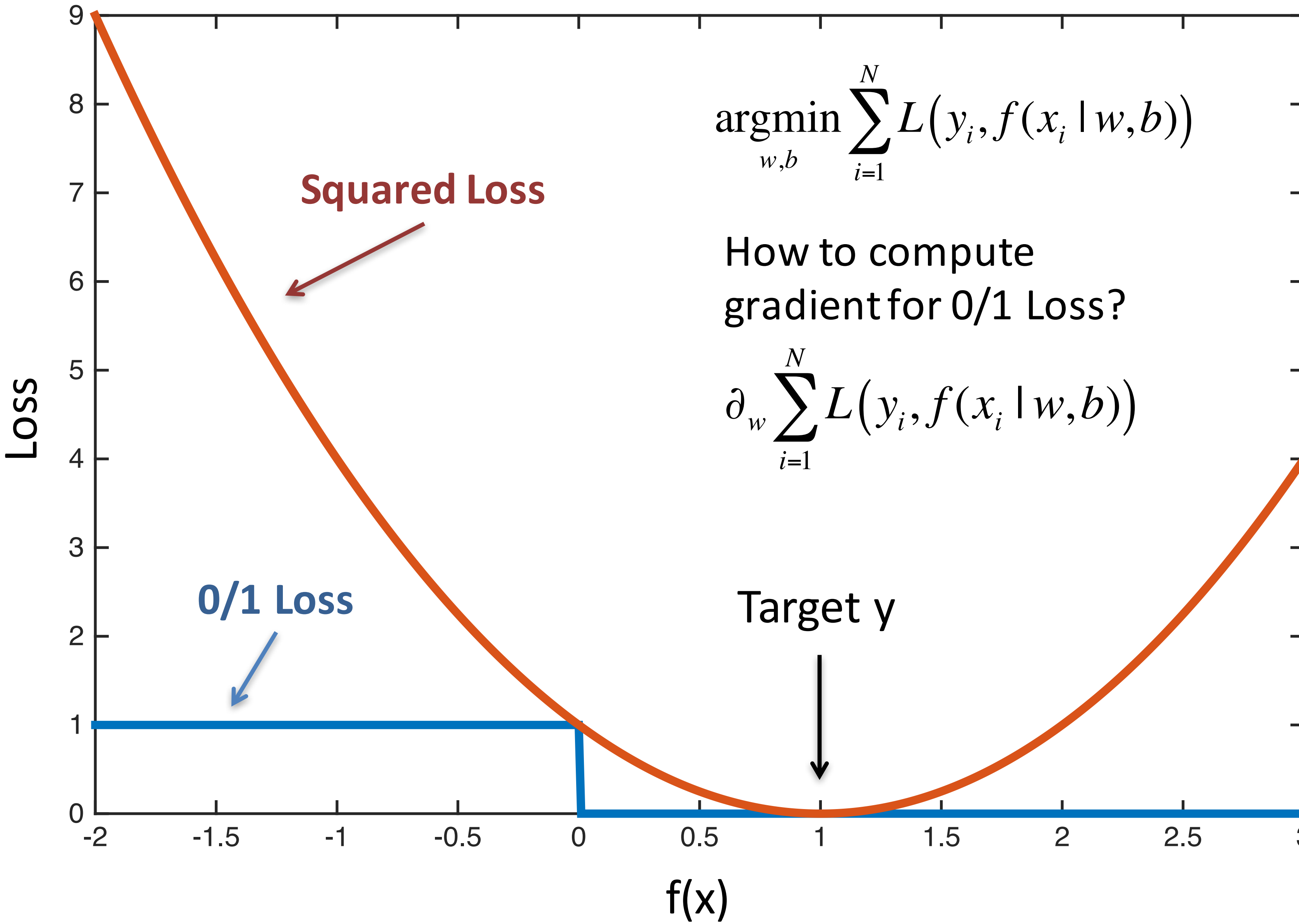
$$= \sum_{i=1}^{N} -2(y_i - f(x_i \mid w, b))\partial_w f(x_i \mid w, b)$$

