CS 760: Machine Learning
Regression I

Kirthi Kandasamy

University of Wisconsin-Madison

February 13, 2023

Logistics

- **Annoucements**:
  - HW2 due on Wednesday at 10am.
  - HW3 will be released this week

Outline

- **Evaluation: Metrics**
  - Cross validation, ROC curves, precision/recall
- **Linear Regression**
  - Setup, normal equations, GD-based solution
- **Logistic Regression**
  - Linear classification, maximum likelihood estimation, setup, comparisons

Outline

- **Evaluation: Metrics**
  - Cross validation, ROC curves, precision/recall
- **Linear Regression**
  - Setup, normal equations, GD-based solution
- **Logistic Regression**
  - Linear classification, maximum likelihood estimation, setup, comparisons

On cross validation

- Can cross validation be used for reporting test scores?
  - Yes, but keep in mind that you are evaluating a method and not a specific hypothesis.
  - If you want to report the test performance of a hypothesis, do **not** use cross validation.

- Can cross validation be used for model selection (hyperparameter tuning)?
  - Yes!

Confusion Matrices: 2-Class Version

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Accuracy: Sufficient?

Accuracy may not be useful measure in cases where

- There is a large class skew
  - Is 98% accuracy good when 97% of the instances are negative?
- There are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
  - Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

Other Metrics

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

True Positives vs. False Positives

- Want: high true positive rate, low false positive rate.

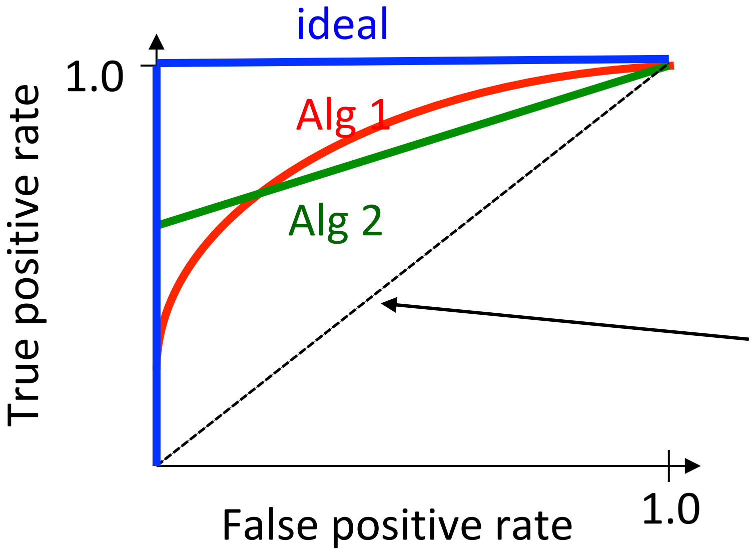
$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

- Say our classifier has a threshold that we can tune
  - Outputs a value. We vote + if beyond threshold
  - Tune very high: **no false positives.** But, **low recall**
  - Tune very low: **good recall**. But, **high false positives**
    - Balance?

Other Metrics: ROC Curves

- A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as the thresholding value is varied.

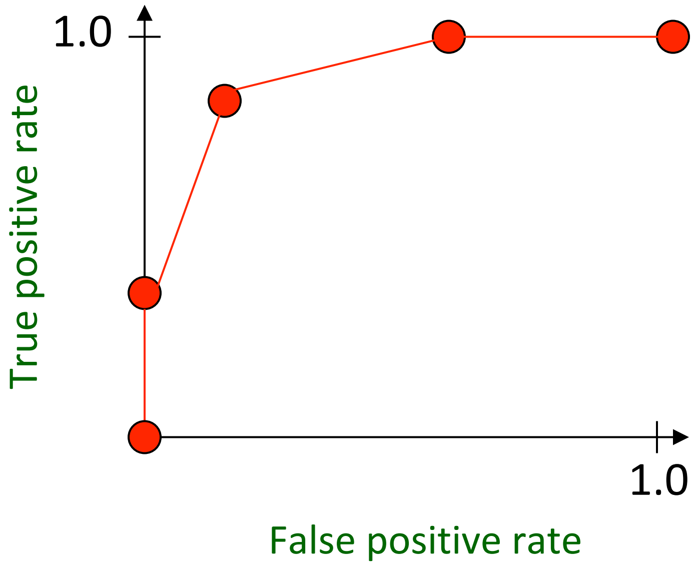


expected curve for random guessing

Different methods can work better in different parts of ROC space.

