

Machine Learning & Data Mining

CMS/CS/CNS/EE 155

Lecture 1:
Administrivia & Basics

Course Info

- Lecture (Tu/Th)
 - 2:30pm – 3:55pm in 105 Annenberg
- Recitation (Th)
 - 7:30pm – 9:00pm in 105 Annenberg
 - As needed (only scheduled first 6 weeks so far)
 - Usually ~1 hour

Staff

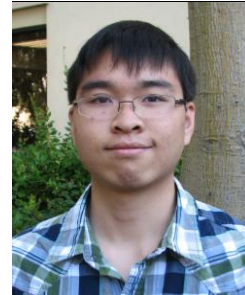
- Instructor: **Yisong Yue**
 - Office in 303 Annenberg
- TAs:



Lucy
Yin



Ritvik
Mishra



Kevin
Tang



Fabian
Boemer

Course Breakdown

- 6 Homeworks, ~60% of final grade
 - Due on Tuesdays via Moodle
 - **Homework 1 will be released tonight.**
 - Due next Tuesday
- 2 Mini-projects, ~24% of final grade
- Final, ~16% of final grade

Regarding Homework 1

- If you have prior experience with CS 156
 - Should be pretty straightforward (3-4 hours)
- If you do not...
 - Might take a while (8-10 hours?)
 - But this will mostly catch you up if you survive
 - Should consider dropping class if too hard

Late Submission Policy

- Up to 48 free late hours
- Afterwards:
 - 50% penalty for one day late
 - Submissions beyond one day late not accepted
- 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.
- Adhere to the Academic Integrity
 - 50ft policy

Course Website

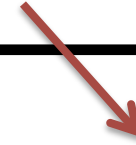
- <http://www.yisongyue.com/courses/cs155>
- Linked to from my website:
 - <http://www.yisongyue.com>
- Up-to-date office hours
- Lecture notes, additional reading, homeworks, etc.

Moodle & Piazza

- Moodle:
 - <https://courses.caltech.edu/course/view.php?id=2132>
 - Linked to from course website
 - Submission, Solutions, Grades
 - Video Recordings of Lectures
 - Requires Enrollment Key
- Piazza
 - <https://piazza.com/class/iigtx6k96rvy>
 - Linked to from course website
 - Discussion Forum (use it!)

Machine Learning & Data Mining

Computer Algorithm



Process of Converting

Data & Experience

Into **Knowledge**



Computer Model

Machine Learning vs Data Mining

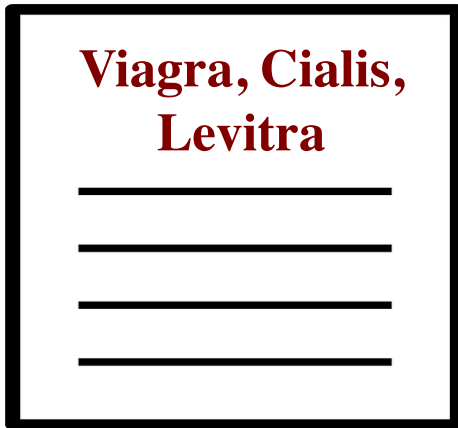
- **ML focuses more on algorithms**
 - Typically more rigorous
 - Also on analysis (learning theory)
- **DM focuses more on knowledge extraction**
 - Typically uses ML algorithms
 - Knowledge should be human-understandable
- **Huge overlap**

Course Outline

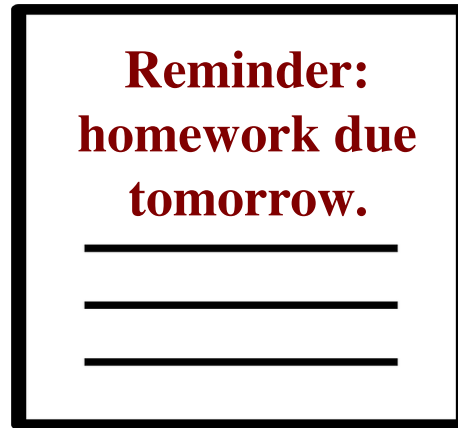
- Review over basics (this week)
- Modern techniques:
 - Lasso
 - Ensemble Methods (e.g., Boosting)
 - Sequence Prediction
 - Embeddings, Latent Factor Models
 - Deep Learning

