Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 1:

Administrivia & Review

Course Info

- Lecture (Tu/Th)
  - 2:30pm – 3:55pm in 105 Annenberg
- Recitation (W)
  - 5:00pm – 7:00pm in 105 Annenberg
  - As needed
  - Usually less than the full 2 hours

Staff

- Instructor: **Yisong Yue**
  - Office in 303 Annenberg

- TAs:
  - Bryan He
  - Masoud Farivar
  - Shenghan Yao
  - Vighnesh Shiv
  - Minfa Wang
  - Vinny Augustine

Course Prerequisites

- Algorithms and statistics
  - CS/CNS/EE/NB 154 or CS/CNS/EE 156a
- Calculus and algebra
  - Comfort in mathematical reasoning
- **Do Homework 0!**
  - If you have a lot of trouble, consider dropping
    - **(especially underclassmen)**
    - Take CS 156a next year first
  - **If you plan to drop, do it early.**

Course Breakdown

- 5-6 Homeworks, 30-40% of final grade
  - Due Tuesdays
  - **Homework 1 release next Monday**
- 2-3 Mini-projects, 20-30% of final grade
- Final, 30-40% of final grade

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.
- Adhere to the Academic Integrity
  - 50ft 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, homeworks, etc.

Moodle

- https://courses.caltech.edu/course/view.php?id=1787
- Linked to from course website
- Forums & Assignment Submission
- Requires Enrollment Key

Caveats

- This is my first time teaching a course…. ever.

- Please be understanding.

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

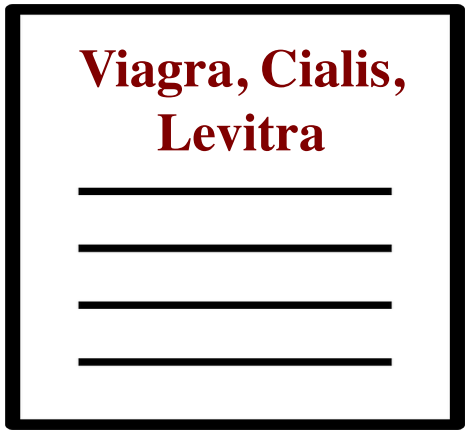
- **Huge overlap**

Course Outline

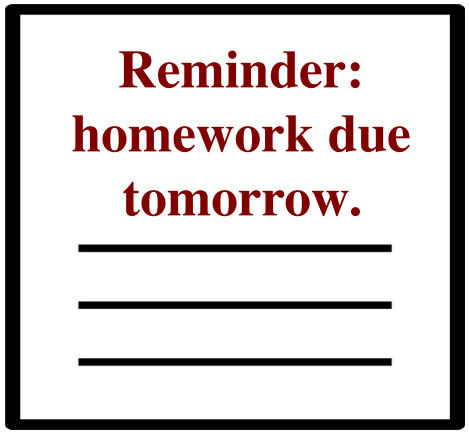
- Review over basics (this week)
- Modern techniques:
  - Lasso
  - HMMs & Graphical Models
  - Ensemble Methods (boosting)
  - Latent factor models, topic models, deep learning
  - Semi-supervised & active learning.

Example: Spam Filtering

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



SPAM!

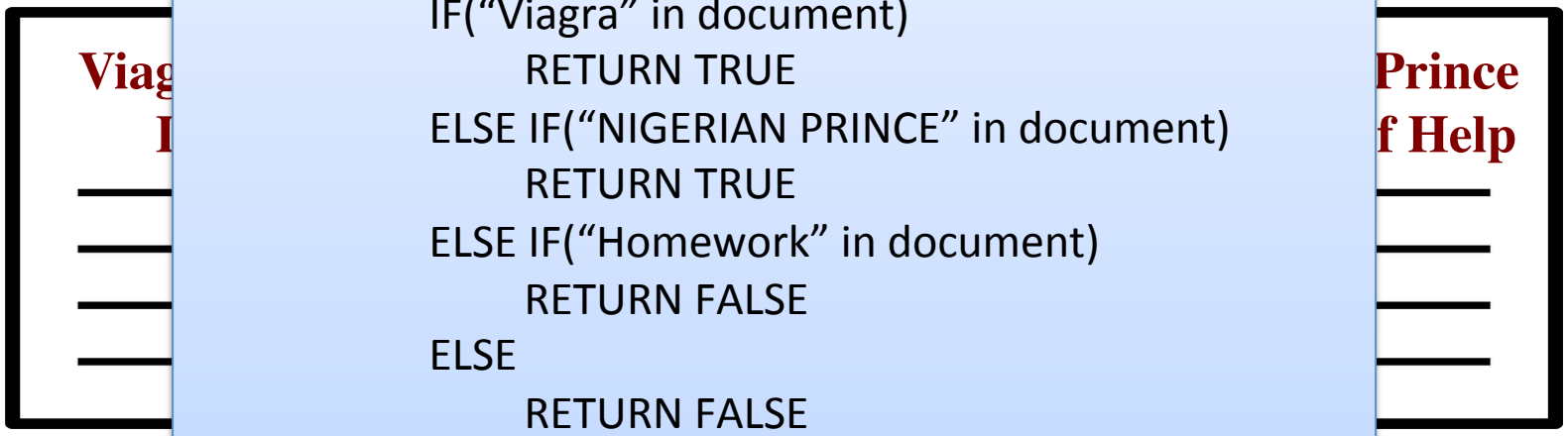


NOT SPAM

SPAM!

Example: Spam Filtering

- **Goal**



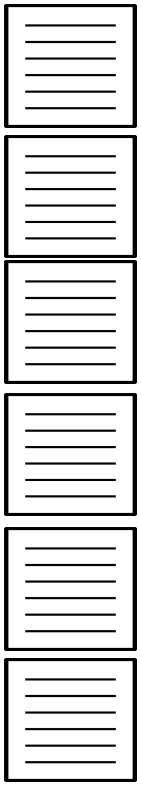

```
FUNCTION SpamFilter(string document)
{
        IF("Viagra" in document)
                RETURN TRUE
        ELSE IF("NIGERIAN PRINCE" in document)
                RETURN TRUE
        ELSE IF("Homework" in document)
                RETURN FALSE
        ELSE
                RETURN FALSE
        END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

}
```

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!

