DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

September 27 2018

Logistics

- HW1 due tomorrow, Friday, Oct 28, at 11:59pm
- Midterm exam has been scheduled
  – Oct 16 during class
- Project proposal: due Oct 22
  – 1 page description of problem you will solve, dataset, and ML algorithms
  – Individual project
  – Project template and potential ideas will be shared soon
- Project milestone: due Nov 13
  – 2 page description on progress
- Project report at the end of semester and project presentations in class (10 minute per project)

Review

- Classification is a supervised learning problem
  - Prediction is binary or multi-class
- Classification techniques
  - Linear classifiers (perceptron): compact, fast to evaluate
    - Can run in online or batch mode
  - Instance learners (kNN): need to store entire training data, fast to evaluate
- Cross-validation should be used for parameter selection and estimation of model error
  - Improves model generalization

Supervised learning

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$

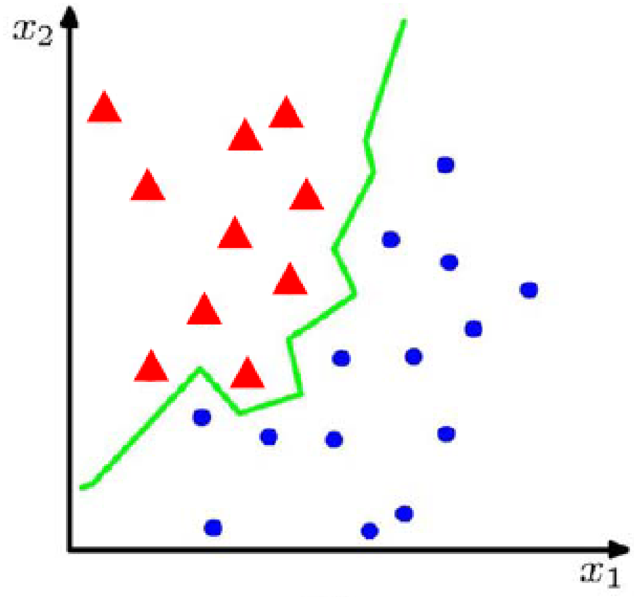
Input: Training examples of unknown target function f

$$\{x^{(i)}, y^{(i)}\}, \text{for } i = 1, \ldots, n$$

Output: Hypothesis $\hat{f} \in H$ that best approximates f

$$\hat{f}\left(x^{(i)}\right) \approx y^{(i)}$$

Classification



Binary or discrete

- Suppose we are given a training set of N observations

$$\{x^{(1)}, \dots, x^{(n)}\} \text{ and } \{y^{(1)}, \dots, y^{(n)}\}, x^{(i)} \in R^d, y^{(i)} \in \{-1, 1\}$$

- Classification problem is to estimate f(x) from this data such that

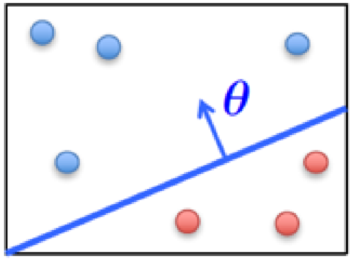
$$f\left(x^{(i)}\right) = y^{(i)}$$

Online Perceptron

Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
  Receive training example $(\boldsymbol{x}^{(i)}, y^{(i)})$
  if $y^{(i)}\theta^T x^{(i)} \leq 0$ // prediction is incorrect
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)}\boldsymbol{x}^{(i)}$

Online learning – the learning mode where the model update is performed each time a single observation is received

Batch learning – the learning mode where the model update is performed after observing the entire training set

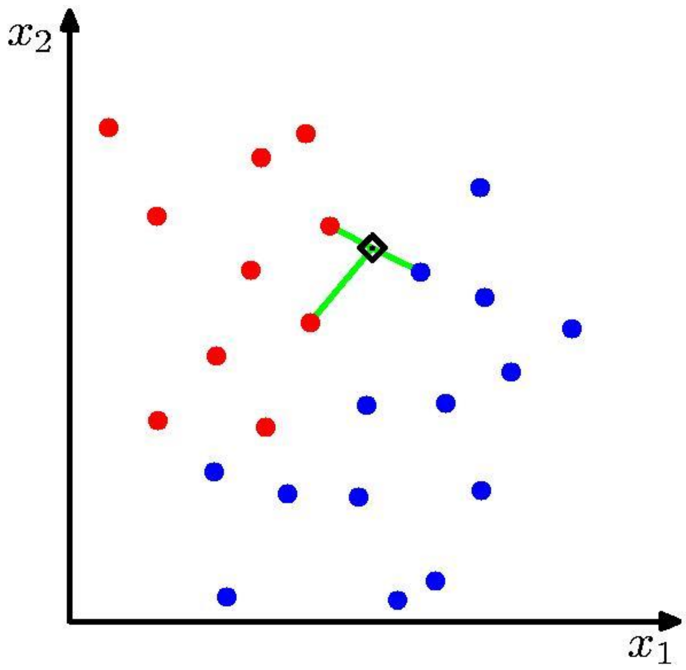


$$h(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x})$$

$$\boldsymbol{\theta}^\intercal \boldsymbol{x} > 0 \implies y = +1$$
$$\boldsymbol{\theta}^\intercal \boldsymbol{x} < 0 \implies y = -1$$