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
}
```

Viagra
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!

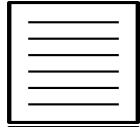
Machine Learning to the Rescue!

Training Set



SPAM!

Build a Generic Representation

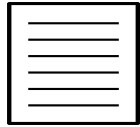


SPAM!



NOT SPAM

Run a Generic Learning Algorithm



NOT SPAM

→ Classification Model



SPAM!



SPAM!

⋮



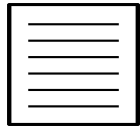
Labeled by Humans

Bag of Words Representation

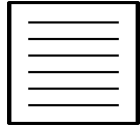
Training Set



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:

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

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

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

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

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

$$b = 1.5$$

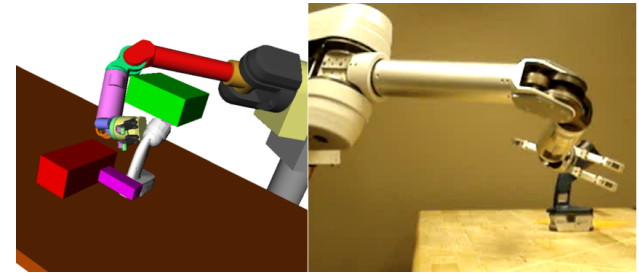
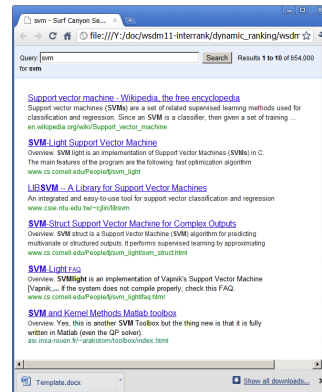
Training Set

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 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

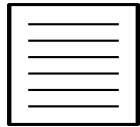
Bag of Words



SPAM!

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

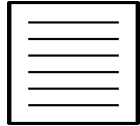
$f(x|w,b) = +0.5$



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

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

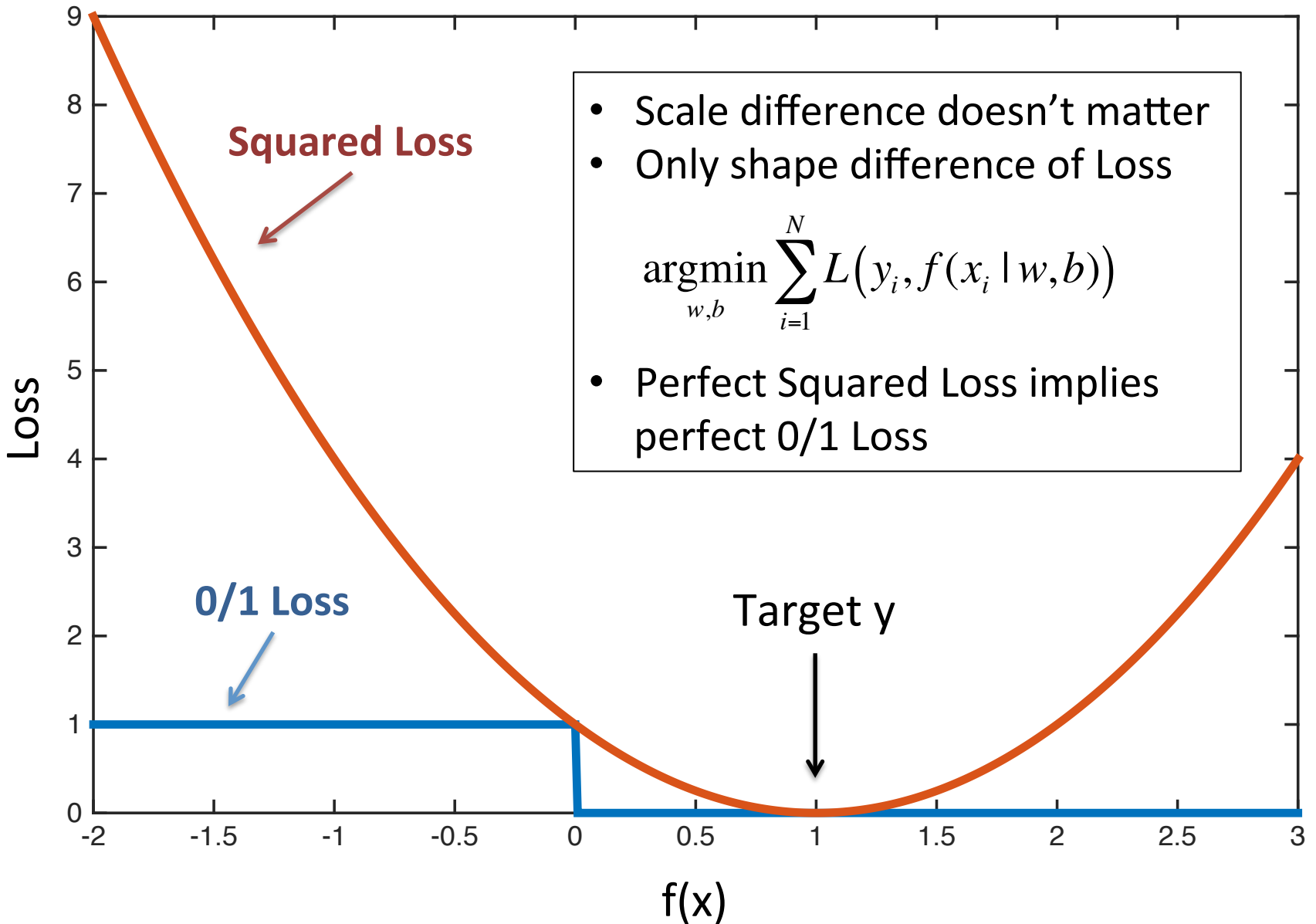