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

Chain Rule

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

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



$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$
How to compute gradient for 0/1 Loss?
$\partial_w \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$
Squared Loss
0/1 Loss
Target y
Loss
f(x)
9
8
7
6
5
4
3
2
1
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

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** $f(x \mid w, b) = \text{sign}(w^T x - b)$
  - Predict which class an example belongs to
  - E.g., spam filtering example
- **Regression** $f(x \mid w, b) = w^T x - b$
  - 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: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \in R^D$ $\quad y \in \{-1, +1\}$
- Model Class: $f(x \mid w, b) = w^T x - b$ **Linear Models**
- Loss Function: $L(a, b) = (a - b)^2$ **Squared Loss**
- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$

Optimization Problem

Recap: Supervised Learning Recipe

- Trainin
- Mode
- Loss F
- Learning Objective:

$x \in R^D$

$y \in \{-1,+1\}$

Linear Models

Squared Loss

Congratulations!

You now know the basic steps to training a model!

But is your model any good?

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

Optimization Problem

Example: Self-Driving Cars

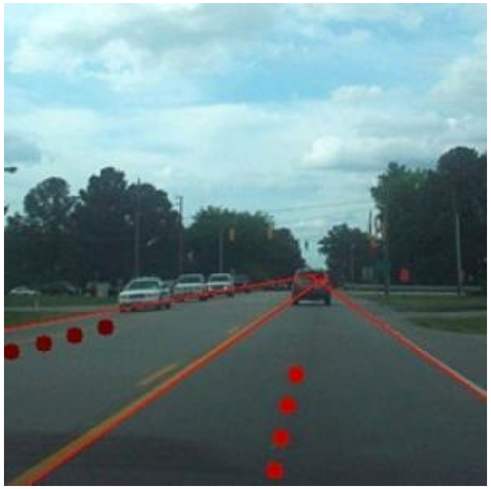
Basic Setup

- Mounted cameras
- Use image features
- Human demonstrations

- $f(x|w)$ = 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)
  - Unknown to us

"All possible emails"

- **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:**

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

Prediction Loss on all possible emails

- **Overfitting:** Test Error >> Training Error

Test Error

- **Test Error:**

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

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

$$f_S = \underset{w,b}{\operatorname{argmin}} \sum_{(x_i,y_i)\in S} L\left(y_i, f(x_i \mid w,b)\right)$$