Linear classifier

kNN



- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

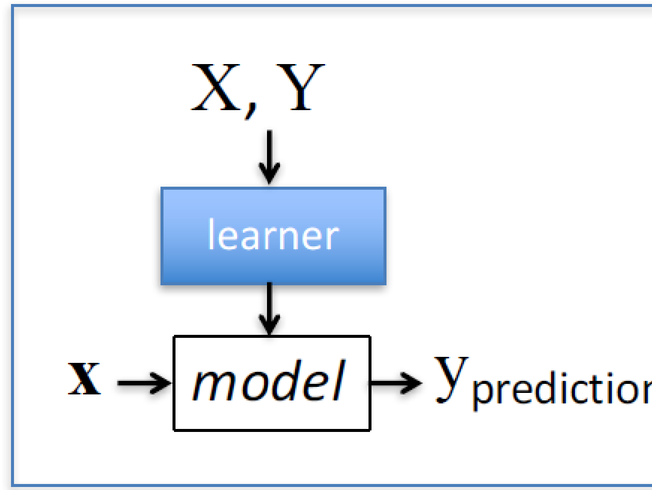
Outline

- Evaluation of classification algorithms
  - Metrics: accuracy, precision, recall
- Cross validation
  - K-fold CV or LOOCV
- Logistic regression
  - Maximum Likelihood Estimation (MLE) of model parameters
- Gradient descent for logistic regression
  - Cross-entropy loss

Evaluation of classifiers

Given: labeled training data $X, Y = \{ x^{(i)}, y^{(i)} \}_{i=1}^{n}$

- Assumes each $x^{(i)} \sim \mathcal{D}(\mathcal{X})$



Train the model:

model ← *classifier*.train(X, Y)

Apply the model to new data:

- Given: new unlabeled instance $\boldsymbol{x} \sim \mathcal{D}(\mathcal{X})$

$y_{\text{prediction}}$ ← *model*.predict($\mathbf{x}$)

Classification Metrics

$$\text{accuracy} = \frac{\#\text{ correct predictions}}{\#\text{ test instances}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\#\text{ incorrect predictions}}{\#\text{ test instances}}$$

Confusion Matrix

- Given a dataset of P positive instances and N negative instances:

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

- Imagine using classifier to identify positive cases (i.e., for information retrieval)

$$\text{precision} = \frac{TP}{TP + FP}$$

Probability that classifier predicts positive correctly

$$\text{recall} = \frac{TP}{TP + FN}$$

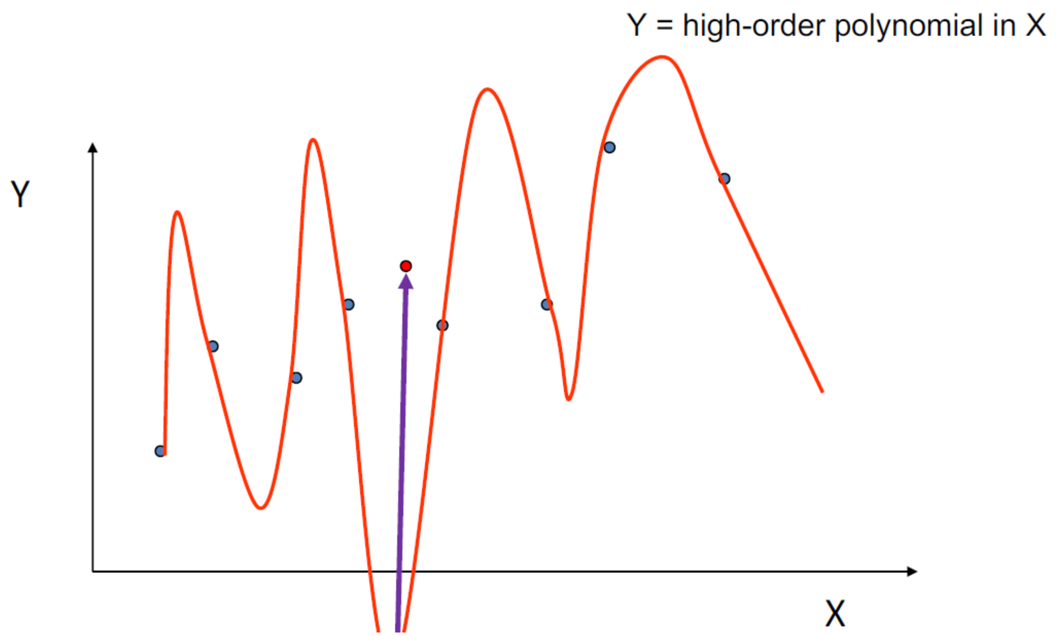
Probability that actual class is predicted correctly