Labeled by Humans

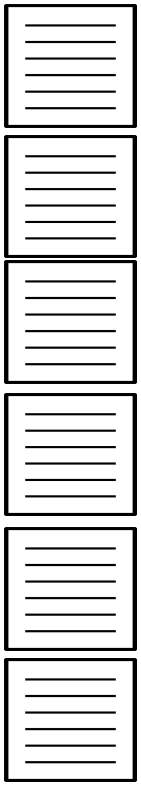
Build a Generic Representation

Run a Generic Learning Algorithm

➔ Classification Model

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)

Linear Classifier:

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

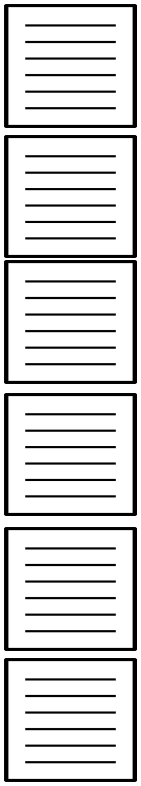
$= \text{sign}(w_1 * x_1 + \ldots w_6 * x_6 - b)$

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

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

Training Set

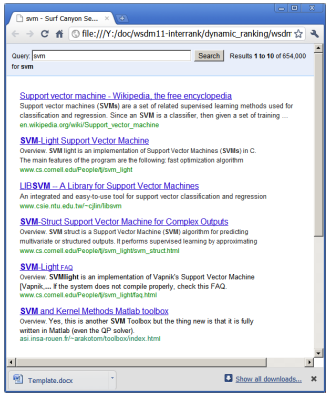
Bag of Words

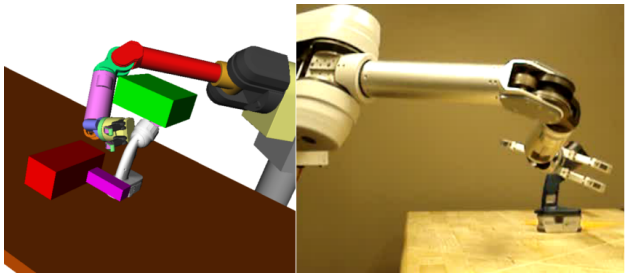


SPAM!	(0,0,0,1,1,1)	f(x|w,b) = +1
SPAM!	(1,0,0,1,0,0)	f(x|w,b) = +1
NOT SPAM	(1,0,1,0,1,0)	f(x|w,b) = -1
NOT SPAM	(0,1,1,0,1,0)	f(x|w,b) = -1
SPAM!	(1,0,1,1,0,1)	f(x|w,b) = +1
SPAM!	(1,0,0,0,0,1)	f(x|w,b) = +1
⋮	⋮	⋮

Linear Models

- Workhorse of Machine Learning





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

Two Basic ML Problems

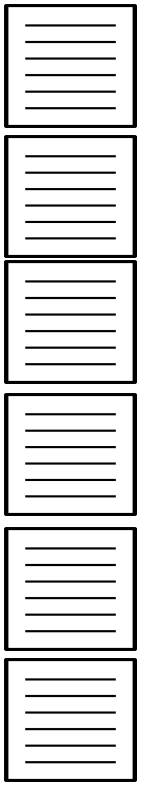
- **Classification** $f(x \mid w, b) = \text{sign}(w^T x - b)$
  - Predict which class an example belongs to
  - E.g., spam filtering example
- **Regression** $f(x \mid w, b) = w^T x - b$
  - Predict a real value or a probability
  - E.g., probability of being spam
- **Highly inter-related**
  - Train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

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

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

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $x \in R^D$ $y \in \{-1, +1\}$
- 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: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$

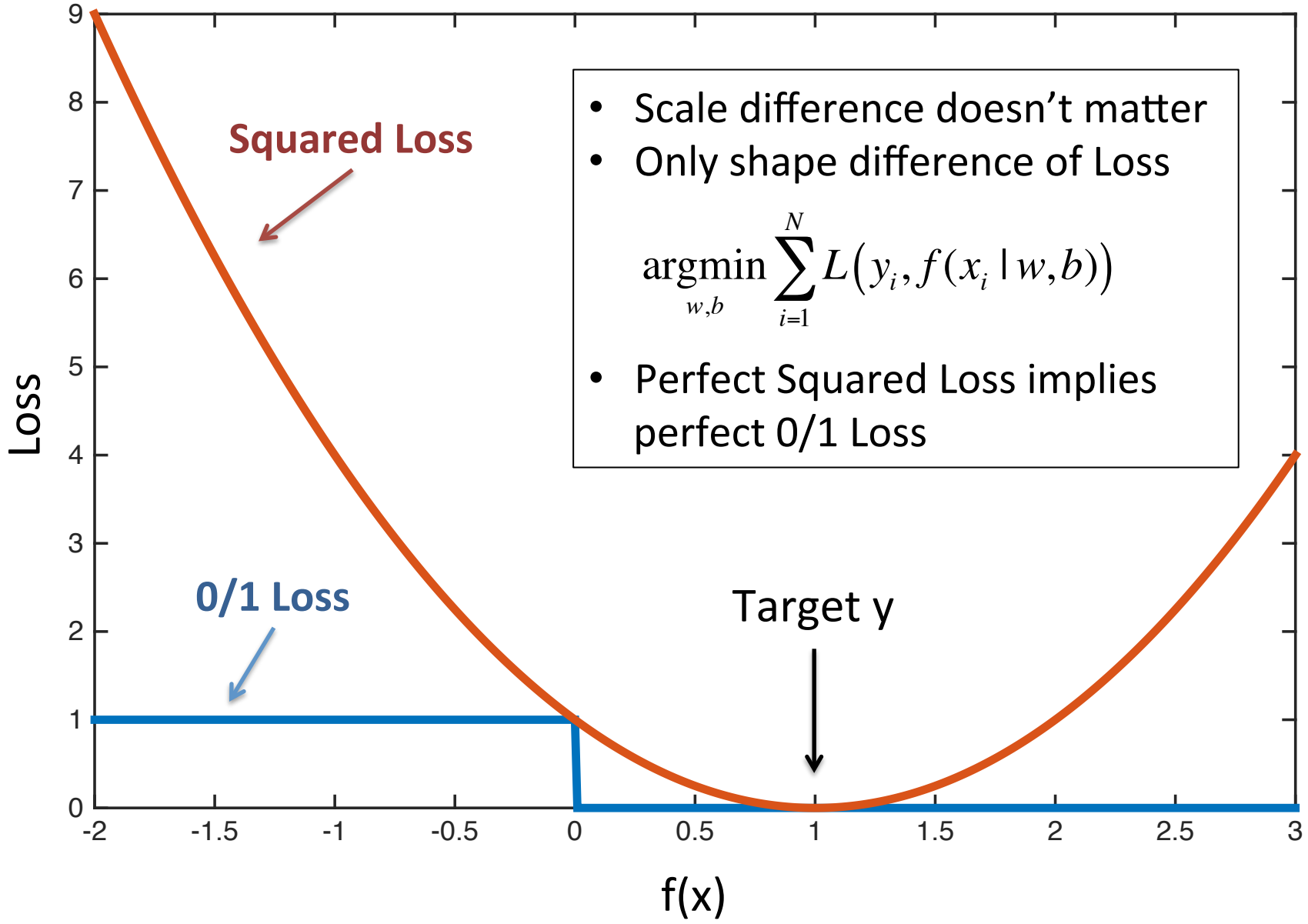
Optimization Problem

Loss Function

- Measures penalty of mis-prediction:
- 0/1 Loss: $L(a,b) = 1_{[a \neq b]}$

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

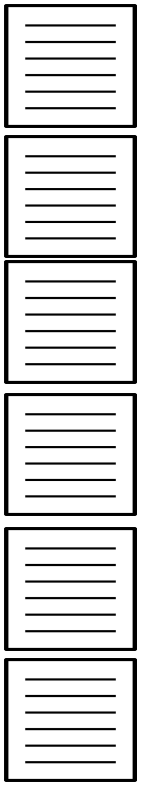
  Classification
- Squared loss: $L(a,b) = (a-b)^2$ **Regression**
- Substitute: a=y, b=f(x)


Squared Loss
0/1 Loss
Target y
Loss
f(x)
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
• Scale difference doesn't matter
• Only shape difference of Loss
$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$
• Perfect Squared Loss implies perfect 0/1 Loss

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

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

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

- Typically, requires optimization algorithm.