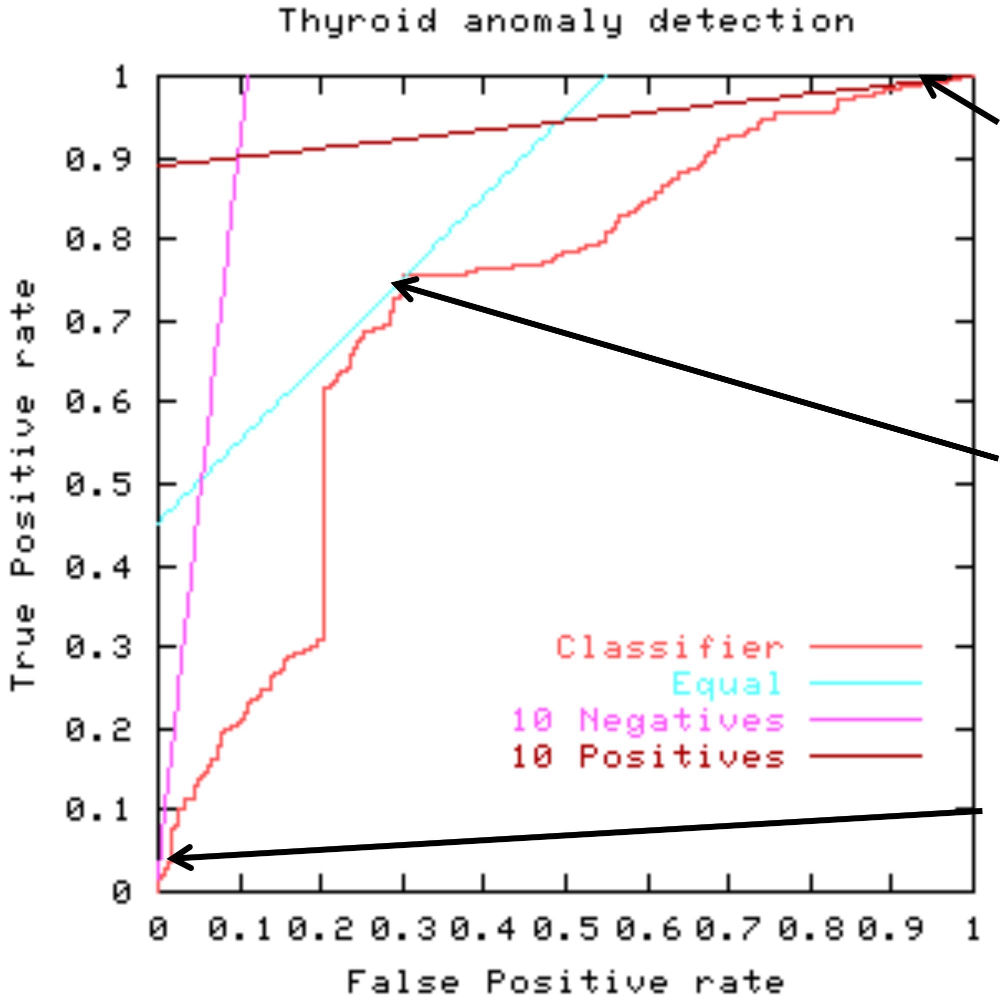
ROC Curves: Plotting

instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



ROC Curves: Misclassification Cost

- The best operating point depends on relative cost of FN and FP misclassifications



best operating point when FN costs 10× FP

best operating point when cost of misclassifying positives and negatives is equal

best operating point when FP costs 10× FN

Other Metrics: Precision

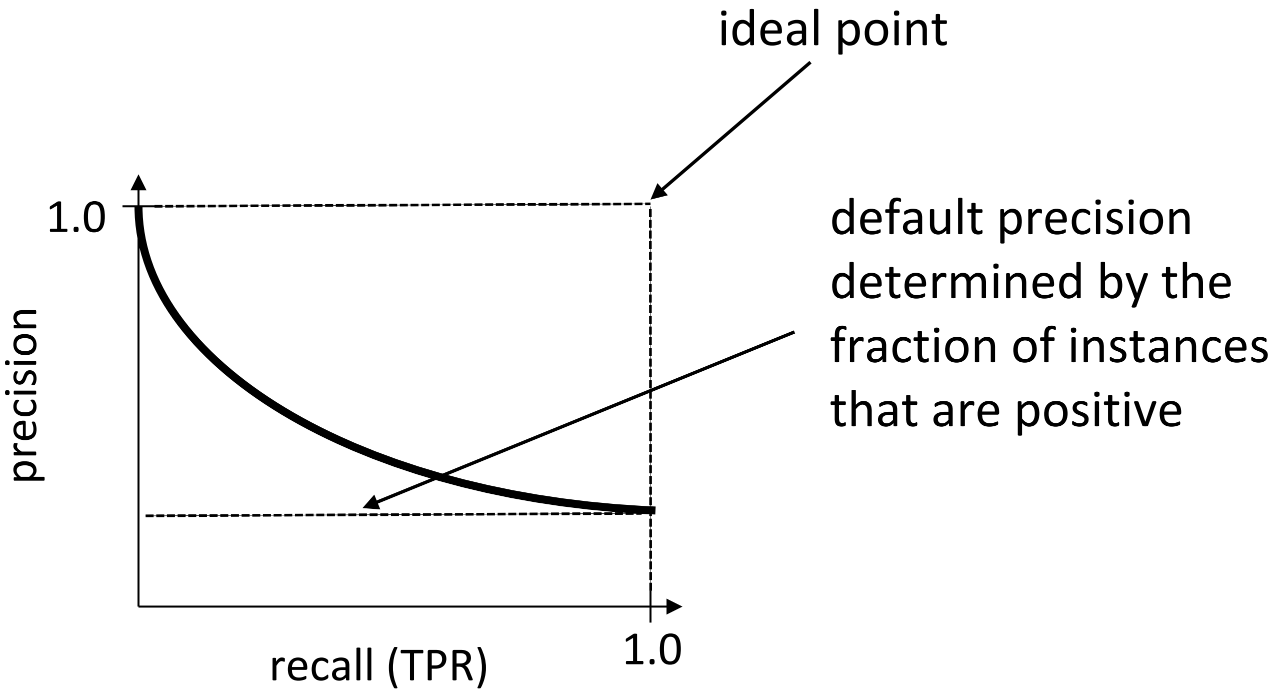
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

$$\text{precision (positive predictive value)} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