Goals of classification

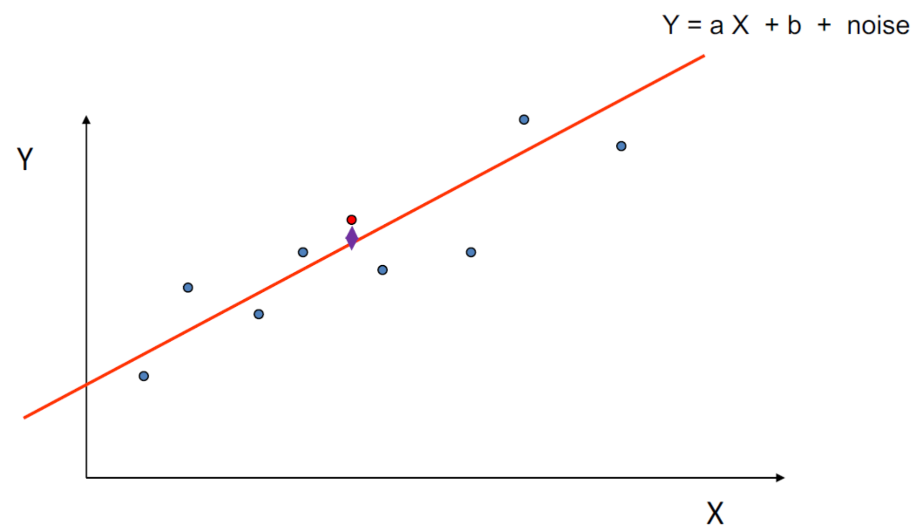
- Produce models with high accuracy / low error
- Generalize well
  - Avoid overfitting (perform well on training set, but poorly on testing data)
- Find the simplest model that produces reasonable accuracy
  - Occam's Razor
- Reduce both bias and variance!

Overfitting

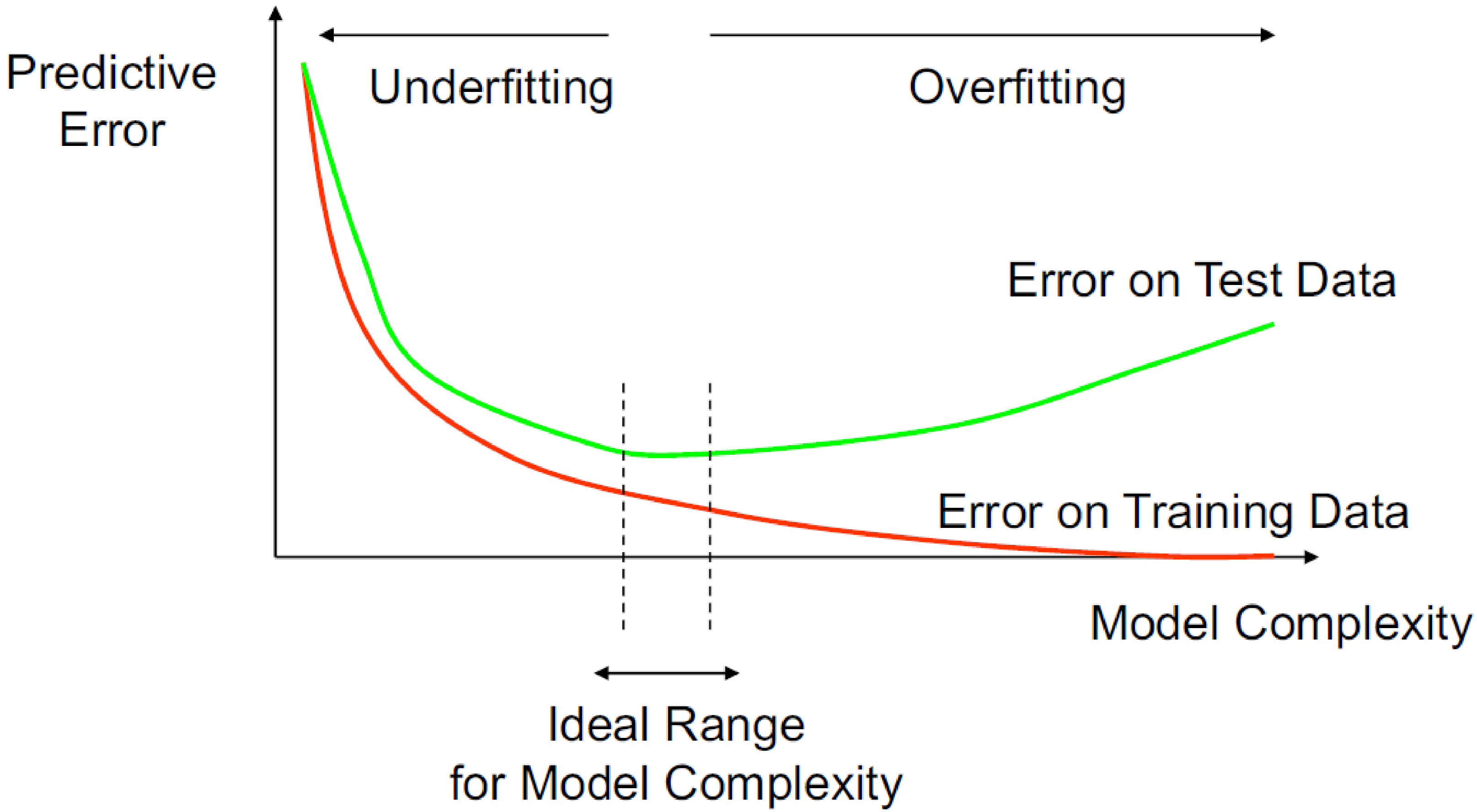
Complex model



The true model



How Overfitting Affects Prediction



How can we avoid over-fitting without having access to testing data?