- Simplest: **Gradient Descent**

Loop for T iterations

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

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

Gradient Review

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

See Recitation on Wednesday!

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

Linearity of Differentiation

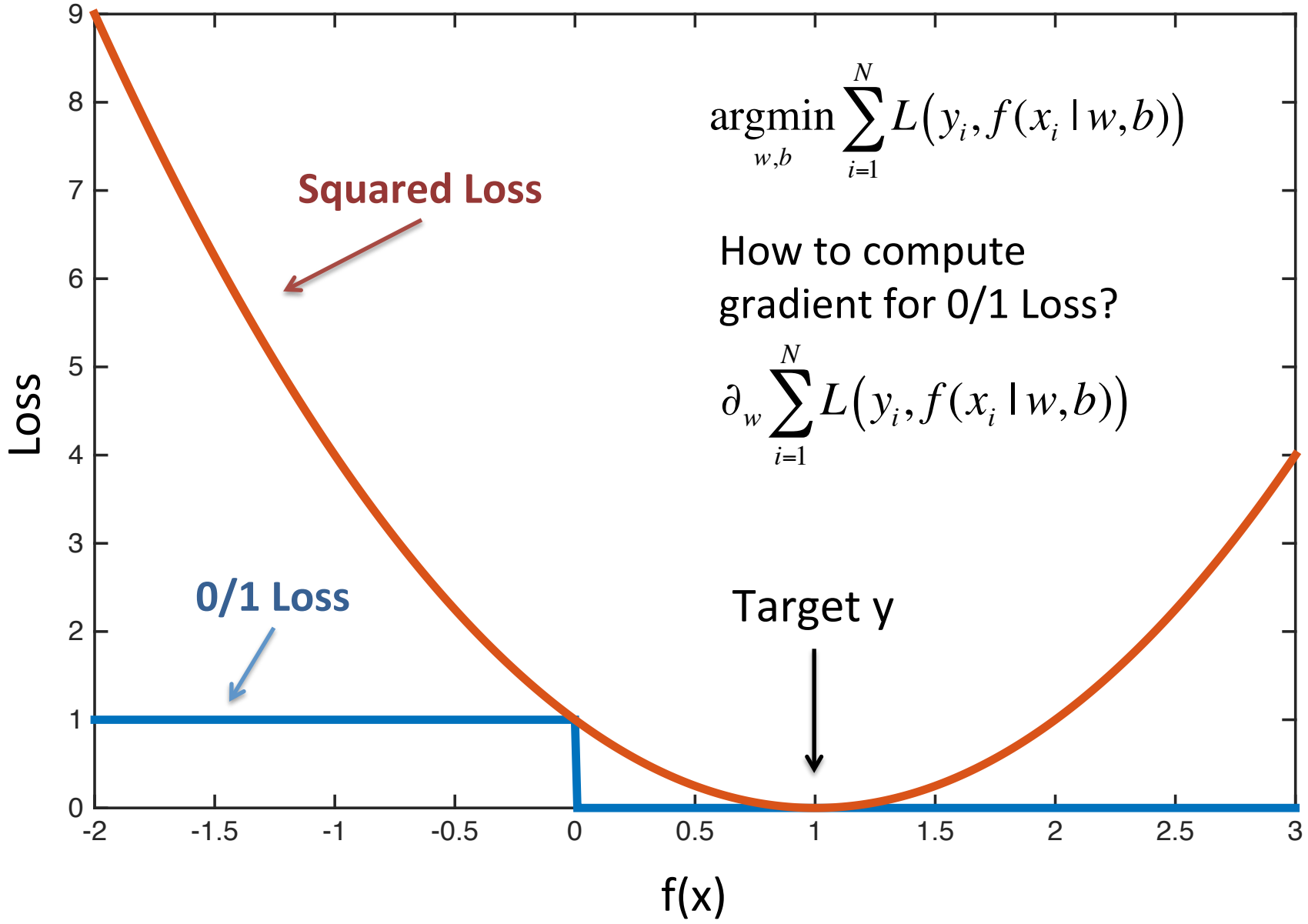
$$= \sum_{i=1}^{N} -2(y_i - f(x_i \mid w,b))\partial_w f(x_i \mid w,b)$$

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

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?

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

0/1 Loss is Intractable

- 0/1 Loss is flat or discontinuous everywhere
- VERY difficult to optimize using gradient descent
- **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: Basic Recipe

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $x \in R^D$ $y \in \{-1, +1\}$
- 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: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w, b)\right)$

Optimization Problem

Recap: Basic Recipe

- Trainii $x \in R^D$ $y \in \{-1,+1\}$
- Mode Linear Models
- Loss F Squared Loss
- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$

Congratulations!
You now know the basic steps to training a model!

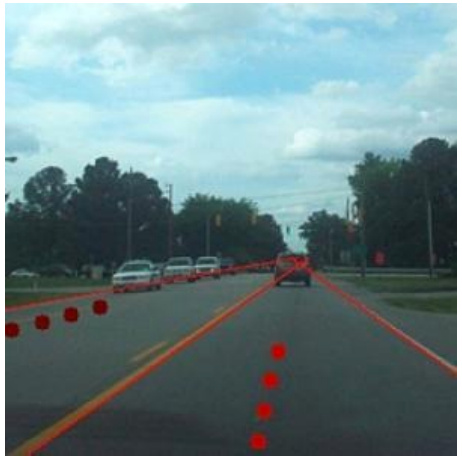
But is your model any good?