- **Expected Test Error:**

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

Bias-Variance Decomposition

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

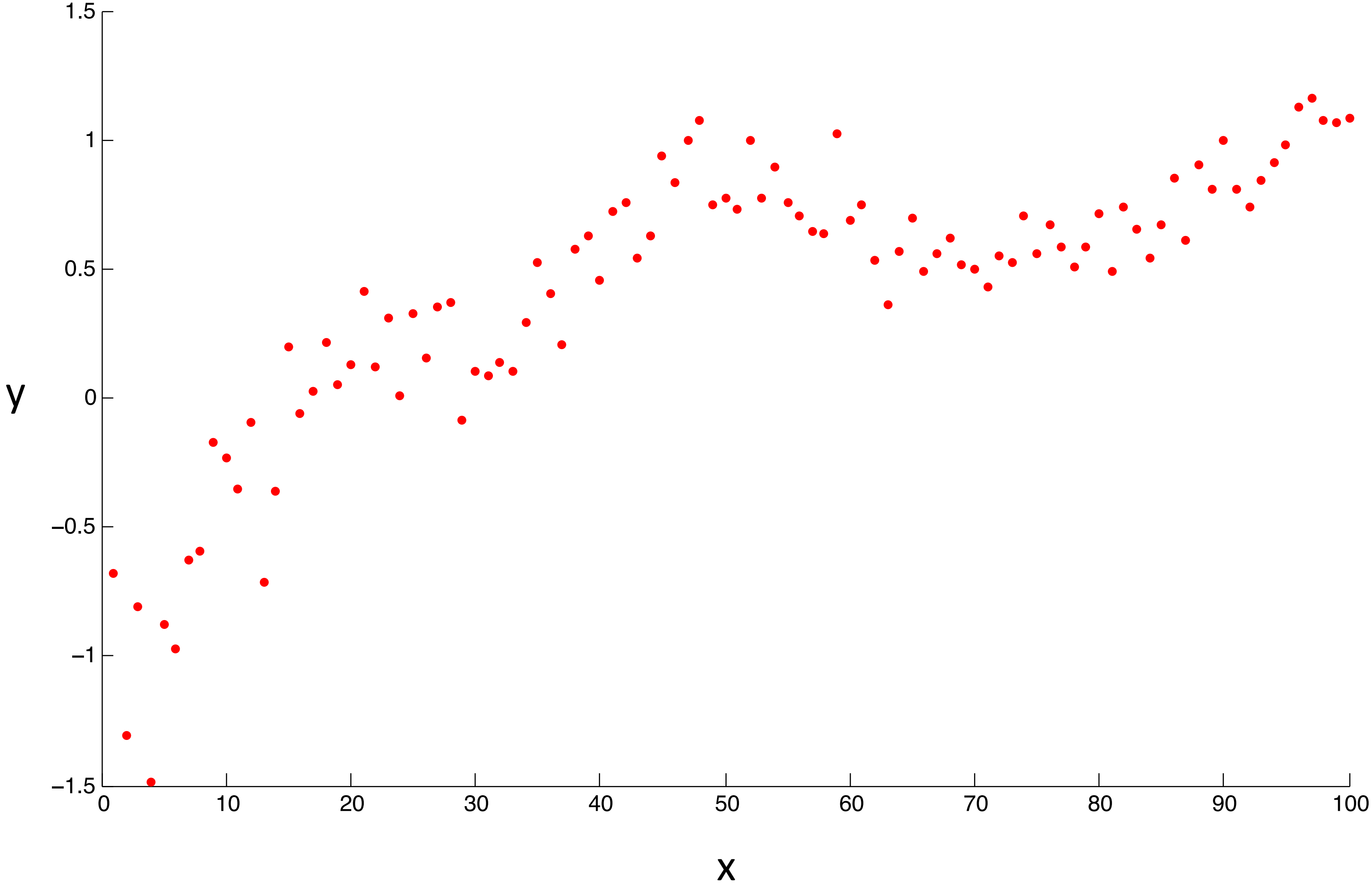
- For squared error:

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

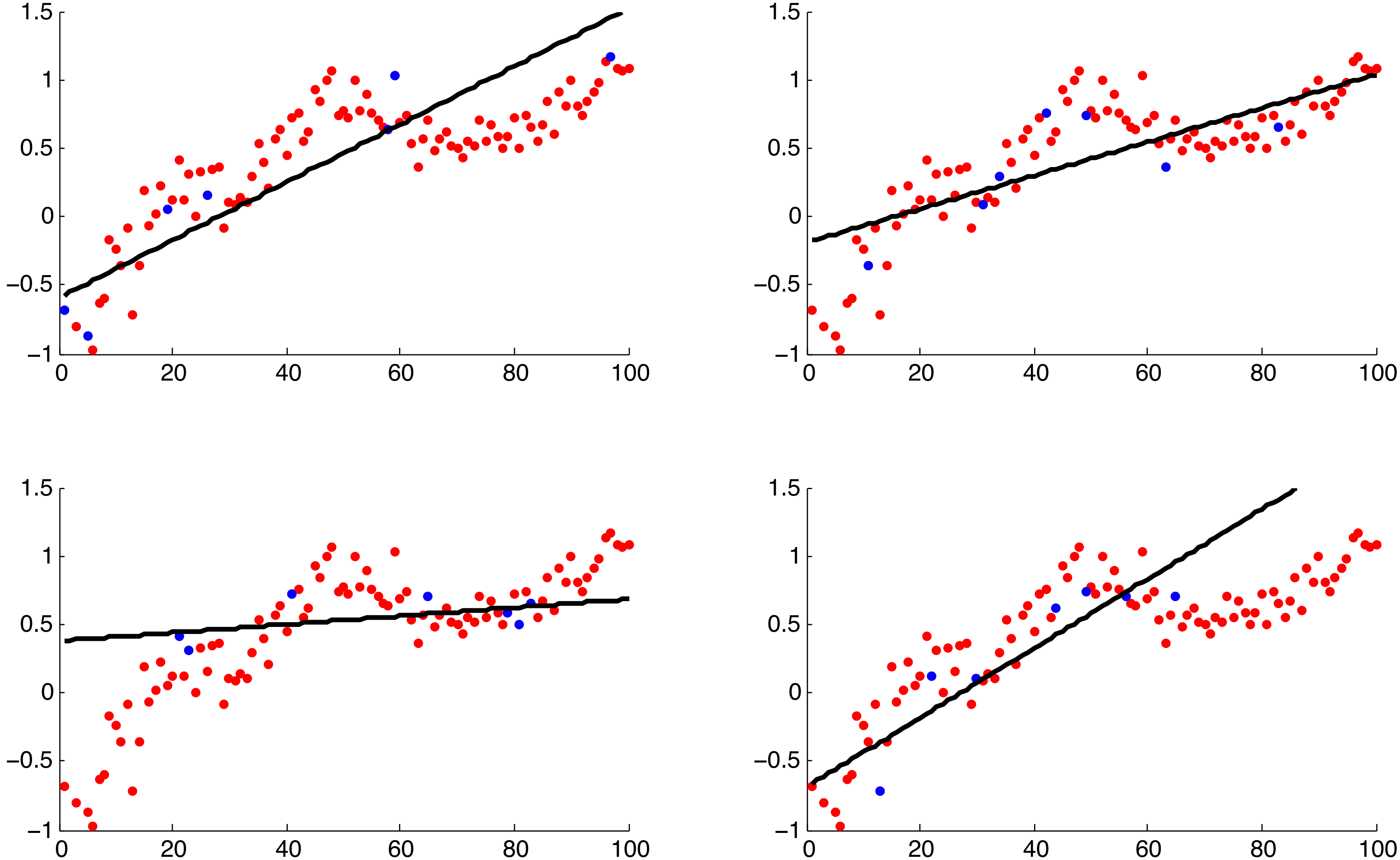