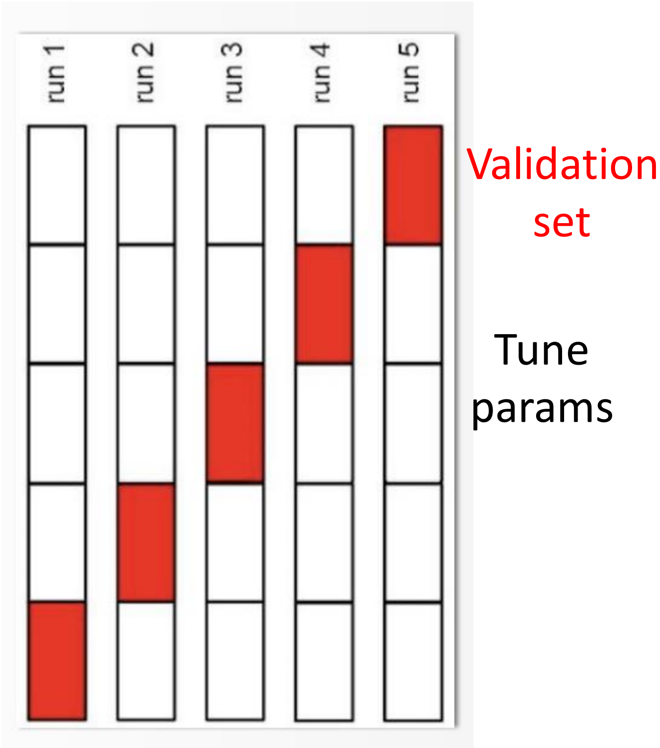
Cross Validation

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Test set
  - Randomly split training set into training and validation, e.g., 66% - 33%

- Features: attribute-value pairs which characterize each x

- Experimentation cycle
  - Select a hypothesis f
    (Tune hyperparameters on held-out or *validation* set)
  - Estimate and reduce average error during multiple runs by randomly choosing validation set
  - Compute final error on testing set

- Evaluation
  - Accuracy: fraction of instances predicted correctly
  - Use other metrics as appropriate (precision, recall)



Test Data

- Improves model generalization
- Avoids overfitting

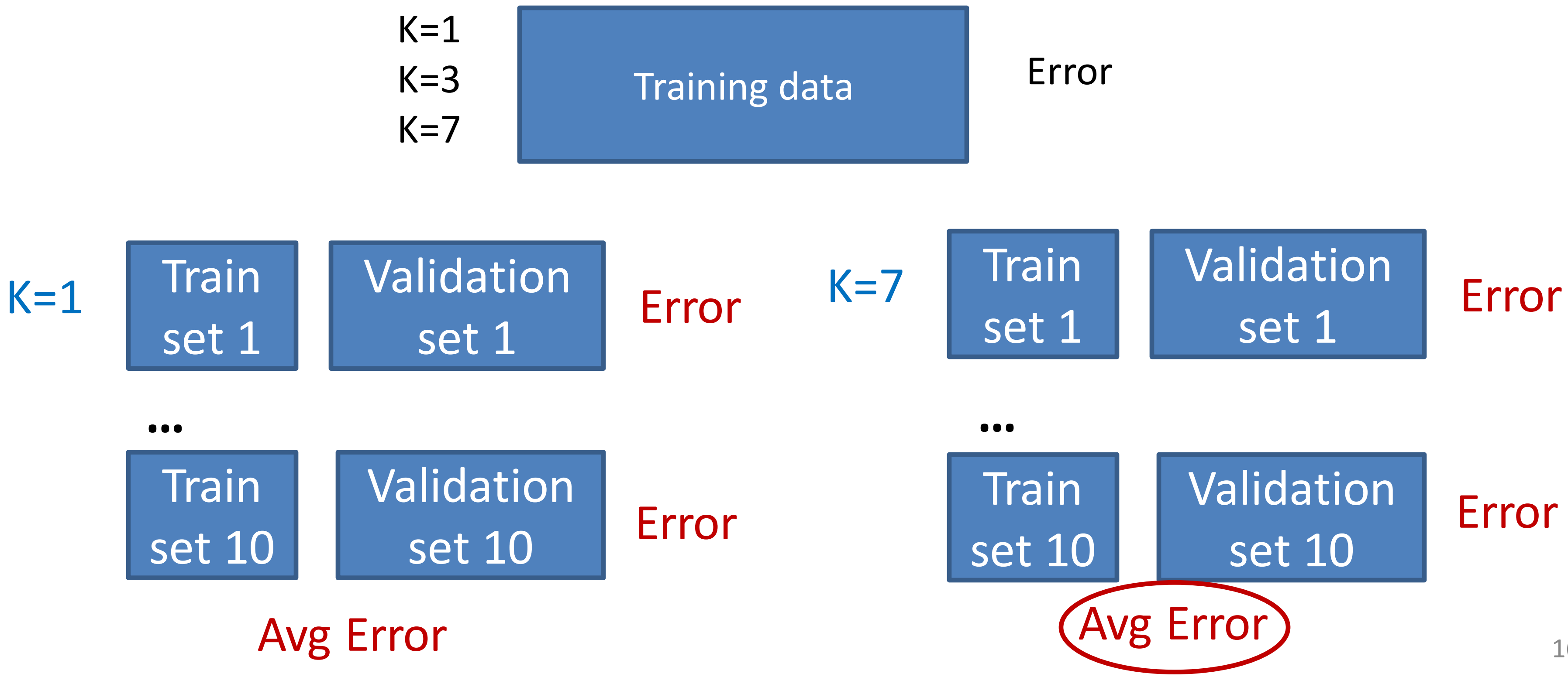
Cross-validation for kNN

As K increases:

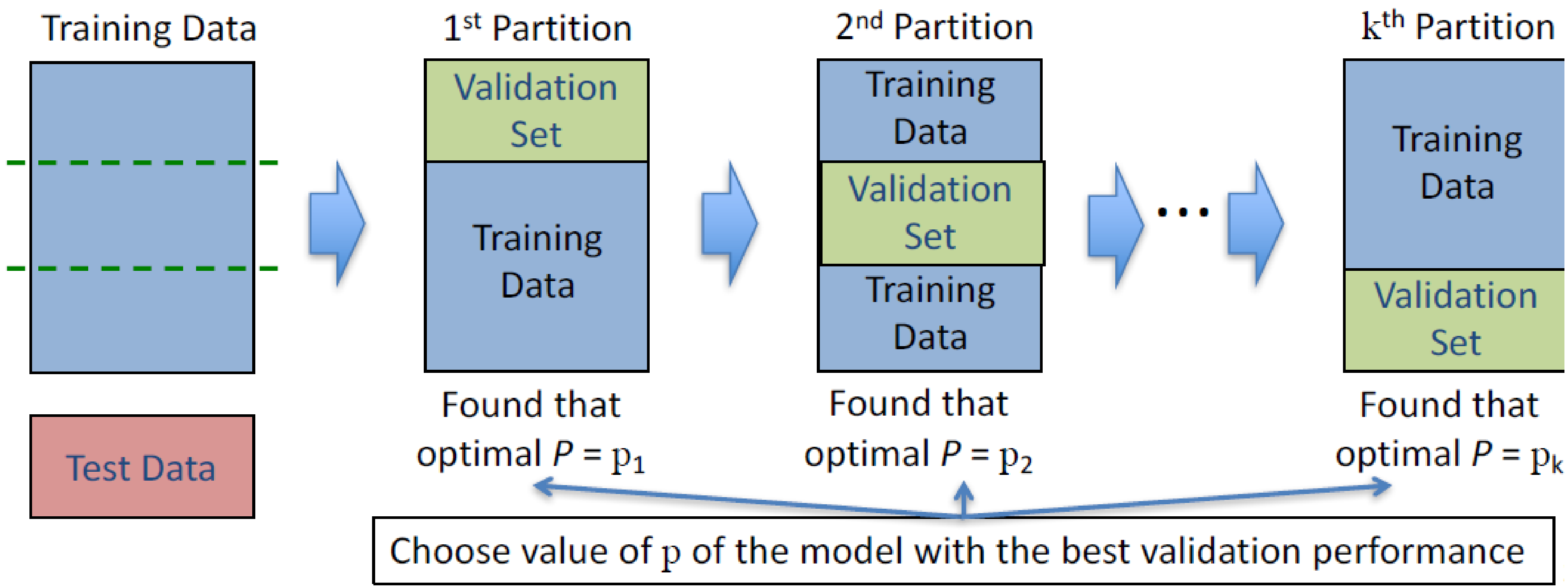
- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation

- Split training data into training and validation
- Hold out validation data and measure error on this



Cross Validation



- k-fold CV
  - Split data into k partitions of equal size
- Leave-one-out CV (LOOCV)
  - k=n (validation set only one point)

Outline

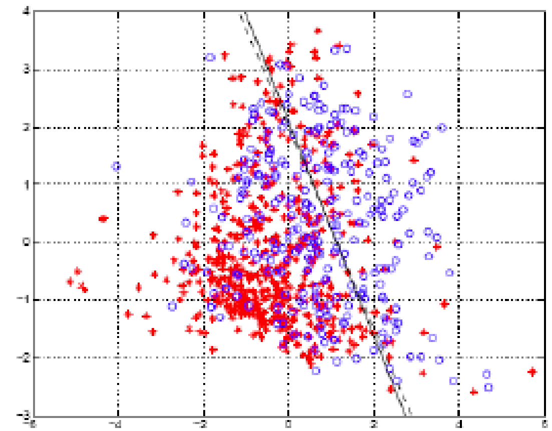
- Evaluation of classification algorithms
  - Metrics: accuracy, precision, recall
- Cross validation
  - K-fold CV or LOOCV
- Logistic regression
  - Maximum Likelihood Estimation (MLE) of model parameters
- Gradient descent for logistic regression
  - Cross-entropy loss

Classification based on Probability

- Instead of just predicting the class, give the probability of the instance being that class
  - i.e., learn $p(y \mid \boldsymbol{x})$
- Comparison to perceptron:
  - Perceptron doesn't produce probability estimate
- Recall that:

$$0 \leq p(\text{event}) \leq 1$$

$$p(\text{event}) + p(\neg\text{event}) = 1$$



Example

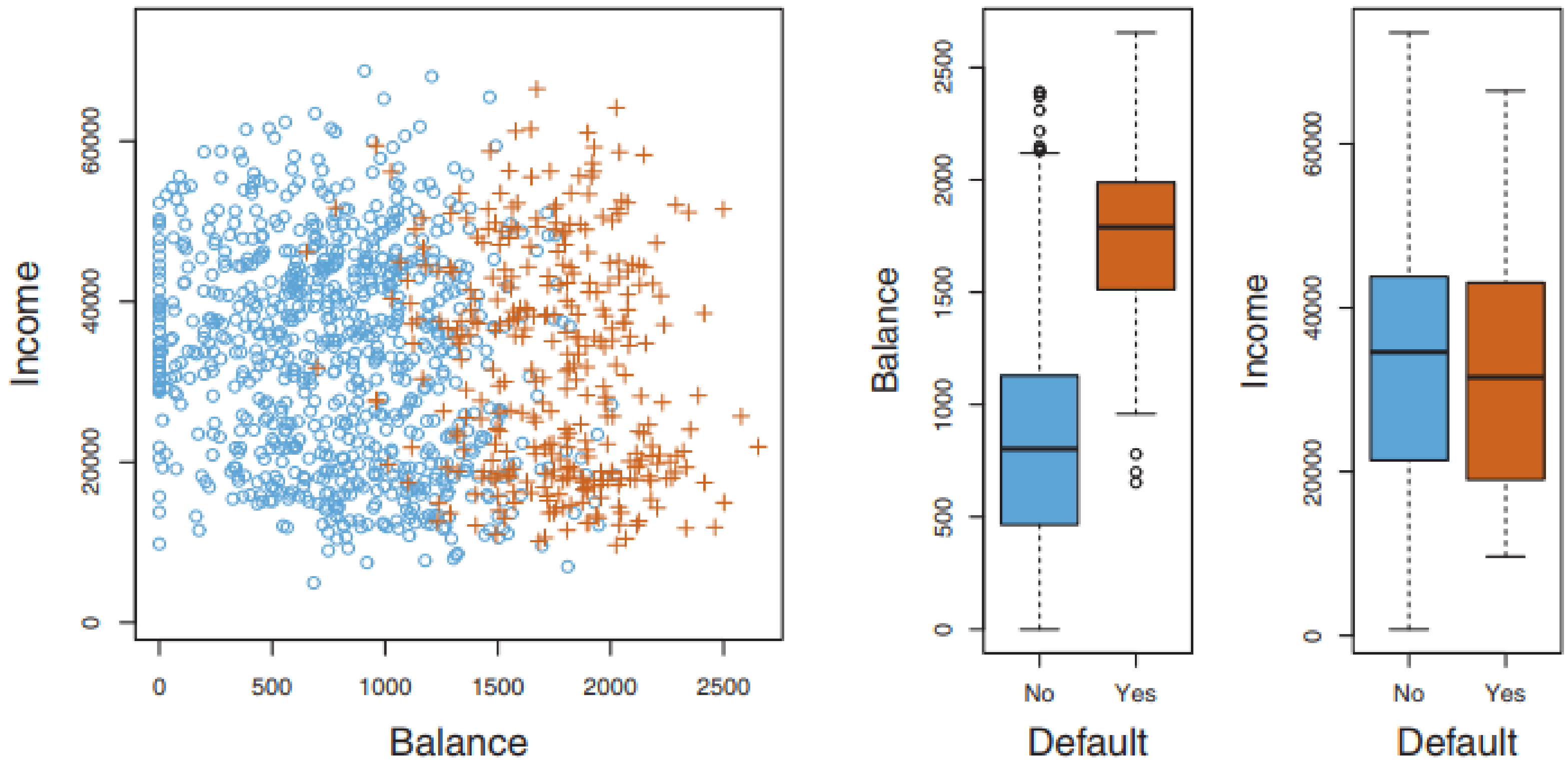


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of* `default` *status.* Right: *Boxplots of* `income` *as a function of* `default` *status.*

Why not linear regression?

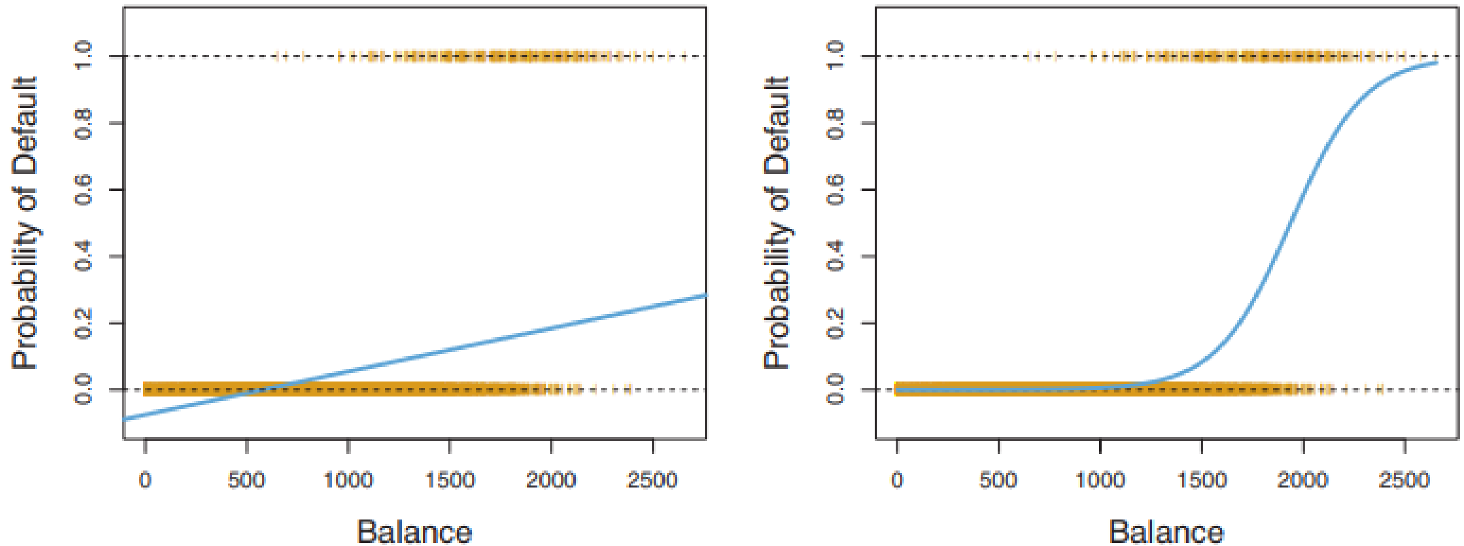


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default`*(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Logistic regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