Optimization Problem

Example: Self-Driving Cars

Basic Setup

- Mounted cameras
- Use image features

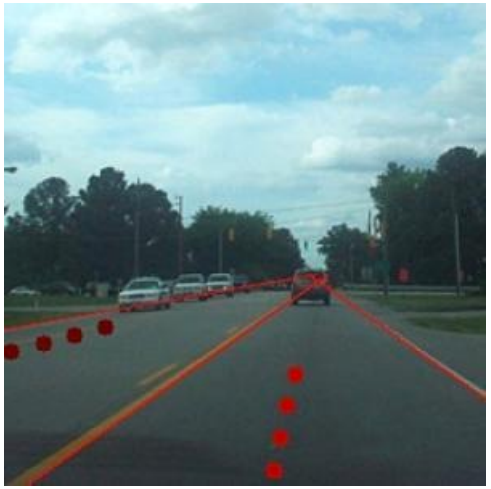


- Human demonstrations

- 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 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:**

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

- **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:

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

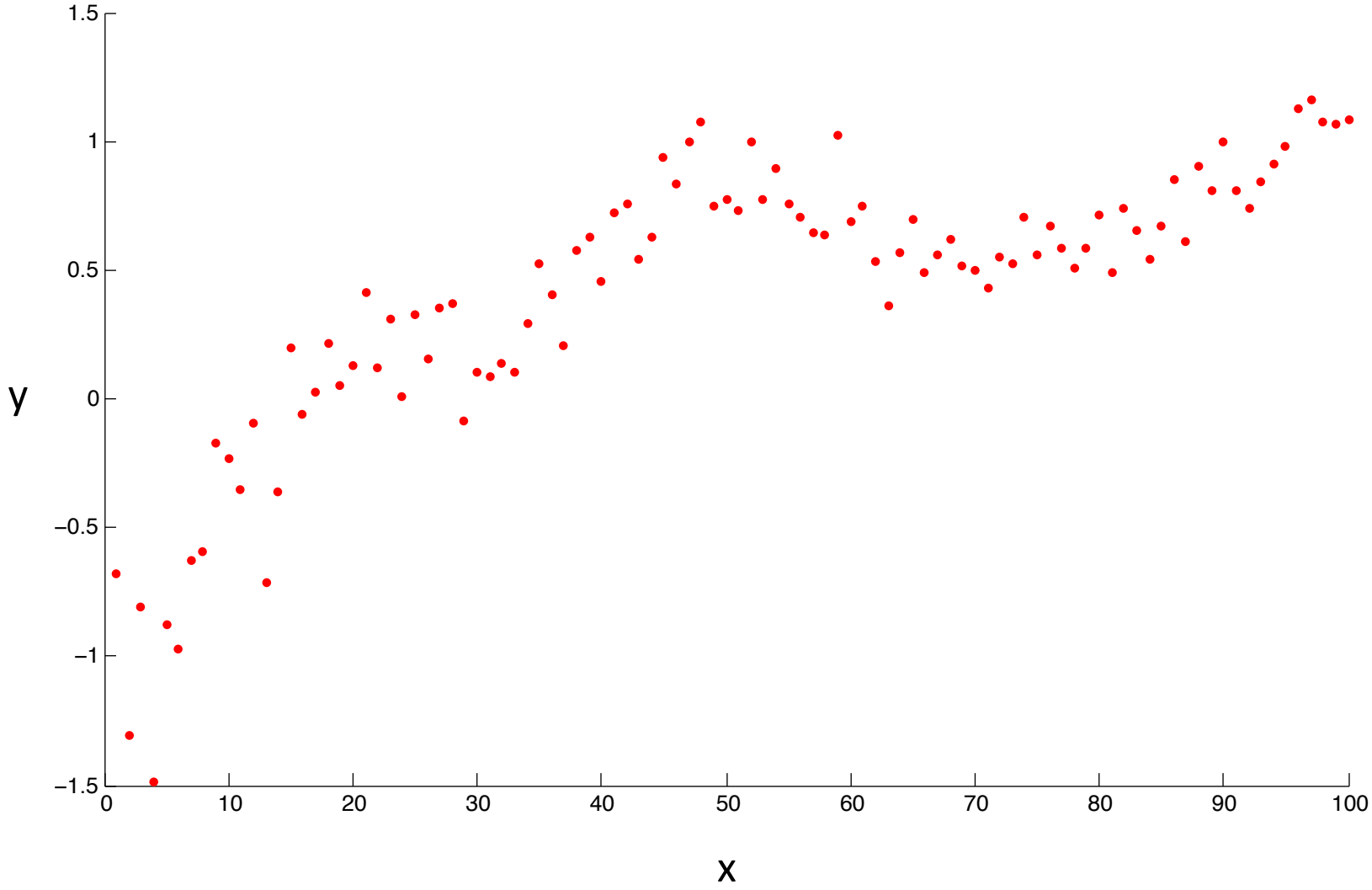
Variance Term

Bias Term

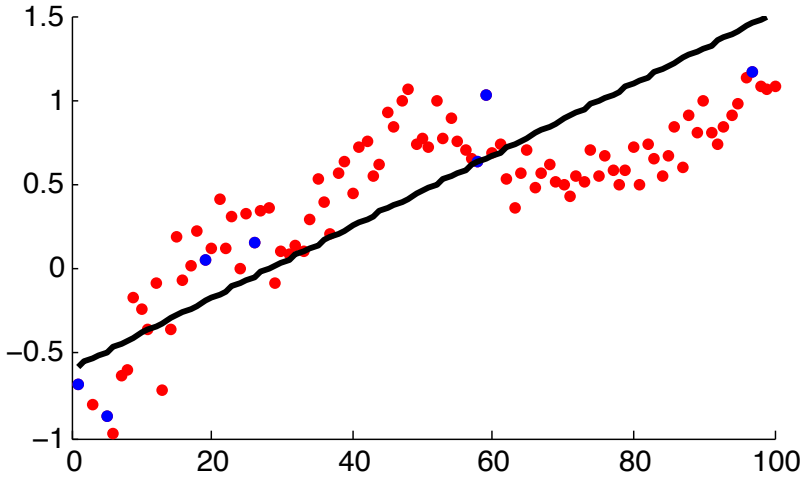
$$F(x) = E_S\left[f_S(x)\right]$$

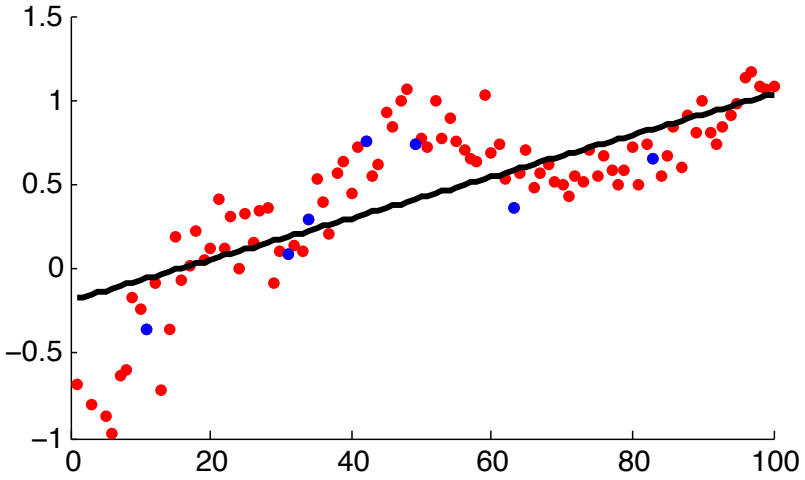
"Average prediction"