Other Metrics: Precision/Recall Curve

- A *precision/recall curve*: plots the precision and recall as the thresholding value is varied.



ideal point

default precision determined by the fraction of instances that are positive

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of confidence
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve* (ideally 1 for both curves)

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- Can identify optimal classification thresholds for tasks with differential misclassification costs

Precision/recall curves

- Well suited for tasks with many negative instances
- We can plot the F1 score,

$$F_1 = \frac{\text{precision} \times \text{recall}}{(\text{precision} + \text{recall})/2}$$

Break & Quiz

Q3-2: Which two extreme points show the best performance and the worst performance respectively on the ROC curve?

1. (1, 1), (0, 0)
2. (0, 1), (1, 0)
3. (1, 0), (0, 1)
4. (0, 1), (1, 1)

Q3-2: Which two extreme points show the best performance and the worst performance respectively on the ROC curve?

1. (1, 1), (0, 0)
2. [✓ (0, 1), (1, 0)]
3. (1, 0), (0, 1)
4. (0, 1), (1, 1)

A ROC curve plots the TP-rate vs. the FP-rate, so usually the x-axis is for FP-rate and y-axis is for TP.-rate When TP-rate = 1 and FP-rate = 0, all instances are correctly classified thus achieving the best result. When TP-rate = 0 and FP-rate = 1, all instances are wrongly classified thus achieving the worst result.

Outline

- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall
- **Linear Regression**
  - Setup, normal equations, GD-based solution
- **Logistic Regression**
  - Linear classification, maximum likelihood estimation, setup, comparisons

Linear Regression: Setup

- **Training**: Given a dataset, where $x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}$

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

  - We will assume, $x_1^{(i)} = 1$ for all $i \in \{1, \ldots, m\}$

- Find $f_\theta(x) = \theta^T x = \sum_{i=1}^{d} \theta_i x_i$ which minimizes

Loss function

$$\ell(f_\theta) = \frac{1}{n} \sum_{j=1}^{n} (f_\theta(x^{(j)}) - y^{(j)})^2$$

Hypothesis Class

Linear Regression: Notation

- **Matrix notation**: set *X* to be the matrix whose *j*th row is $(x^{(j)})^T$
  - And *y* to be the vector $[y^{(1)}, \ldots, y^{(n)}]^T$

- Can re-write the loss function as

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 = \frac{1}{n}\|X\theta - y\|_2^2$$

Linear Regression: Optimizing for θ

- **Set gradient to 0 w.r.t. the weight,**

$$\nabla \ell(f_\theta) = \nabla \frac{1}{n} \|X\theta - y\|_2^2 = 0$$

$$\implies \nabla[(X\theta - y)^T (X\theta - y)] = 0$$

$$\implies \nabla[\theta^T X^T X\theta - 2\theta^T X^T y + y^T y] = 0$$

$$\implies 2X^T X\theta - 2X^T y = 0$$

$$\implies \theta = (X^T X)^{-1} X^T y$$

Regularizing: Ridge Regression

- Same setup, new loss:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

Regularization parameter

  - Conveniently, still have a closed form solution

$$\theta = (X^T X + \lambda n I)^{-1} X^T y$$

  - **Goal**: Prevent large weights

Regularizing: Lasso

- Another type of regularization:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_1$$

Regularization parameter

  - **Goal**: encourage sparse coefficients.

  - No closed form solutions, but efficient solutions via convex optimization.

Evaluation: Metrics

- MSE/RMSE (mean-square error + root version)
- MAE (mean average error)
- R-squared (more on this next)
- Usually, compute on training data… (but should do cross validation!)
  - Fixed-design LR
  - Random-design LR

R-squared

- Several ways to define it, one way:

$$R^2 = 1 - \frac{\sum_j (y^{(j)} - f_\theta(x^{(j)}))^2}{\sum_j (y^{(j)} - \bar{y})^2}$$

Empirical mean of labels

- Intuition: how much of the variance in y is predictable by x

Iterative Methods: Gradient Descent

- What if there's no closed-form solution?
- Use an iterative approach. Goal: get closer to solution.

- Gradient descent.
  - Suppose we're computing $\min_{\theta} g(\theta)$
  - Start at some θ_0
  - Iteratively compute $\theta_{t+1} = \theta_t - \alpha \nabla g(\theta_t)$
  - Stop after some # of steps