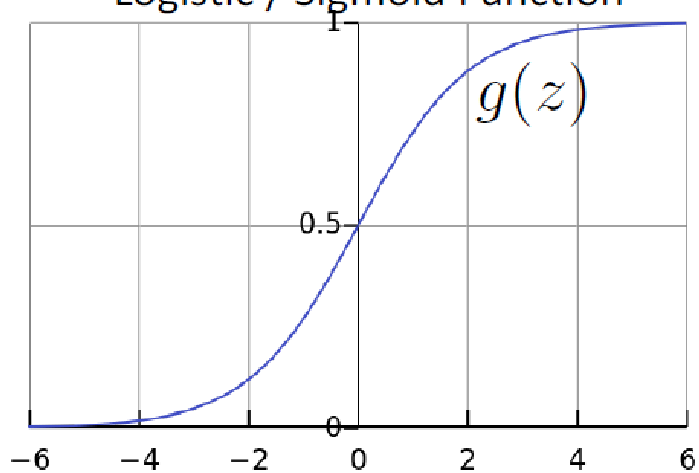
Can't just use linear regression with a threshold

- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}}$$



Interpretation of Model Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

LR is a Linear Classifier!

- Predict $y = 1$ if:

$$\mathrm{P}[y = 1|x;\theta] > \mathrm{P}[y = 0|x;\theta]$$

$$\mathrm{P}[y = 1|x;\theta] > ½$$

$$\frac{1}{1 + e^{-\theta^T x}} > \frac{1}{2}$$

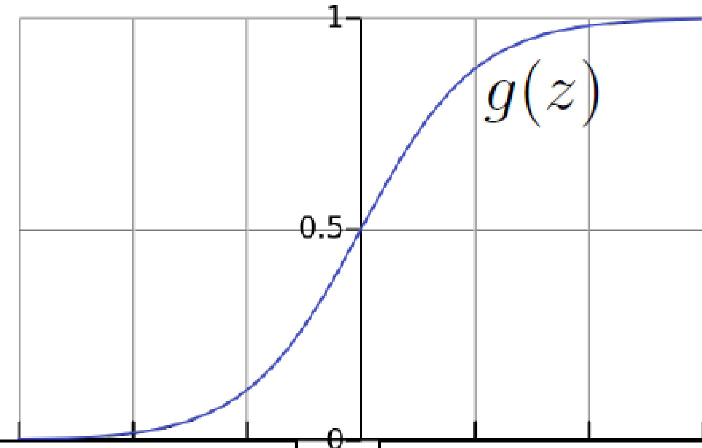
- Equivalent to:
  - $e^{\theta_0 + \sum_{i=1}^{d} \theta_j x_j} > 1$
  - $\theta_0 + \sum_{i=1}^{d} \theta_j x_j > 0$

Logistic Regression is a linear classifier!

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

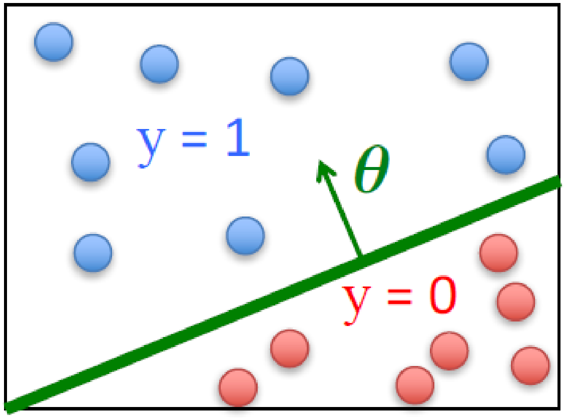
$$g(z) = \frac{1}{1+e^{-z}}$$



$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Logistic Regression is a linear classifier!

Logistic Regression

- Given $\left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)} \right), \left(\boldsymbol{x}^{(2)}, y^{(2)} \right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)} \right) \right\}$
  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x}\right)$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^{\mathsf{T}} = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

Logistic Regression Objective

- Can't just use squared loss as in linear regression:

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

Maximum Likelihood Estimation (MLE)

Given training data $X = \{x^{(1)}, \dots, x^{(n)}\}$ with labels $\mathrm{Y} = \{y^{(1)}, \dots, y^{(n)}\}$

What is the likelihood of training data for parameter θ?