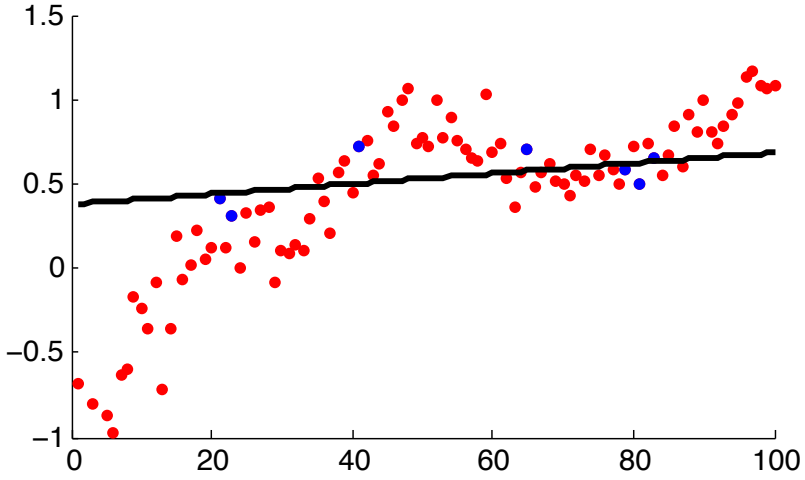
Example P(x,y)