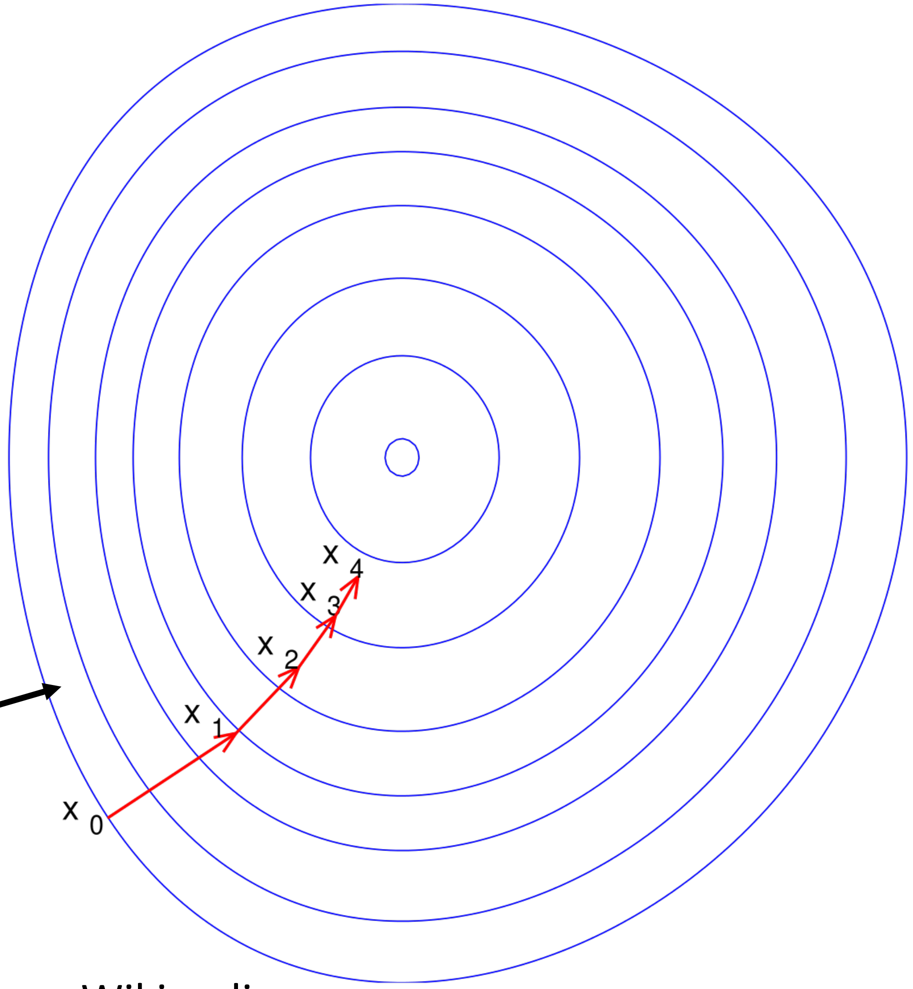
Learning rate/step size

Gradient Descent: Illustration

- **Goal**: steps get closer to minimizer

- Some notes:
  - Step size can be fixed or a function

  - Under certain conditions, will converge to global minimum
    - Need **convexity** for this



Level Sets

Wikipedia

Gradient Descent: Linear Regression

- Back to our linear regression problem.

- Want to find $\min_\theta \ell(f_\theta) = \min_\theta \frac{1}{n}\|X\theta - y\|_2^2$
  - What is our gradient? $\nabla \ell(f_\theta) = \frac{1}{n}(2X^T X\theta - 2X^T y)$
  - So, plugging in , we get

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{n}(2X^T X\theta_t - 2X^T y)$$

Linear Regression: Normal Equations vs GD

- Let us compare **computation costs**.
- Normal Equations
  - Check dimensions

$$\theta = (\underbrace{X^T X}_{d \times d})^{-1} \underset{\substack{\uparrow \\ d \times n}}{X^T} \underset{\substack{\uparrow \\ n \times 1}}{y}$$

  - Cost: (i) invert matrix, Θ(d^3). (ii) multiplication, Θ(d^2n).
  - **Total**: Θ(d^2n+ d^3).

Recall: by standard methods, inverting a square *m* x *m* matrix is Θ(m^3).

Multiplying a *m* x *p* with a *p* x *q* matrix is Θ(*mpq*)

Linear Regression: Normal Equations vs GD

- Let us compare **computation costs**.
- Normal Equations $\theta = (X^T X)^{-1} X^T y$
  - **Total Cost**: $\Theta(d^2 n + d^3)$.

- Gradient Descent: t iterations

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{n}(2X^T X\theta_t - 2X^T y)$$

  - Cost: $\Theta(dn)$ at each step.
  - **Total Cost**: $\Theta(dnt)$.

If we do “few” steps t, then **GD is cheaper**: $t<\max\{d, d^2/n\}$

Gradient Descent: Convergence

- Even if GD is cheaper, what does it give us?
- Let us analyze it. We'll need some ingredients
  - *Convex function g*
  - Differentiable (need this for gradients)
  - Lipschitz-continuous gradients