Define likelihood function

$$Max_\theta \, L(\theta) = P[Y|X;\theta]$$

Assumption: training points are independent

$$L(\theta) = \prod_{i=1}^{n} P[y^{(i)}|x^{(i)};\theta]$$

Log Likelihood

- Max likelihood is equivalent to maximizing log of likelihood

$$L(\theta) = \prod_{i=1}^{n} P[y^{(i)} | x^{(i)}, \theta]$$

$$\log L(\theta) = \sum_{i=1}^{n} \log P[y^{(i)} | x^{(i)}, \theta]$$

- They both have the same maximum θ_{MLE}

MLE for Logistic Regression

$$p(y|x,\theta) = h_\theta(x)^y \big(1 - h_\theta(x)\big)^{1-y}$$

$$\begin{aligned}\boldsymbol{\theta}_{\text{MLE}} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} y^{(i)} \log h_\theta\big(x^{(i)}\big) + (1 - y^{(i)}) \log\big(1 - h_\theta(x)\big)\end{aligned}$$

- Substitute in model, and take negative to yield

Logistic regression objective:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

Objective for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right) \right]$$

- Cost of a single instance:

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

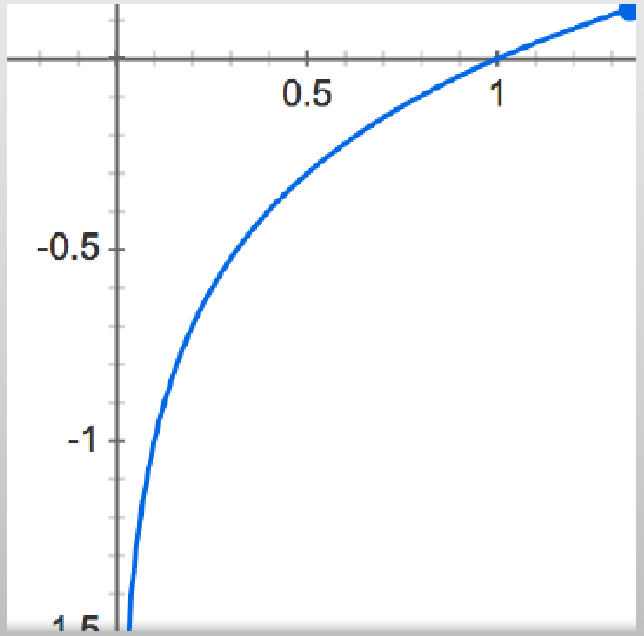
- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \underbrace{\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\right)}_{\text{Cross-entropy loss}}$$

Intuition

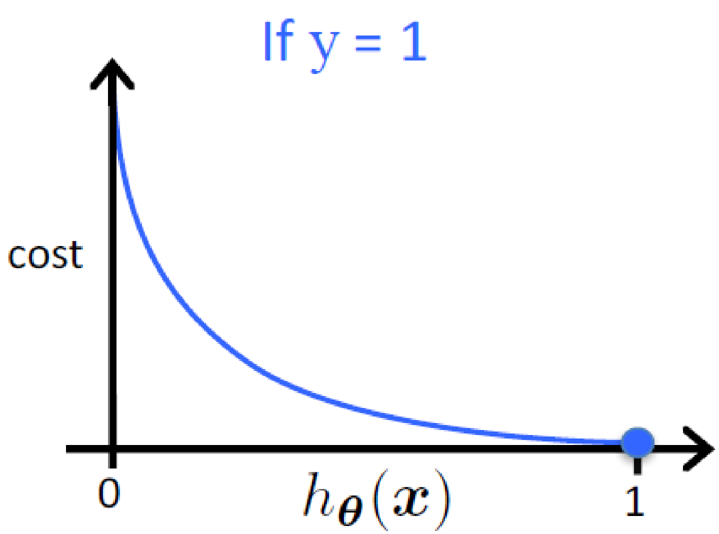
$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



Intuition

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0 \end{cases}$$

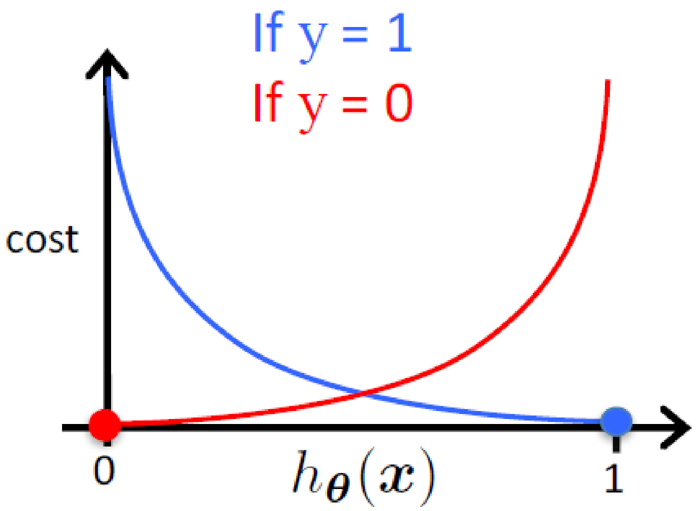


If y = 1

- Cost = 0 if prediction is correct
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0, \text{cost} \to \infty$
- Captures intuition that larger mistakes should get larger penalties
  - e.g., predict $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$, but y = 1

Intuition

$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ \boxed{-\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0} \end{cases}$$



If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \rightarrow 0, \text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties

Gradient Descent for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

Regularized Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- We can regularize logistic regression exactly as before:

$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda\sum_{j=1}^{d}\theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

L2 regularization

Review

- Cross-validation should be used to avoid over-fitting
  - K-fold or LOOCV
- Evaluating fit of a model using different metrics
  - Accuracy, precision, recall
- Logistic regression
  - Estimates $\Pr[Y = 1|X = x]$ using sigmoid
  - Maximum Likelihood Estimation (MLE) for objective
  - Can use gradient descent for training
  - Very interpretable

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!