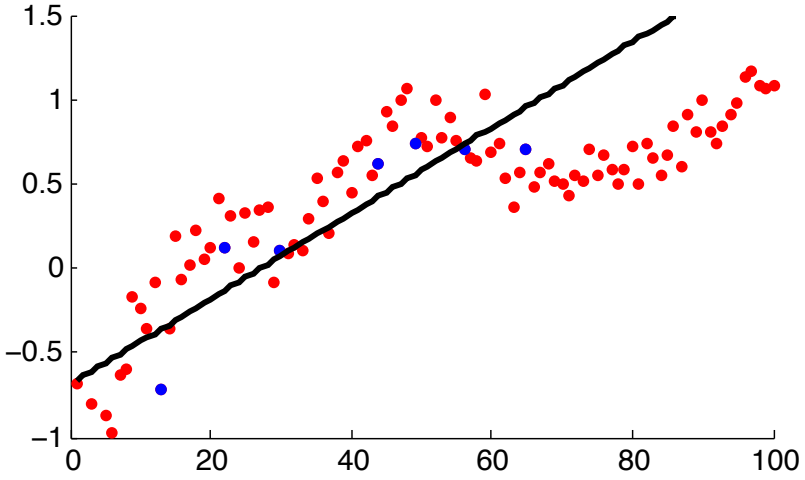


$f_S(x)$ Linear

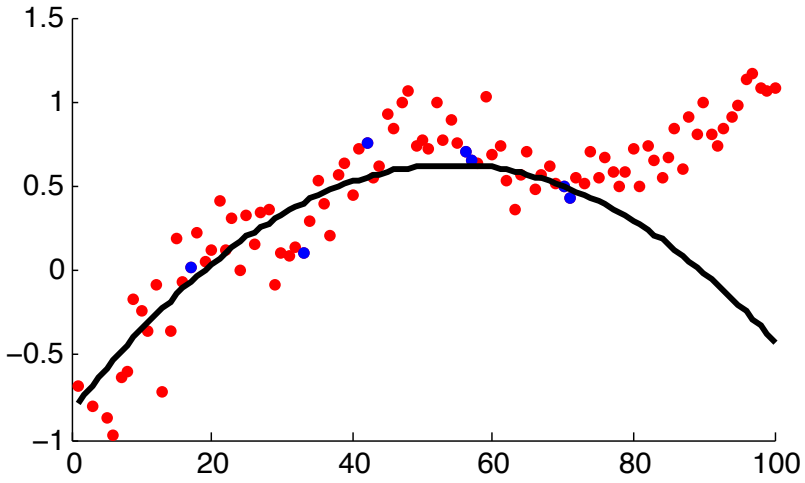

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

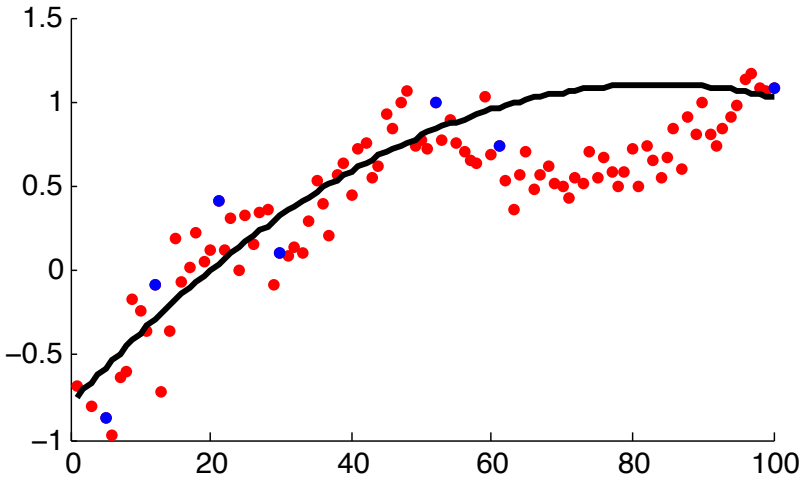

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

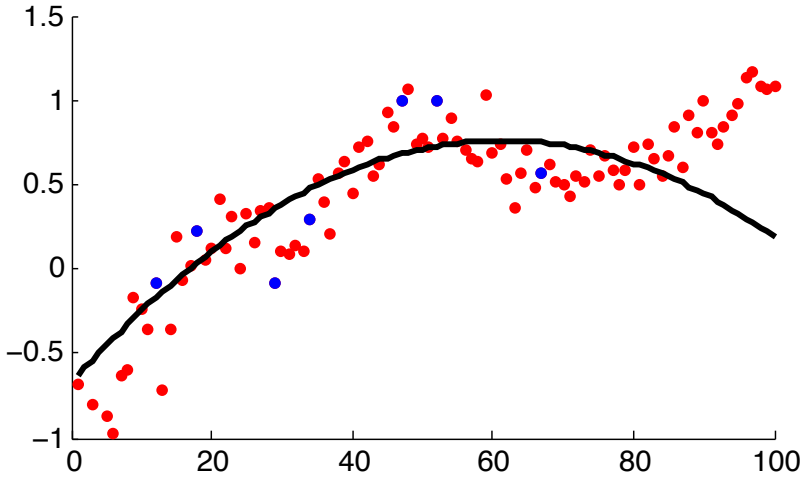

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

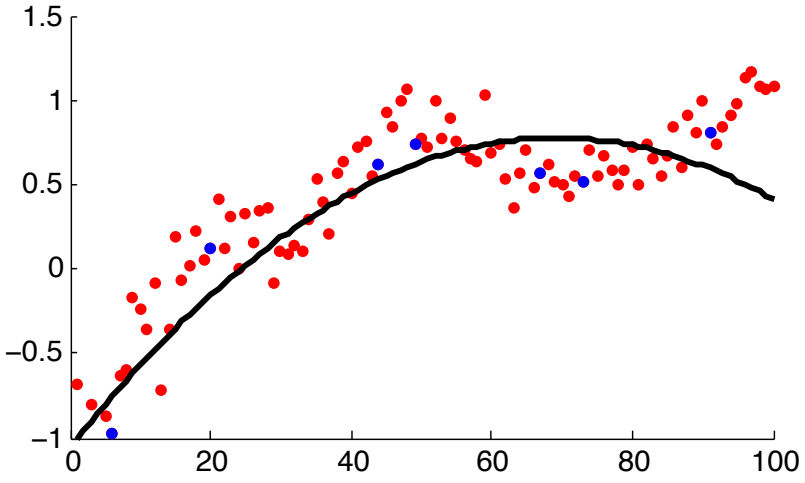

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

$f_s(x)$ Quadratic

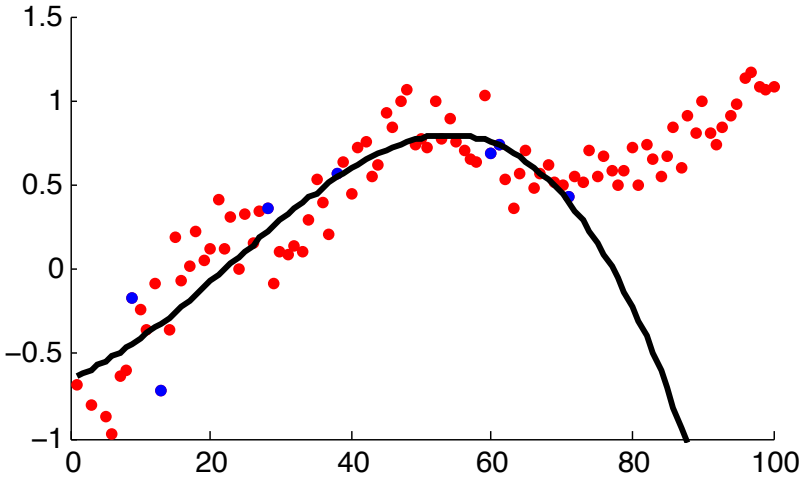


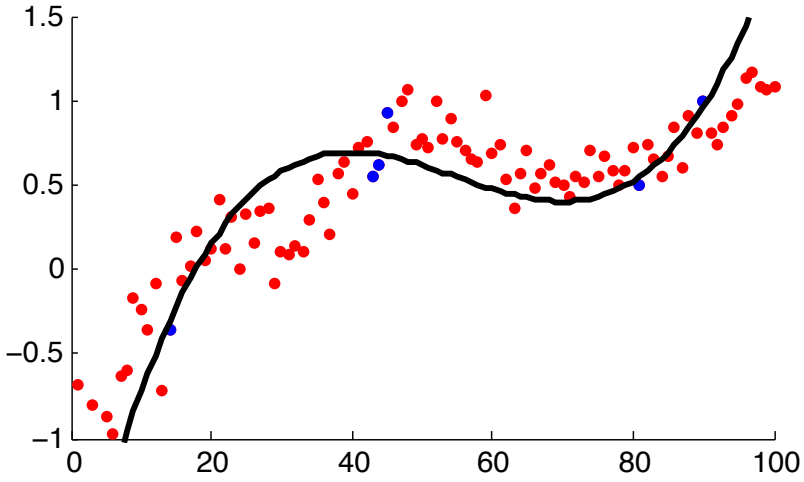


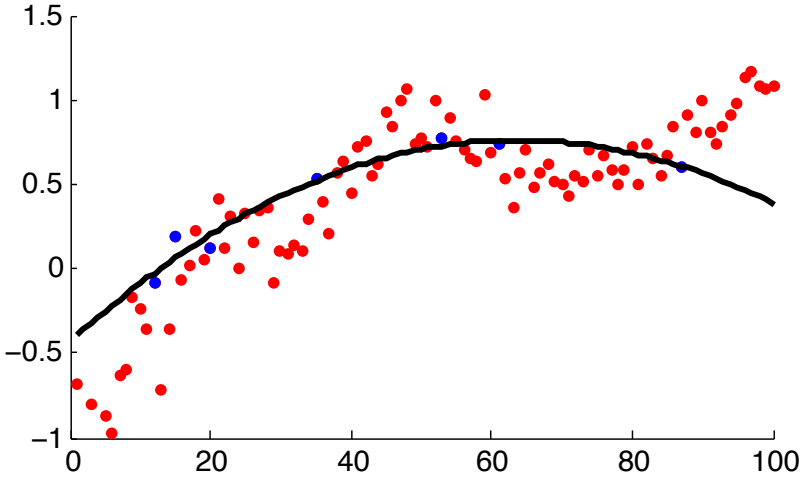


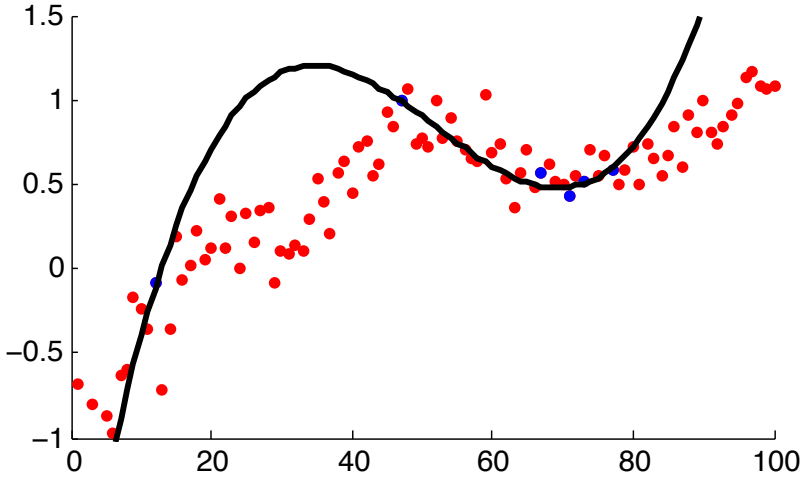


$f_S(x)$ Cubic

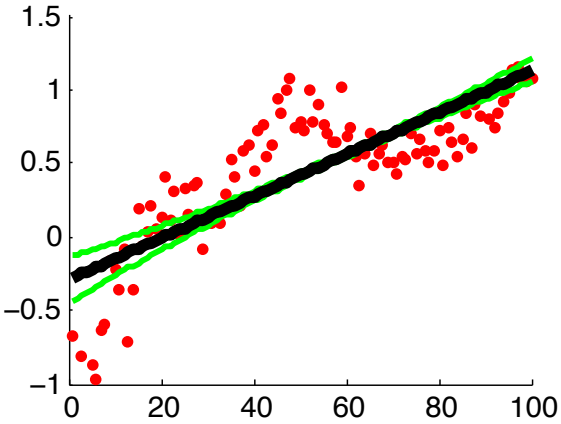

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

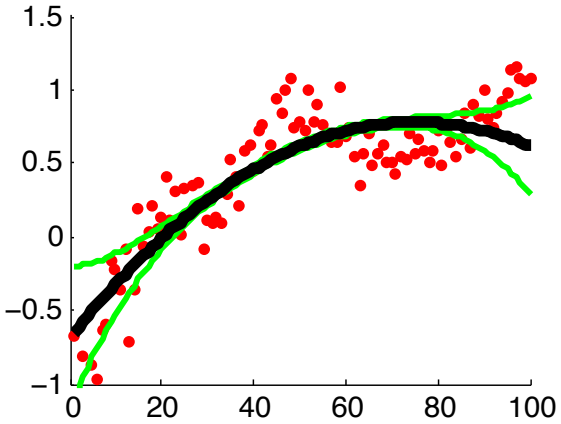

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

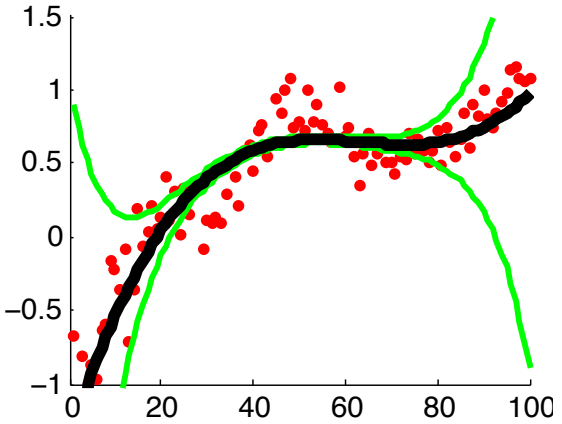

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