$$\|\nabla g(x_1) - \nabla g(x_2)\|_2 \leq L\|x_1 - x_2\|_2$$

  - If we run *t* steps with fixed step size, starting at x_0

$$g(x_t) - g(x^*) \leq \frac{\|x_0 - x^*\|_2^2}{2t\alpha}$$

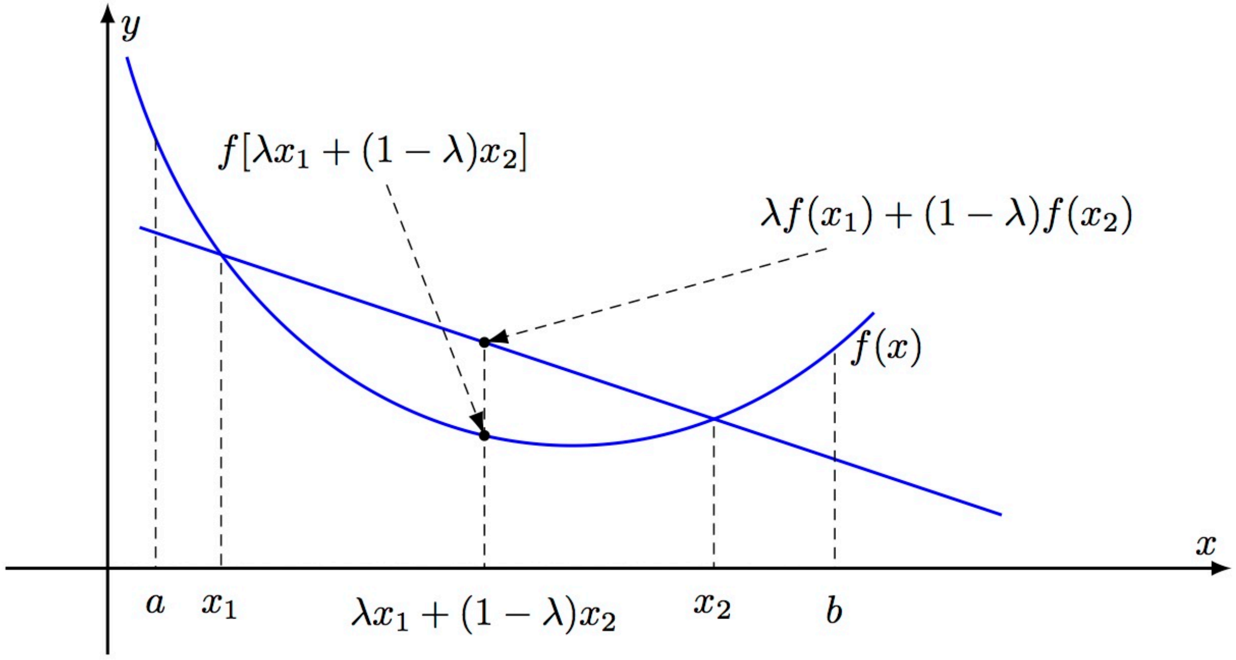
Minimizer

Proof: next time!

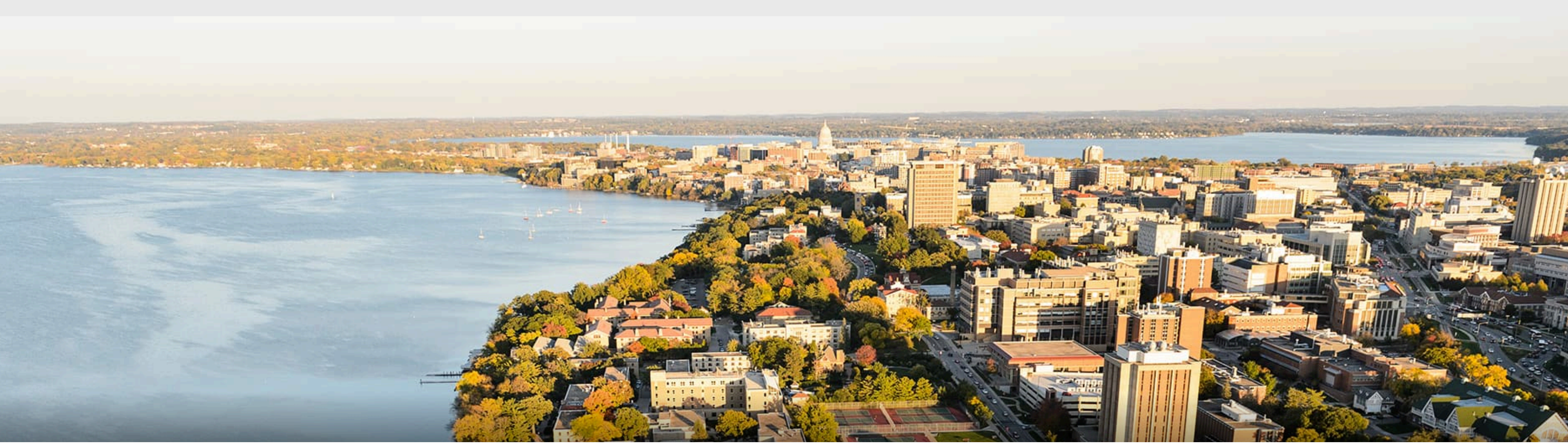
Gradient Descent Analysis : Convexity

- Recall the definition of a convex function. For f, with convex domain, for all x_1, x_2 in this domain and all $\lambda \in [0, 1]$

$$f(\underbrace{\lambda x_1 + (1 - \lambda)}_{\textbf{Convex combination}} x_2) \leq \underbrace{\lambda f(x_1) + (1 - \lambda) f(x_2)}_{\textbf{Line segment} \text{ joining } f(x_1) \text{ and } f(x_2)}$$



y
$f[\lambda x_1 + (1 - \lambda)x_2]$
$\lambda f(x_1) + (1 - \lambda)f(x_2)$
$f(x)$
x
a
x_1
$\lambda x_1 + (1 - \lambda)x_2$
x_2
b



Break & Quiz

Q: Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*

B. *Start introducing polynomial degree variables*

C. *Use L1 regularization*

D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. A, B
4. A, B, C, D

Q: Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*

B. *Start introducing polynomial degree variables*

C. *Use L1 regularization*

D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. [✓ A, B]
4. A, B, C, D

In case of under fitting, you need to induce more variables in variable space or you can add some polynomial degree variables to make the model more complex to be able to fit the data better. No regularization methods should be used because regularization is used in case of overfitting.