Loss Function

- 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

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

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

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

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

Loop for T
iterations

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

Gradient Review

**More Details
Next Lecture**

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

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

Linearity of Differentiation

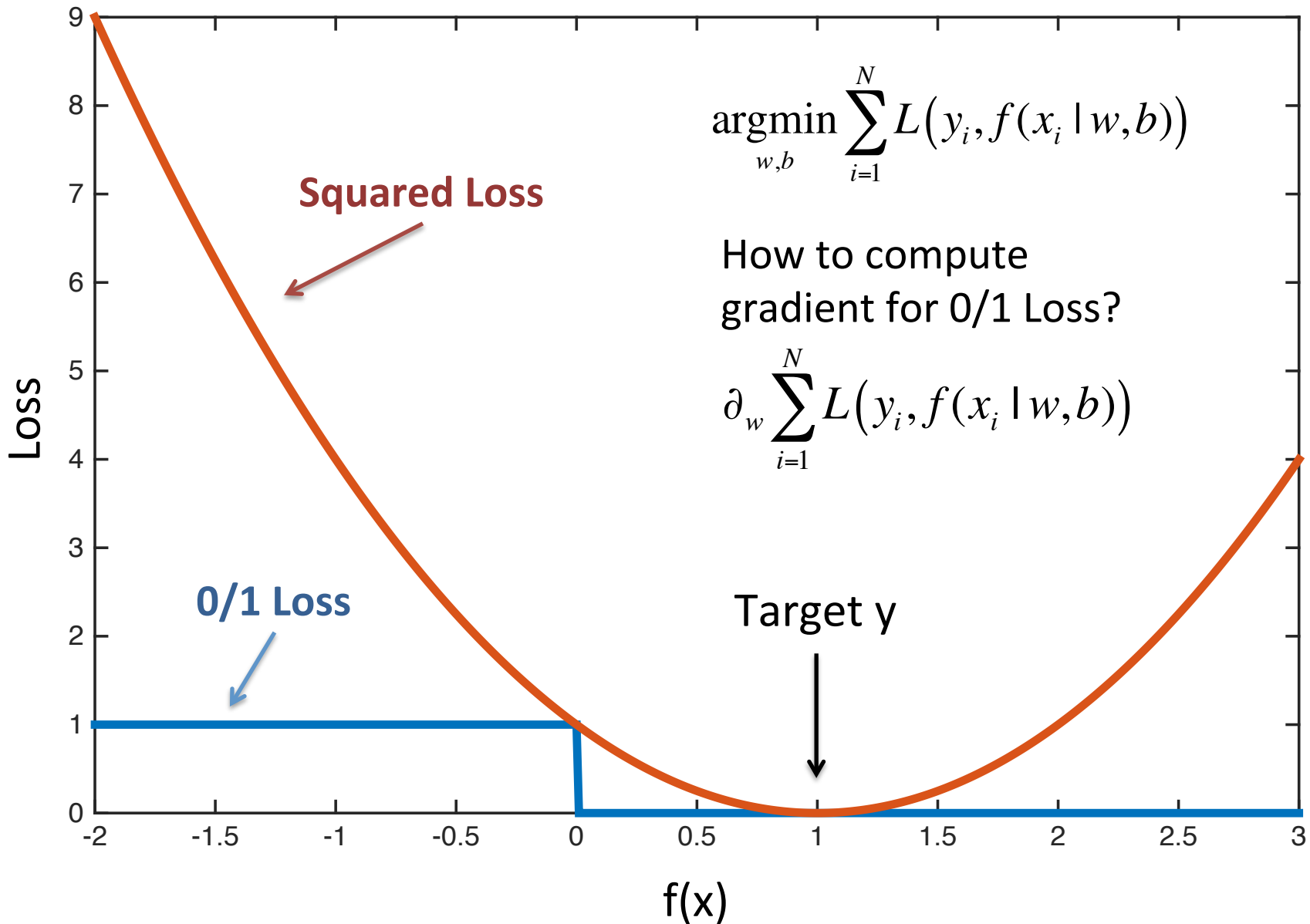
$$= \sum_{i=1}^N -2(y_i - f(x_i | w, b)) \partial_w f(x_i | w, b)$$

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