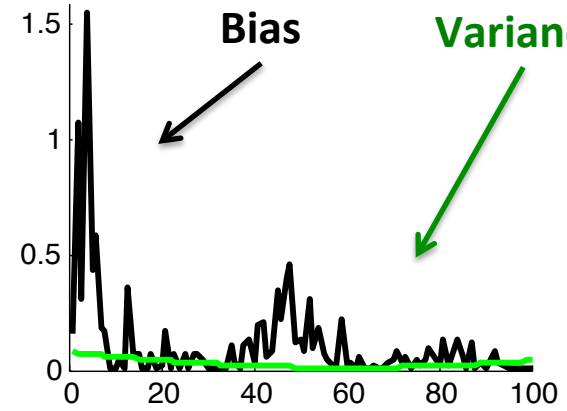

1.5
1
0.5
0
−0.5
−1
0
20
40
60
80
100

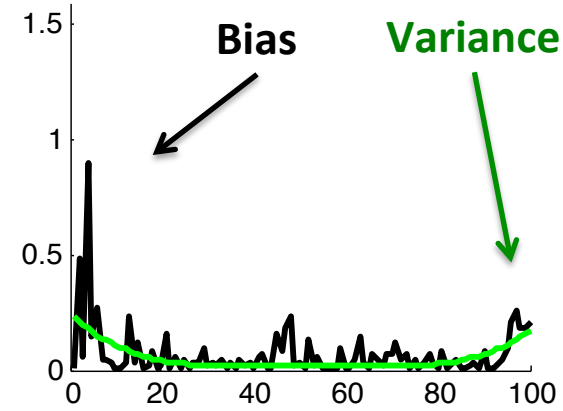
Bias-Variance Trade-off

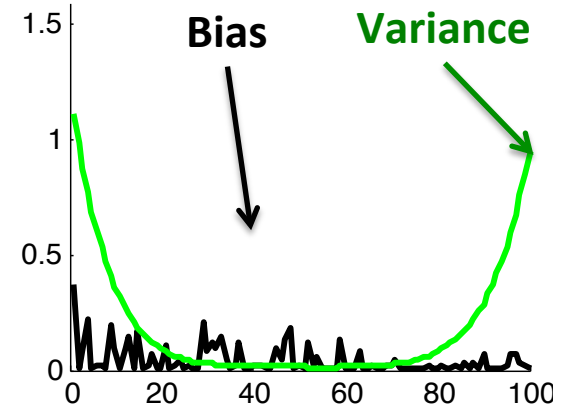




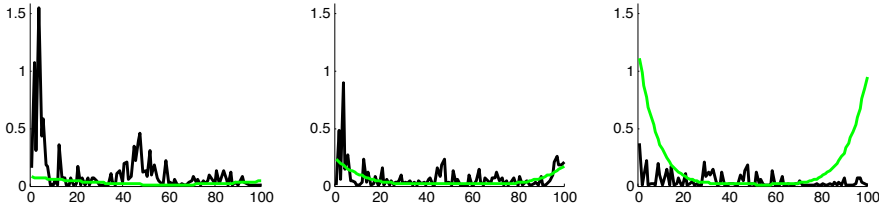






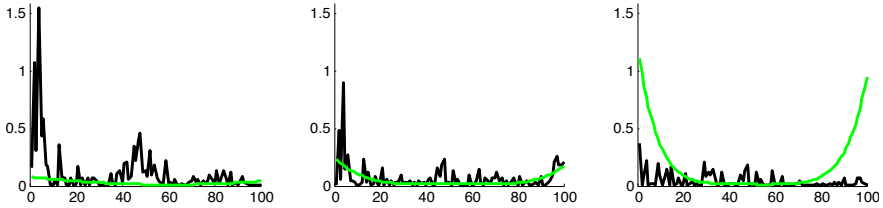


Overfitting vs Underfitting



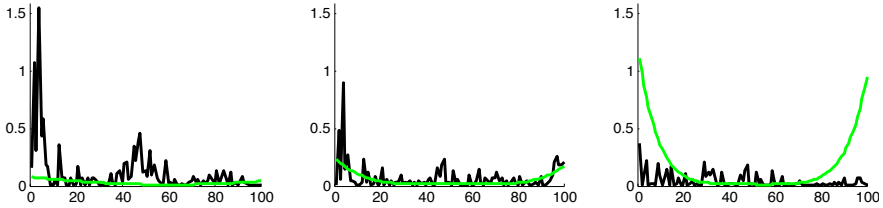
- High variance implies **overfitting**
  - Model class unstable
  - Variance increases with model complexity
  - Variance reduces with more training data.
- High bias implies **underfitting**
  - Even with no variance, model class has high error
  - Bias decreases with model complexity
  - Independent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

Model Selection



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

Model Selection



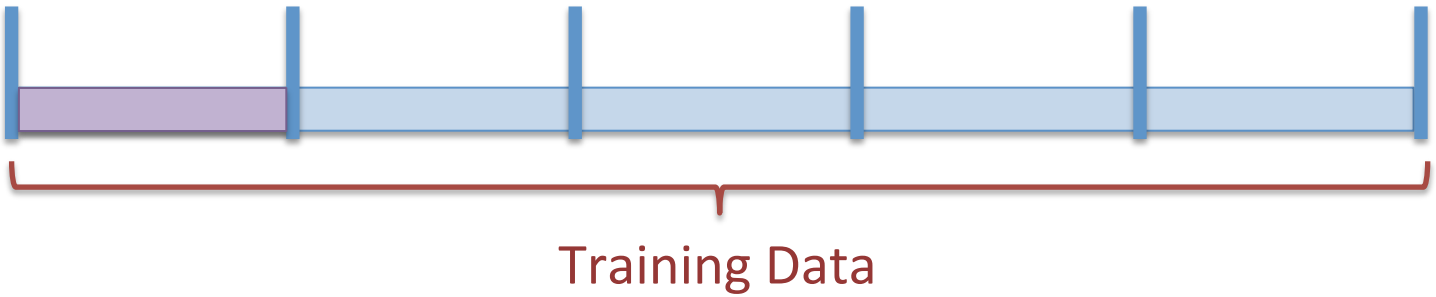
- Fir
- Co
- Sir

But we can't measure generalization error directly!