Q: How do you choose the regularization paramater λ in ridge/lasso regression?

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_1$$

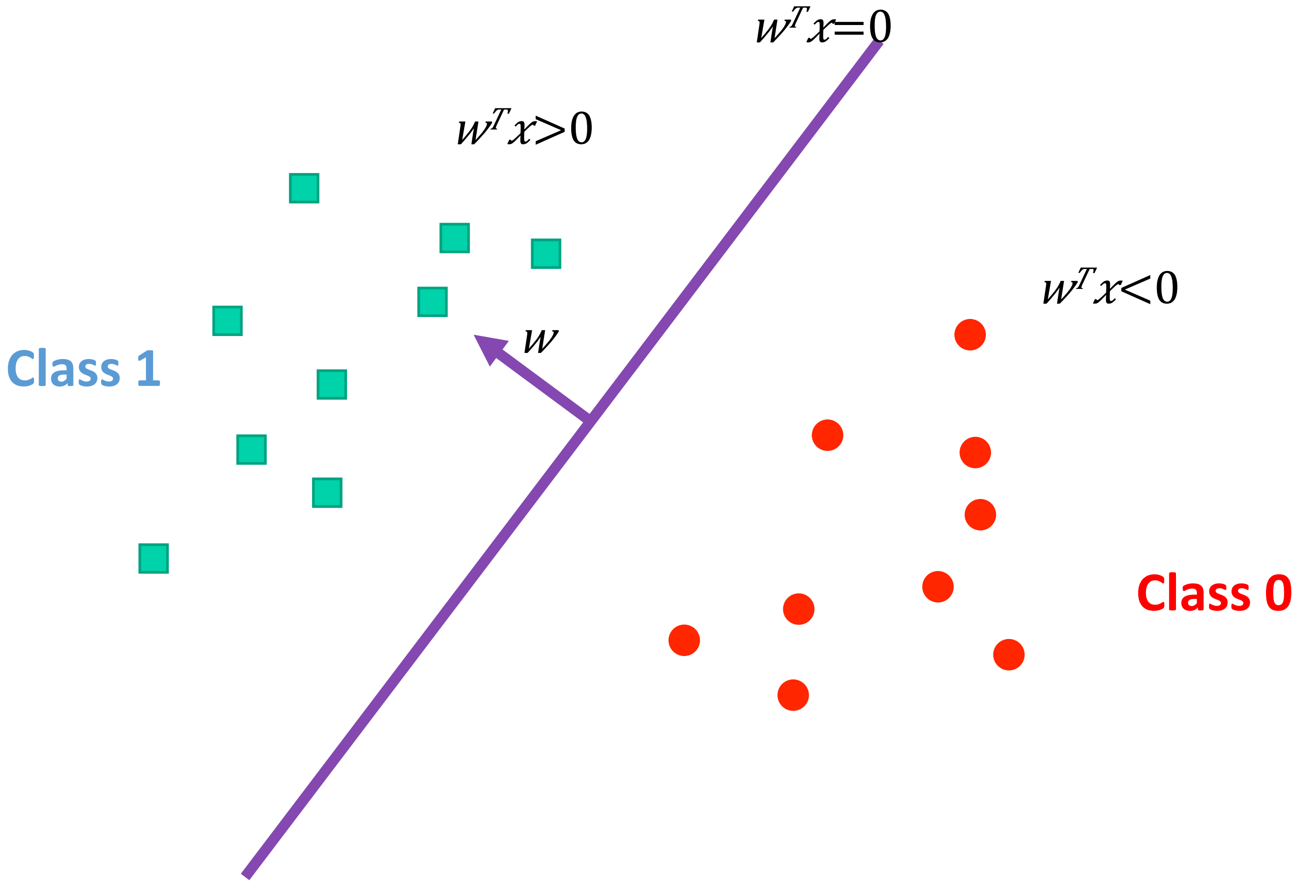
Ans: tuning (validation) set, cross validation etc.

Outline

- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall
- **Linear Regression**
  - Setup, normal equations, GD-based solution
- **Logistic Regression**
  - Linear classification, maximum likelihood estimation, setup, comparisons

Classification: Linear

- We’ve been talking about regression. What about classification with linear models?