Chain Rule

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

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



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: Basic Recipe

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

- Training

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

$x \in \mathbb{R}^D$
 $y \in \{-1, +1\}$

- Model

Linear Models

- Loss F

But is your model any good?

Squared Loss

- Learning Objective:

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

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 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 \gg 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:**

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

- **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:

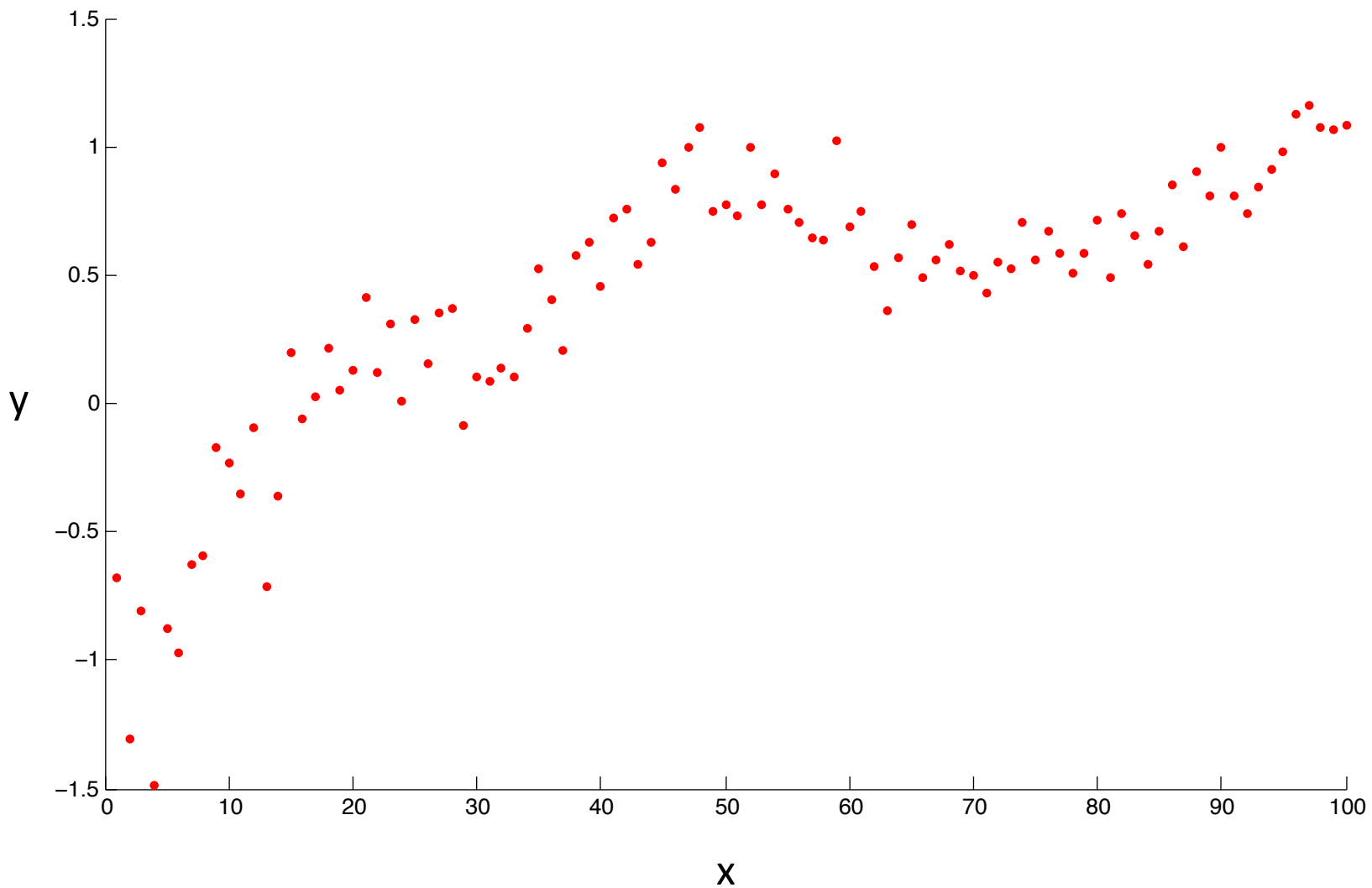
$$E_S [L_P(f_S)] = E_{(x,y) \sim P(x,y)} \left[\underbrace{E_S [(f_S(x) - F(x))^2]}_{\text{Variance Term}} + \underbrace{(F(x) - y)^2}_{\text{Bias Term}} \right]$$

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

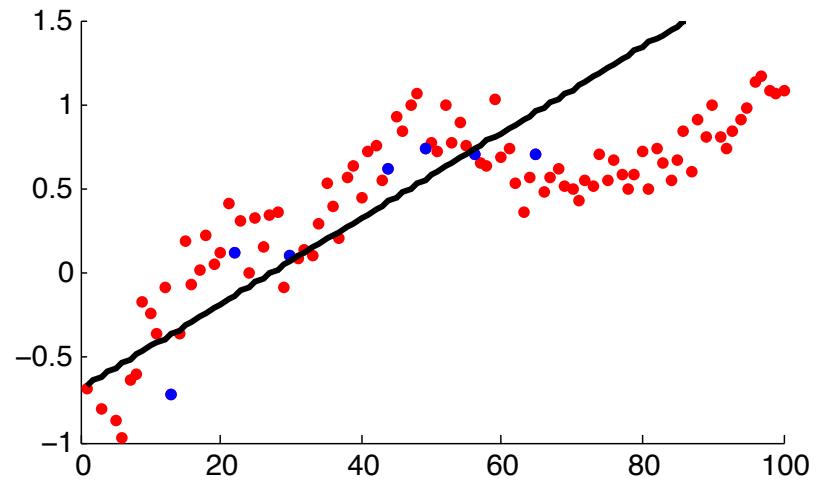
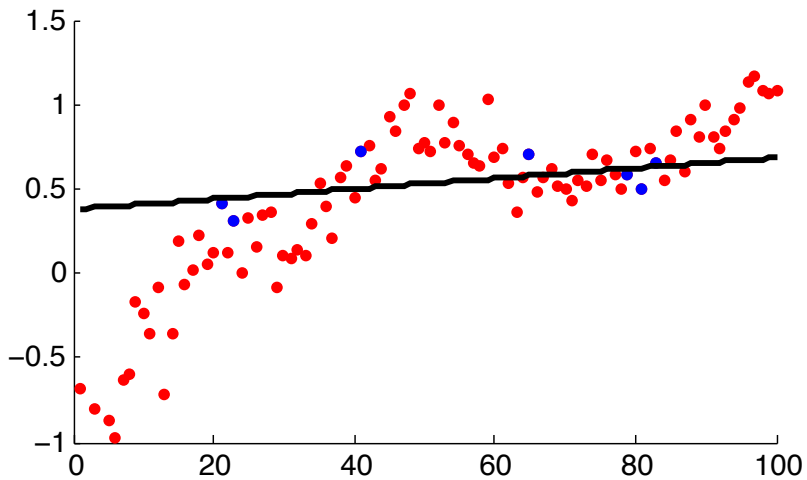
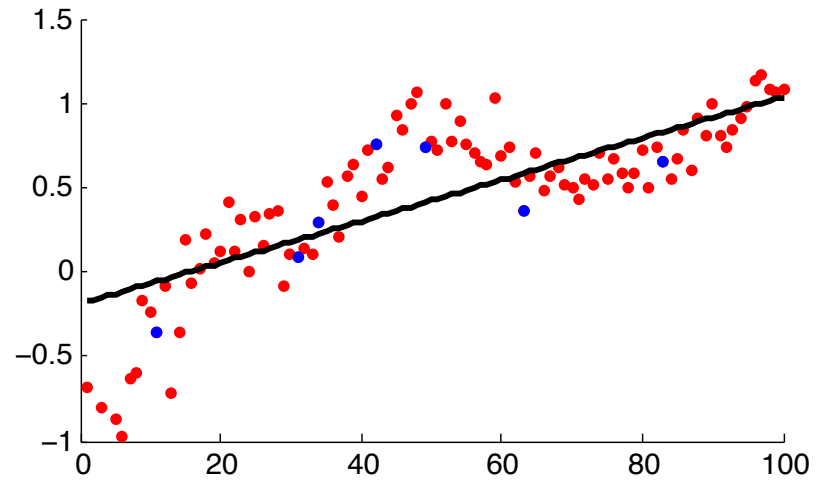
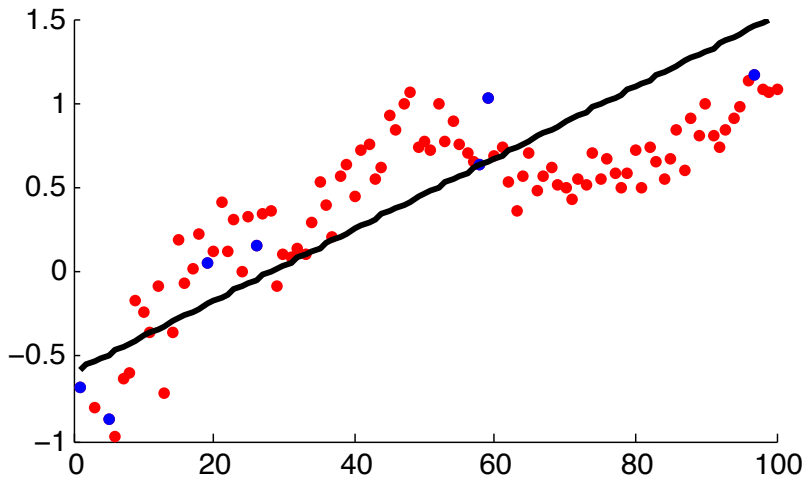


“Average prediction”

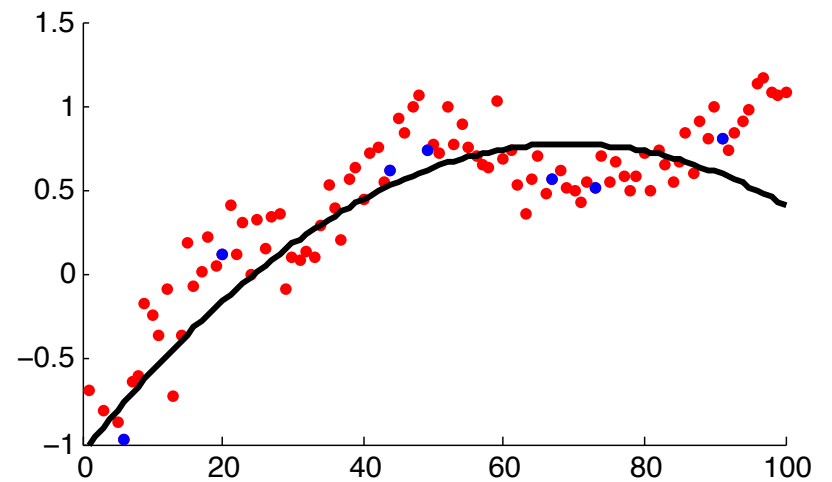
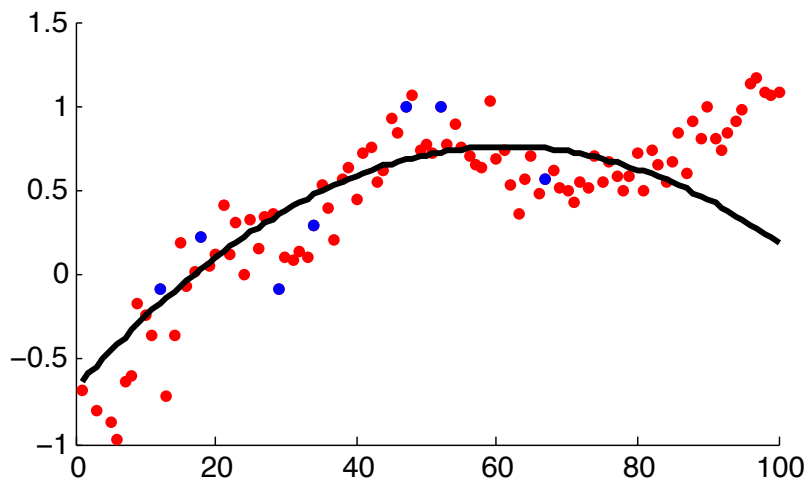
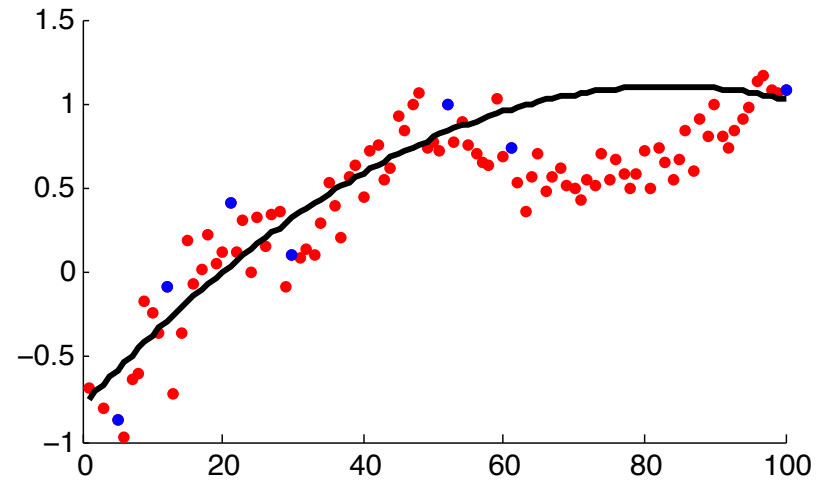
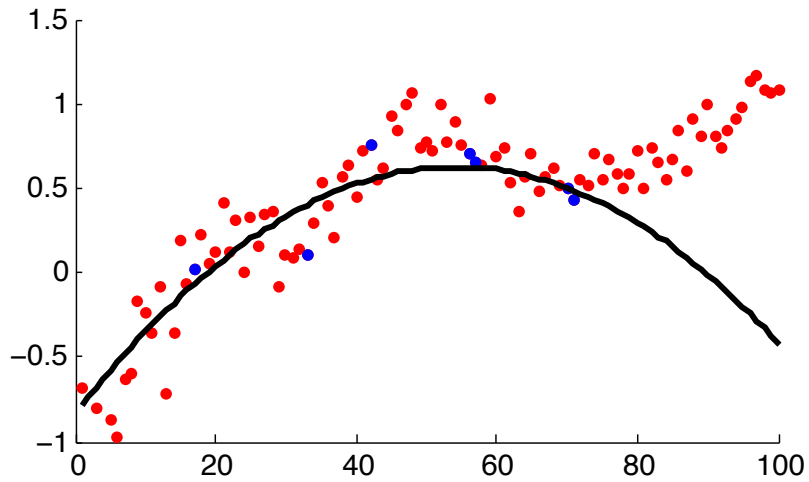