(We don't have access to the whole distribution.)

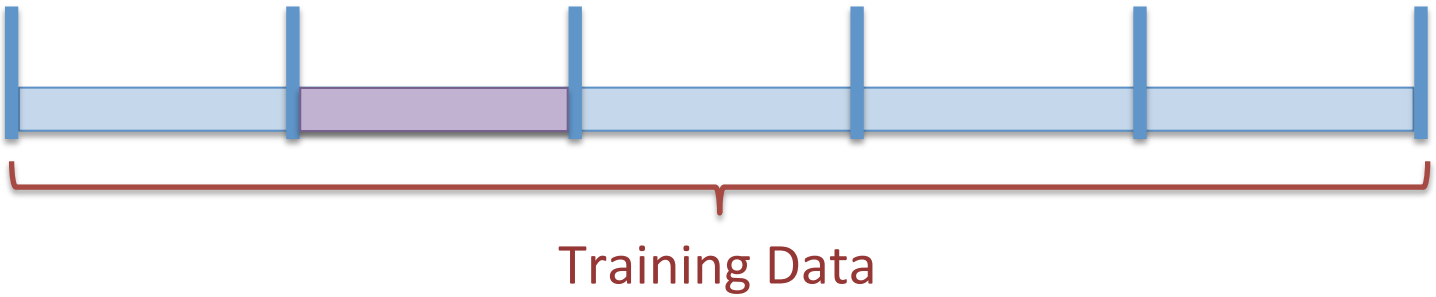
- **Goal:** choose model class with the best generalization error

5-Fold Cross Validation



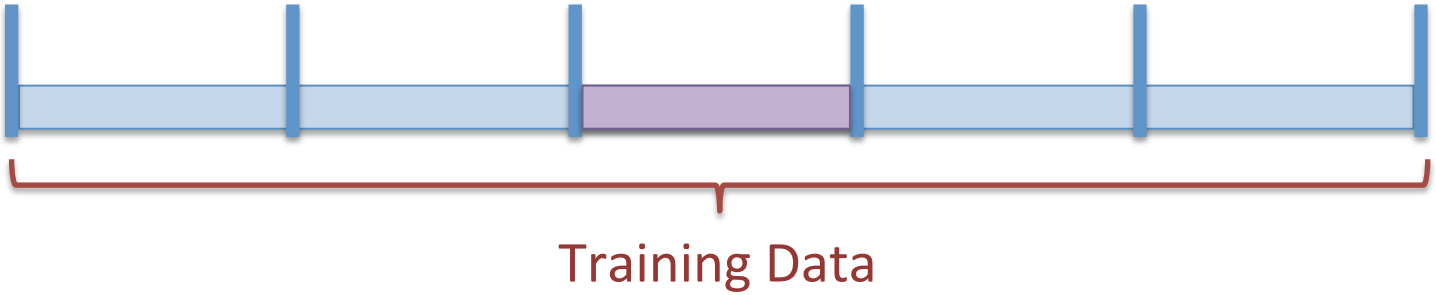
- Split training data into 5 equal partitions
- Train on 4 partitions
- Evaluate on 1 partition
- Allows re-using training data as test data

5-Fold Cross Validation



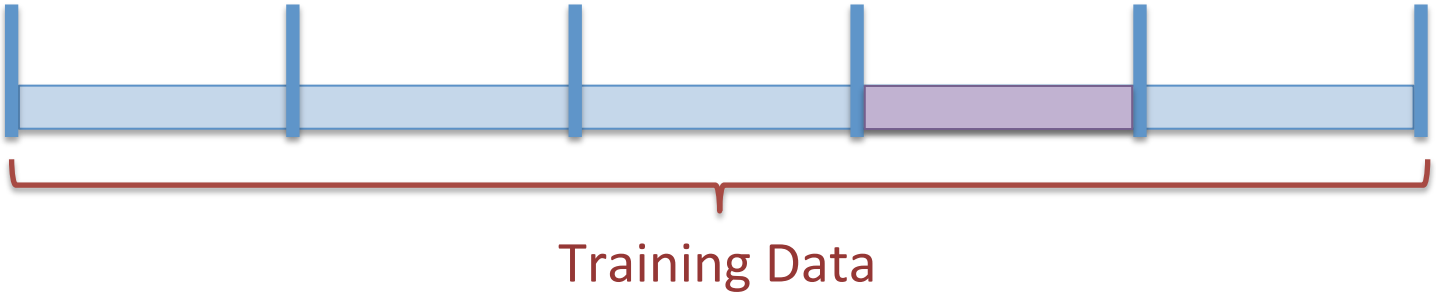
- Split training data into 5 equal partitions
- Train on 4 partitions
- Evaluate on 1 partition
- Allows re-using training data as test data

5-Fold Cross Validation



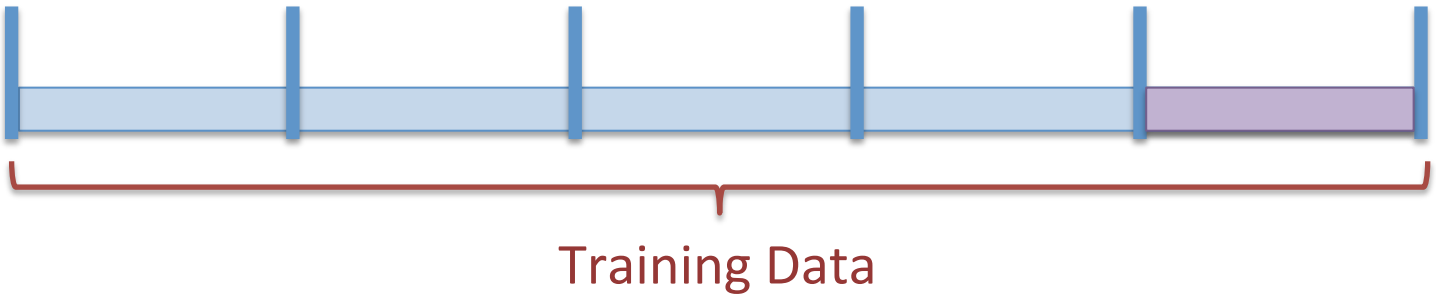
- Split training data into 5 equal partitions
- Train on 4 partitions
- Evaluate on 1 partition
- Allows re-using training data as test data

5-Fold Cross Validation



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

5-Fold Cross Validation



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

Complete Pipeline

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

Training Data

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

Model Class(es)

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

Loss Function

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

Cross Validation & Model Selection

Profit!

Next Lecture

- Beyond basic linear models
  - Logistic Regression, Perceptrons & SVMs
  - Feed-Forward Neural Nets
- Different loss functions
- Different evaluation metrics

- Recitation on Wednesday:
  - Linear Algebra, Vector Calculus & Optimization