Linear Classification: Attempt 1

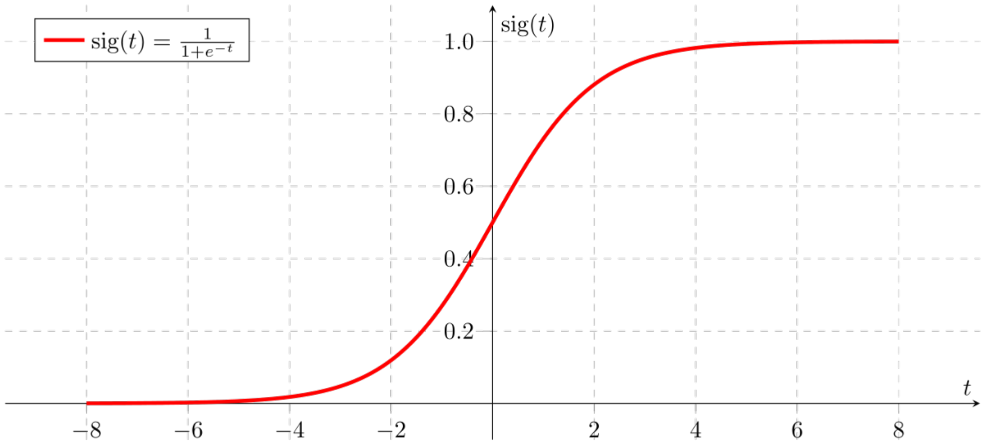
- Hyperplane: solutions to $\theta^T x = c$
  - note: d-1 dimensional
- So… try to use such hyperplanes as separators?
  - Model: $f_\theta(x) = \theta^T x$
  - Predict: y=1 if $\theta^T x > 0$, y=0 otherwise?
  - Ie, $y = \text{step}(f_\theta(x))$
  - Train: 0/1 loss, or, $\ell(f_\theta) = \frac{1}{m}\sum_{i=1}^{m} 1\{\text{step}(f_\theta(x^{(i)}) \neq y^{(i)})$

Difficult to optimize!!

Linear Classification: Attempt 2

- Let us think probabilistically. Learn $P_\theta(y|x)$ instead

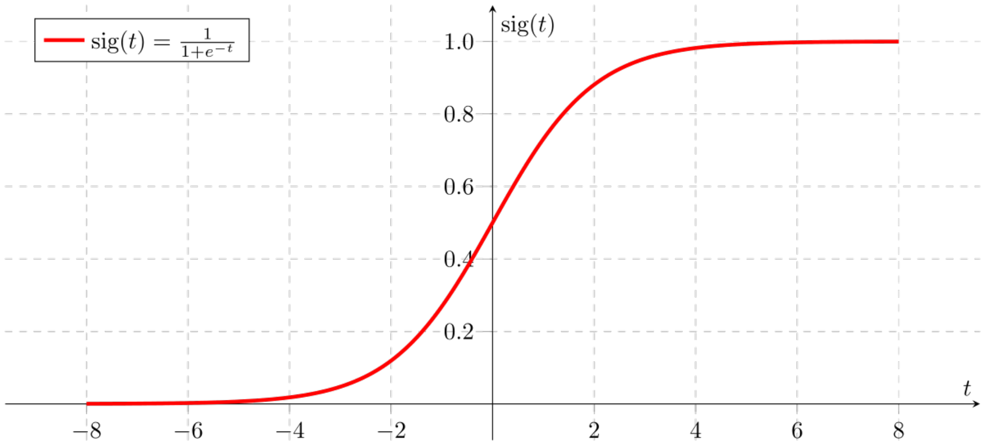
- How?
  - Specify the conditional distribution $P_\theta(y|x)$
  - Use maximum likelihood estimation (MLE) to derive a loss
  - Run gradient descent (or related optimization algorithm)



Linear Classification: Attempt 2

- Let us think probabilistically. Learn $P_\theta(y|x)$ instead

- How?
  - Specify **the conditional distribution** $P_\theta(y|x)$
  - Use **maximum likelihood estimation (MLE)** to derive a loss
  - Run gradient descent (or related optimization algorithm)



Likelihood Function

- Captures the probability of seeing some data as a function of model parameters:

$$\mathcal{L}(\theta; X) = P_\theta(X)$$

  - If data is iid, we have $\mathcal{L}(\theta; X) = \prod_j p_\theta(x_j)$

- Often more convenient to work with the log likelihood
  - Log is a monotonic + strictly increasing function

Maximum Likelihood

- For some set of data, find the parameters that maximize the likelihood / log-likelihood

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta; X)$$

- Example: suppose we have n samples from a Bernoulli distribution

$$P_\theta(X = x) = \begin{cases} \theta & x = 1 \\ 1 - \theta & x = 0 \end{cases}$$

Then,

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1 - \theta)^{n-k}$$

Maximum Likelihood: Example

- Want to maximize likelihood w.r.t. Θ

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1-\theta)^{n-k}$$

- Differentiate (use product rule) and set to 0. Get

$$\theta^{h-1}(1-\theta)^{n-h-1}(h - n\theta) = 0$$

- So: ML estimate is $\hat{\theta} = \dfrac{h}{n}$

h = |{x_i | x_i = 1}|

ML: Conditional Likelihood

- Similar idea, but now using conditional probabilities:

$$\mathcal{L}(\theta; Y, X) = p_\theta(Y|X)$$

  - If data is iid, we have

$$\mathcal{L}(\theta; Y, X) = \prod_j p_\theta(y_j|x_j)$$

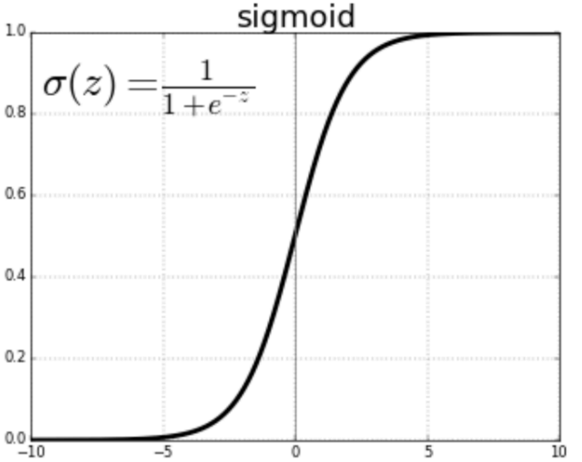
  - Now we can apply this to linear classification: yields **logistics regression.**