$$F(x) = E_S\left[f_S(x)\right]$$

"Average prediction"

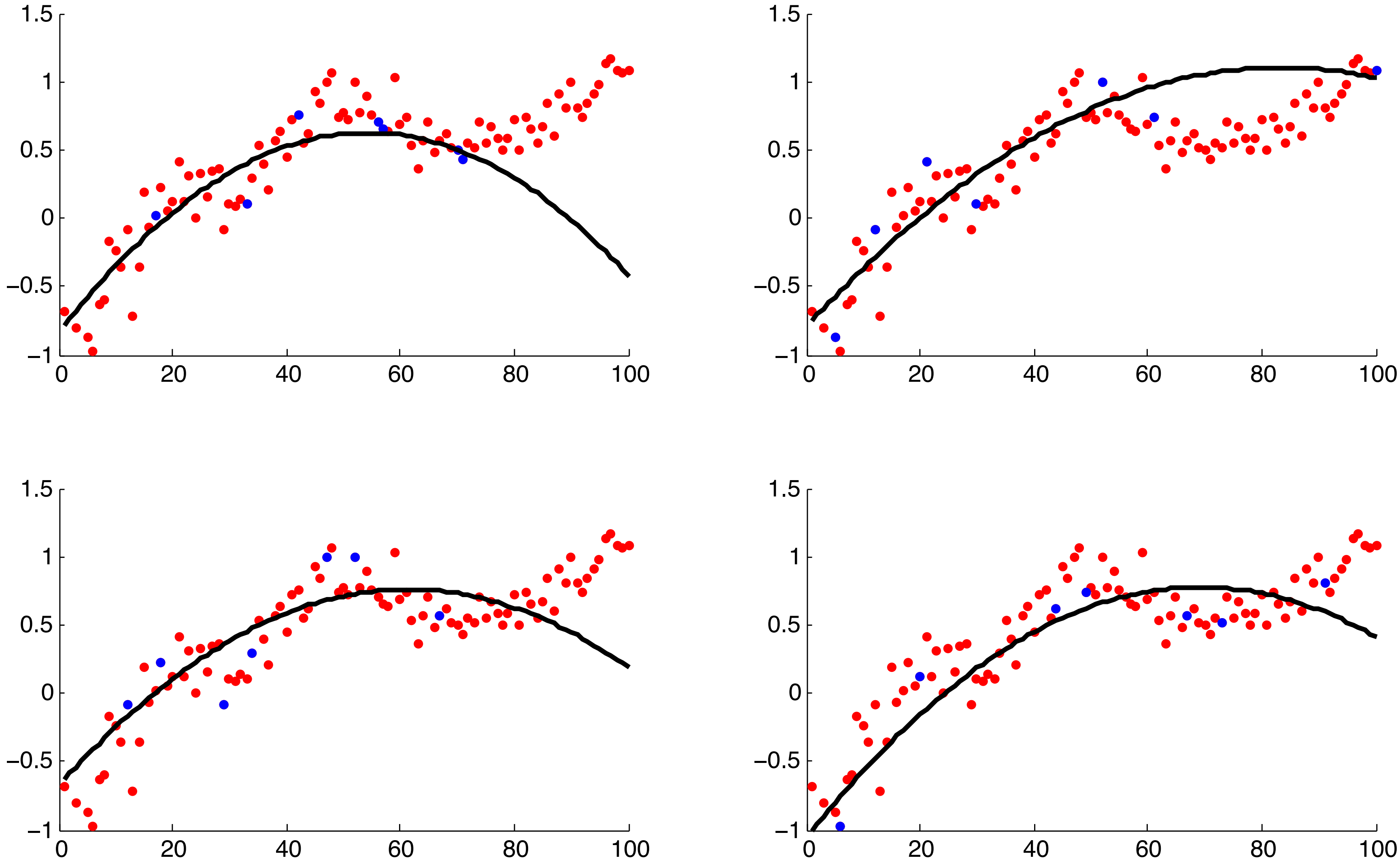
Example P(x,y)