Example $P(x,y)$



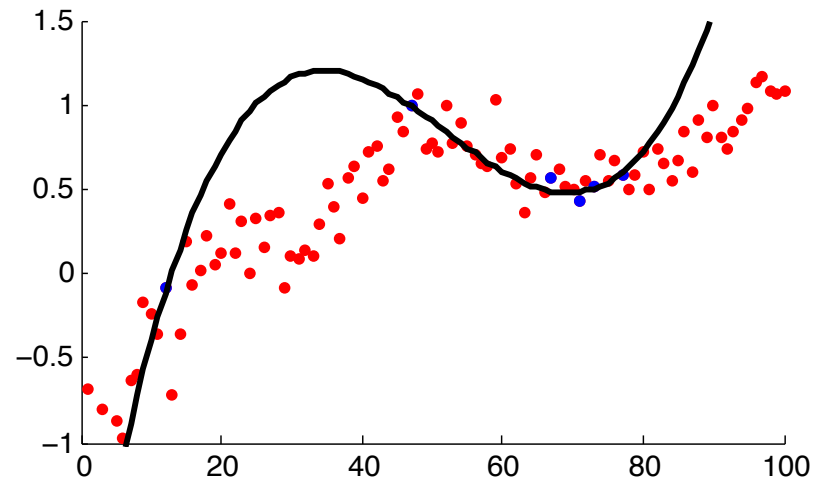
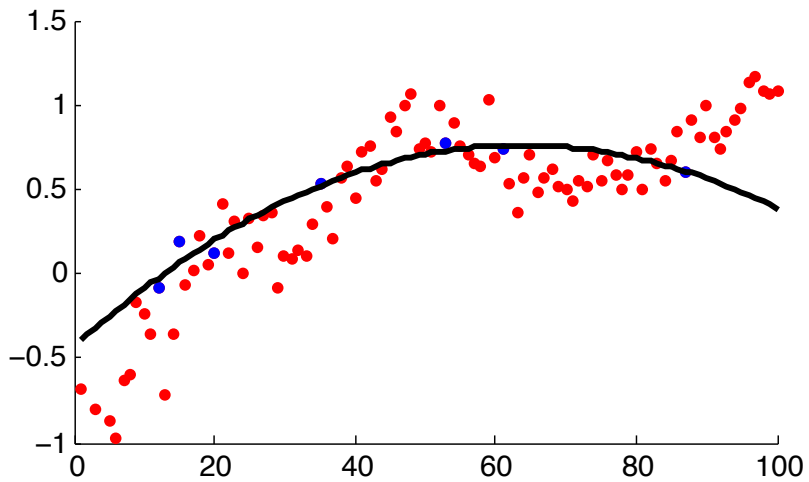
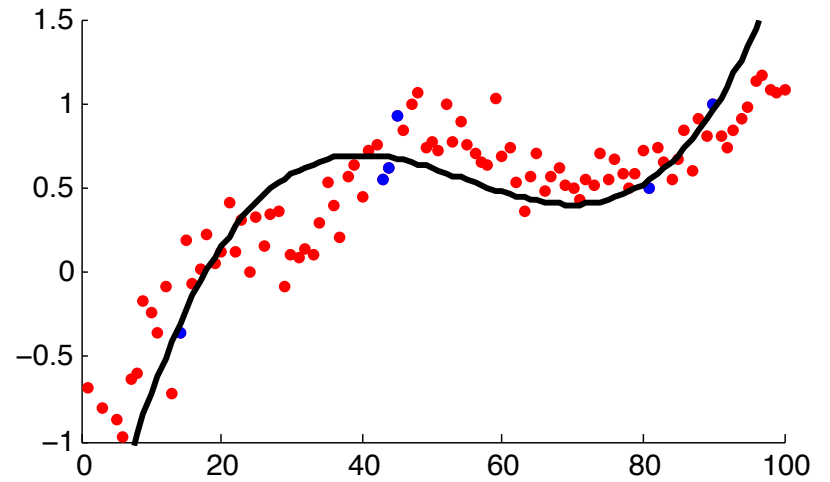
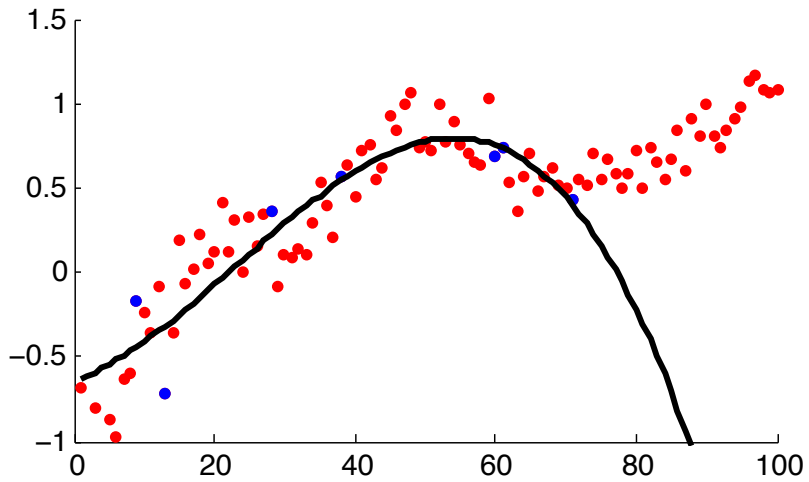
$f_S(x)$ Linear



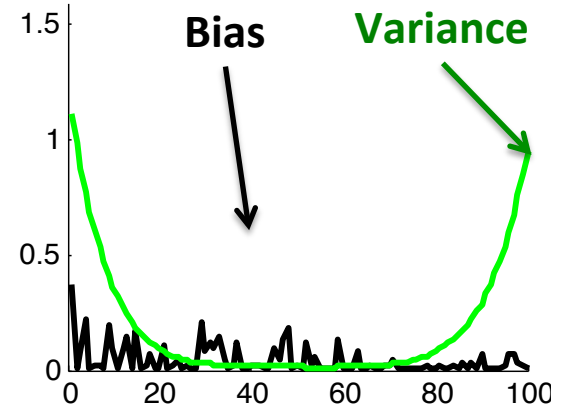
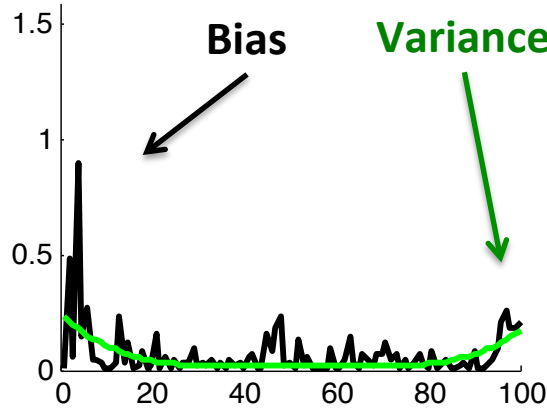
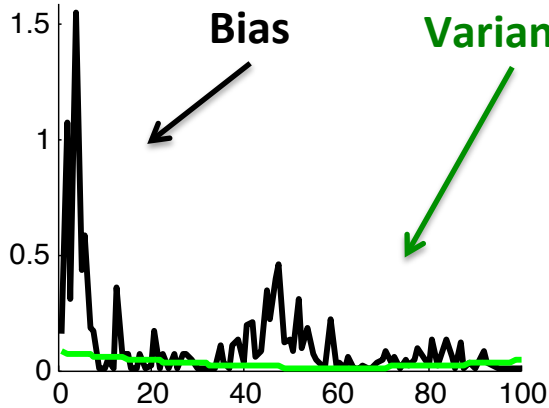
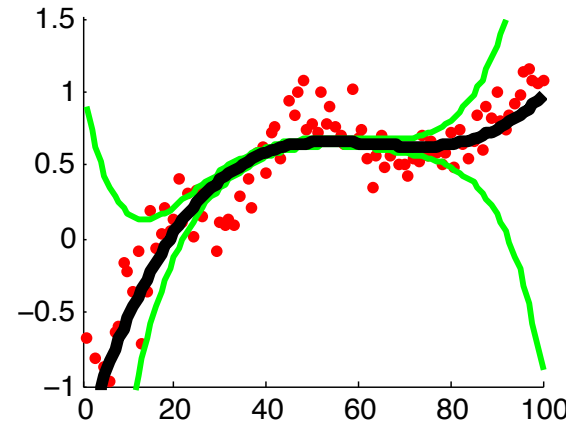
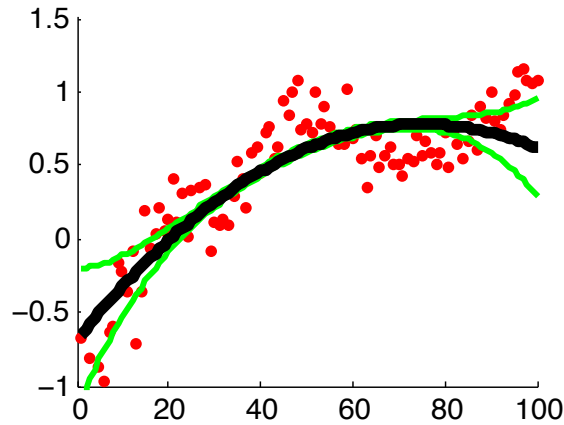
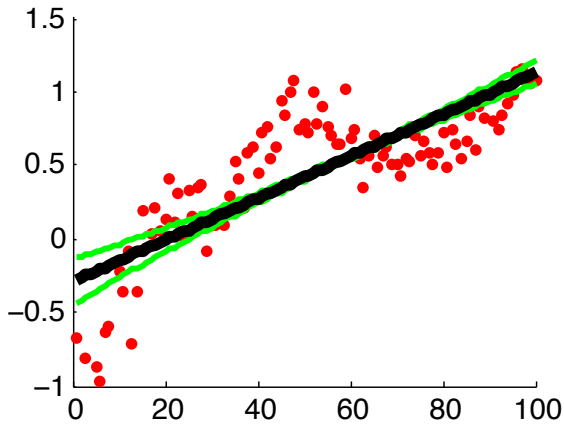
$f_S(x)$ Quadratic



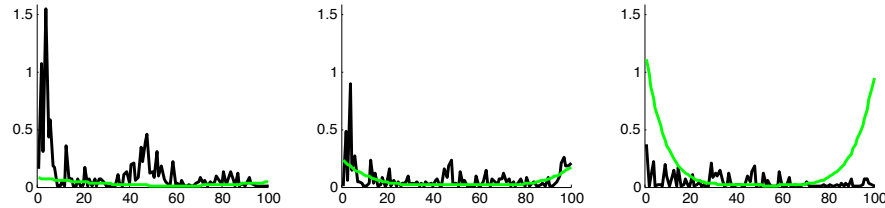
$f_S(x)$ Cubic



Bias-Variance Trade-off

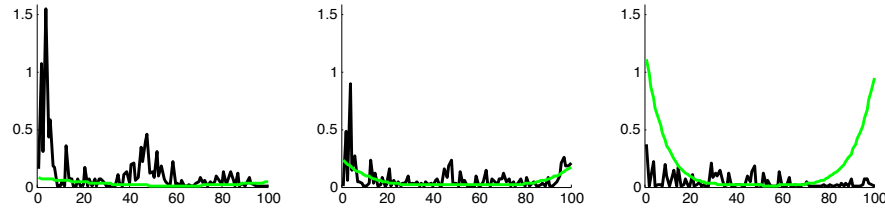


Overfitting vs Underfitting



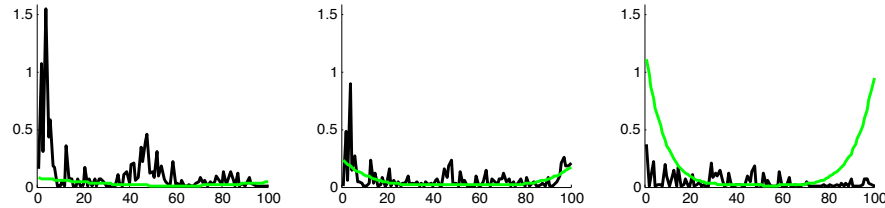
- High variance implies **overfitting**