Logistic Regression: Conditional Distribution

- Notation: $$\sigma(z) = \frac{1}{1+\exp(-z)} = \frac{\exp(z)}{1+\exp(z)}$$

↑

Sigmoid



- **Conditional Distribution**:

$$P_\theta(y=1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

Logistic Regression: Loss

- Conditional MLE:

$$\log\ \text{likelihood}(w|x^{(i)}, y^{(i)}) = \log P_\theta(y^{(i)}|x^{(i)})$$

- So: $$\min_\theta \ell(f_\theta) = \min_\theta -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

Or,

$$\min_\theta -\frac{1}{n}\sum_{y^{(i)}=1} \log \sigma(\theta^T x^{(i)}) - \frac{1}{n}\sum_{y^{(i)}=0} \log(1-\sigma(\theta^T x^{(i)}))$$

Logistic Regression: Sigmoid Properties

- **Bounded**:

$$\sigma(z) = \frac{1}{1+\exp(-z)} \in (0,1)$$

- **Symmetric**:

$$1-\sigma(z) = \frac{\exp(-z)}{1+\exp(-z)} = \frac{1}{\exp(z)+1} = \sigma(-z)$$

- **Gradient**:

$$\sigma'(z) = \frac{\exp(-z)}{(1+\exp(-z))^2} = \sigma(z)(1-(\sigma(z))$$

Logistic regression: Summary

- **Logistic regression = sigmoid conditional distribution + MLE**

- More precisely:
  - Give training data iid from some distribution D,
  - **Train**:

$$\min_{\theta} \ell(f_\theta) = \min_{\theta} -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

  - **Test**: output label probabilities

$$P_\theta(y=1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

Logistic Regression: Comparisons

- Recall the first attempt:

$$\ell(f_\theta) = \frac{1}{m}\sum_{i=1}^{m} 1\{\text{step}(f_\theta(x^{(i)}) \neq y^{(i)})$$

Difficult to optimize!!

Logistic Regression: Comparisons

- What if we run least squares linear regression?

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2$$

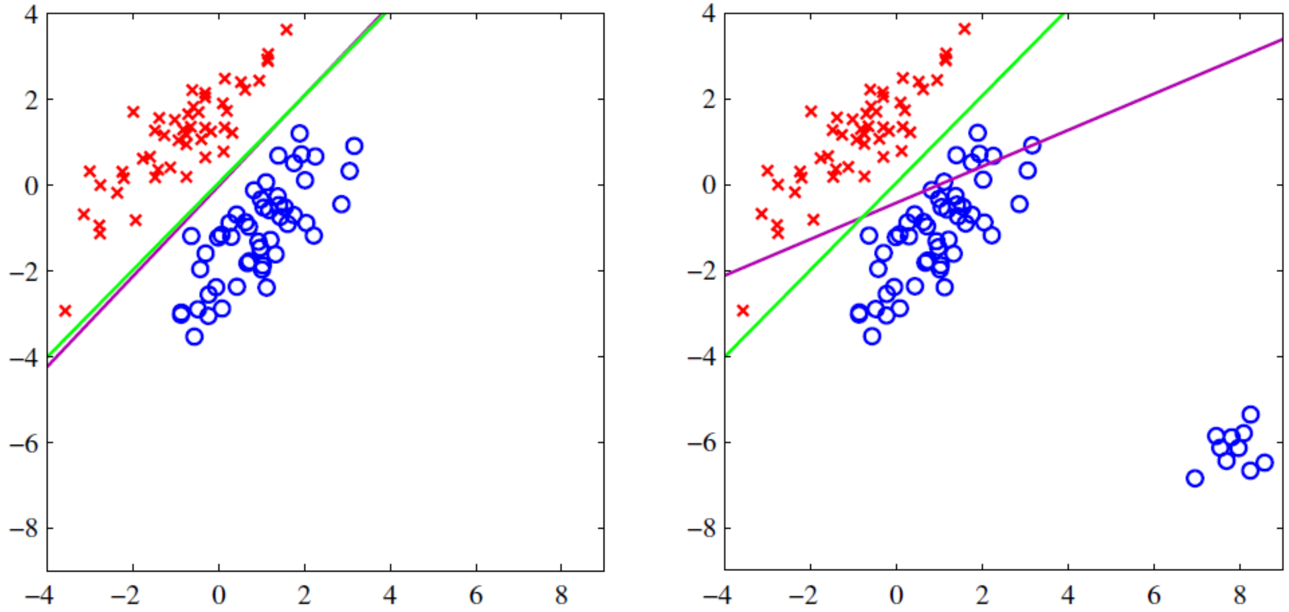


Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Figure: *Pattern Recognition and Machine Learning*, Bishop

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, and Fred Sala