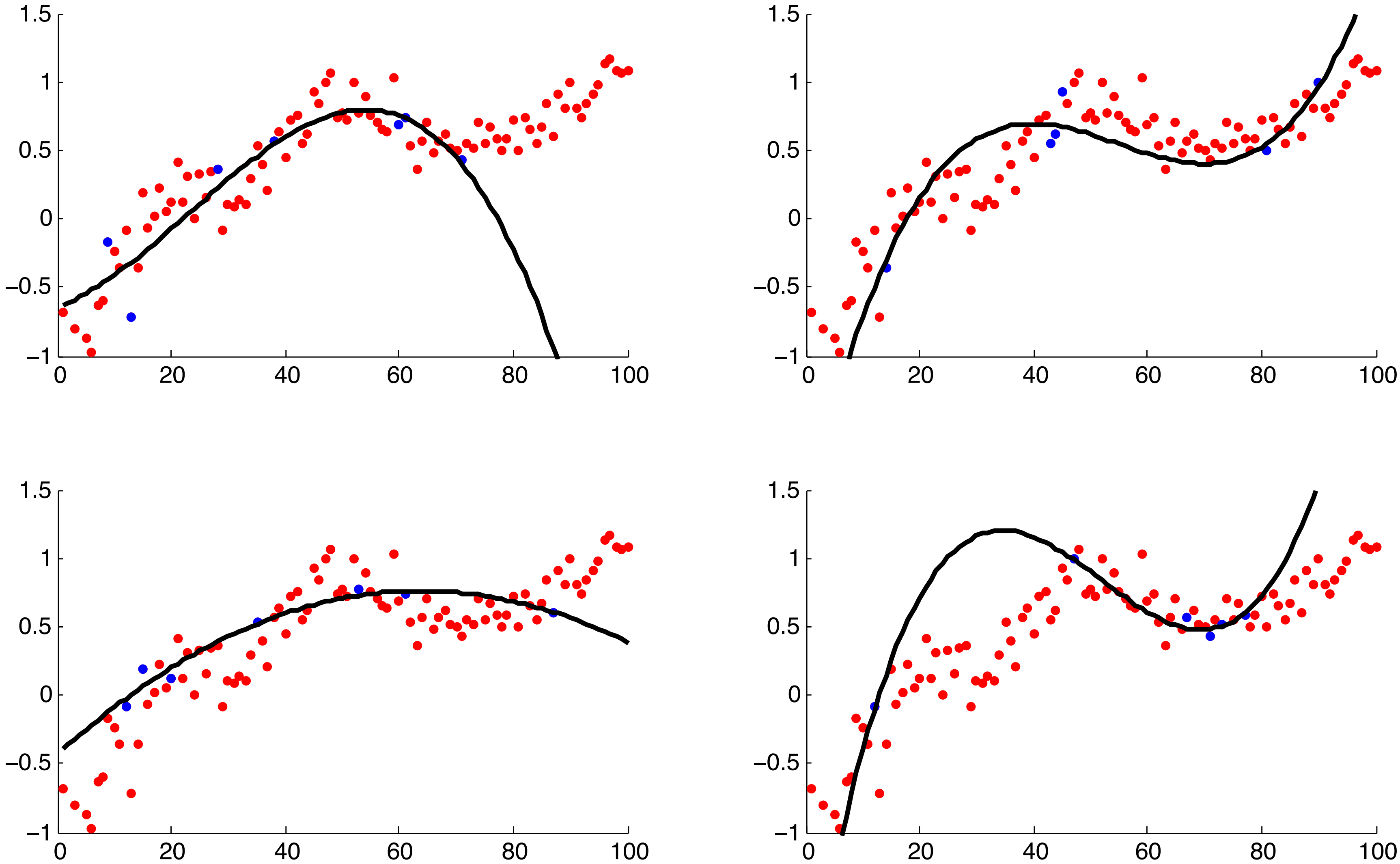
$f_S(x)$ Linear



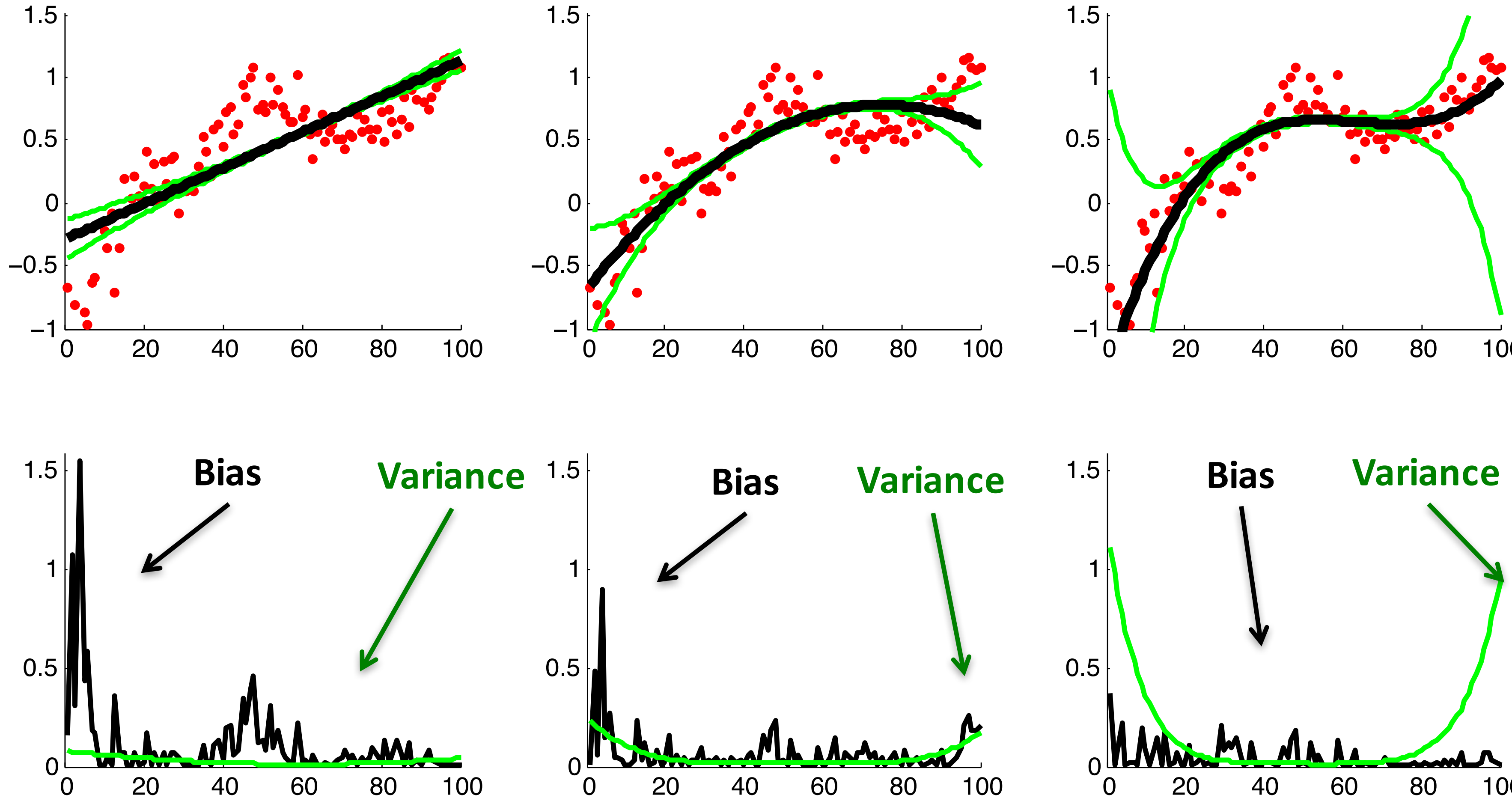
$f_S(x)$ Quadratic



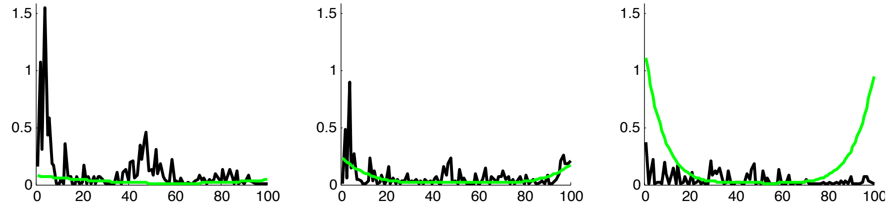
$f_S(x)$ Cubic



Bias-Variance Trade-off

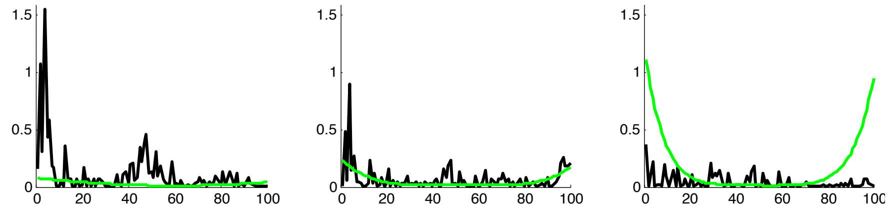


Overfitting vs Underfitting



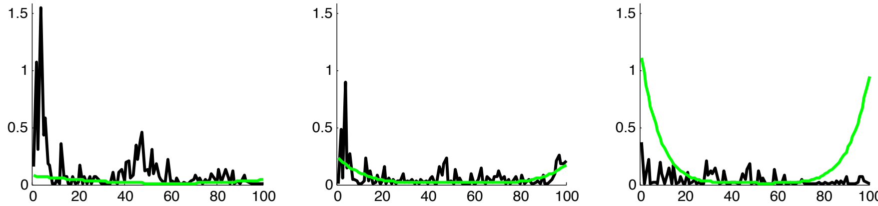
- High variance implies **overfitting**
  - Model class unstable
  - Variance increases with model complexity
  - Variance reduces with more training data.
- High bias implies **underfitting**
  - Even with no variance, model class has high error
  - Bias decreases with model complexity
  - Independent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