 - Model class unstable
 - Variance increases with model complexity
 - Variance reduces with more training data.
- High bias implies **underfitting**
 - Even with no variance, model class has high error
 - Bias decreases with model complexity
 - Independent of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

Model Selection



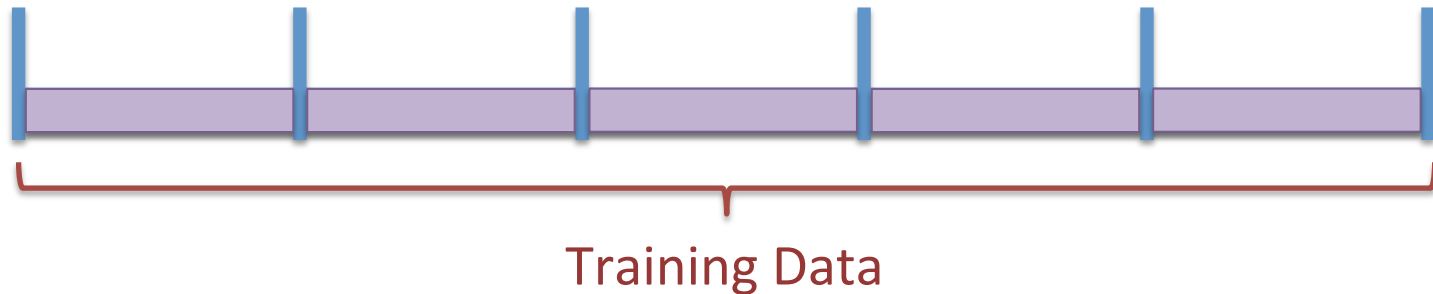
- Finite training data
- Complex model classes overfit
- Simple model classes underfit
- **Goal:** choose model class with the best generalization error

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

Complete Pipeline

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

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