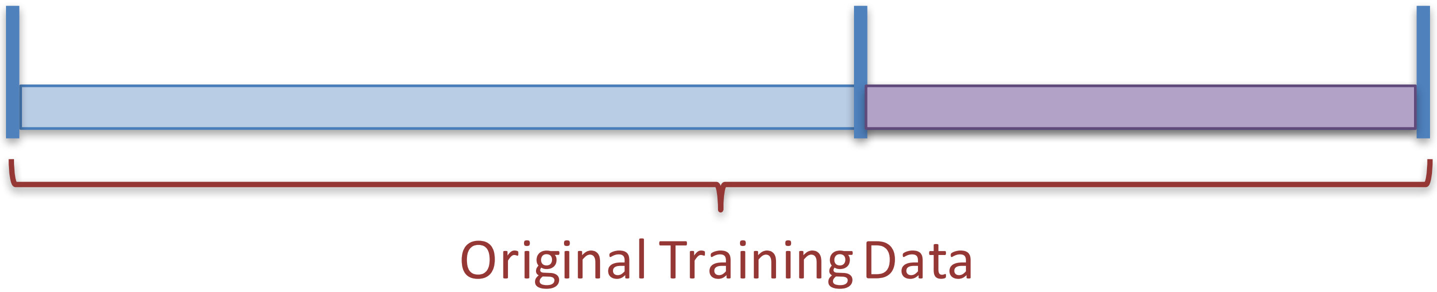
- Finite training data

But we can’t measure test error directly!

(Don’t have the whole distribution.)

error

Use a Validation Set!



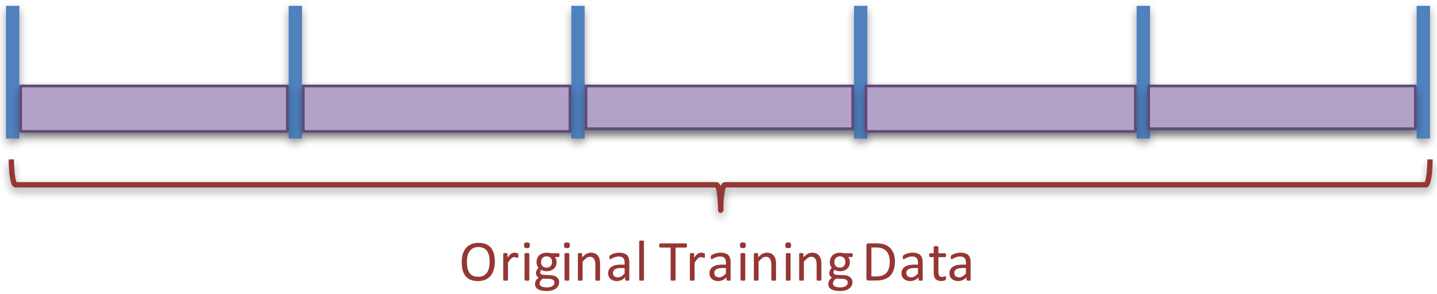
- Split data to Training Set and Validation Set
- Train model on Training Set
- Evaluate on Validation Set

Keep training and evaluation separate!

- What's wrong with this?
  - **If dataset small, validation set small!**

5-Fold Cross Validation

(one validation option)

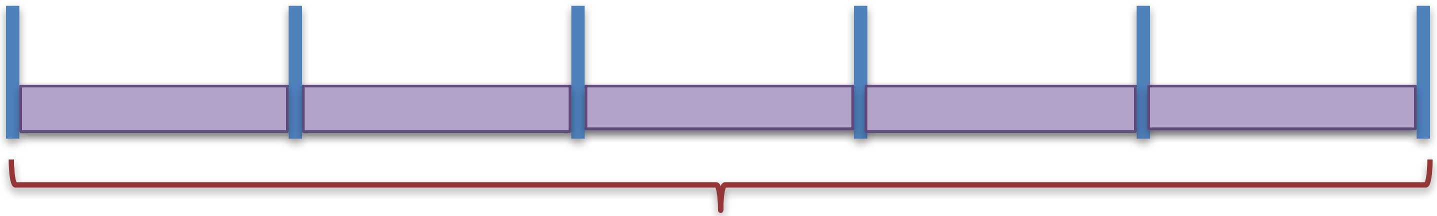


- Split data into 5 equal partitions

- 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)



Original Training Data

- 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

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

Model Class(es)

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

Loss Function

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

Cross Validation & Model Selection

Profit!

Next Lecture

- Perceptron
- Stochastic Gradient Descent

- Recitation on Thursday
  - Introduction to Python
  - 7pm in Annenberg 105