DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

Fall 2019

Outline

- Classification
- Linear classification
- Perceptron
  - Online and batch perceptron
- LDA
  - Generative models
- Logistic regression
  - Classification based on probability

Supervised learning

Problem Setting

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

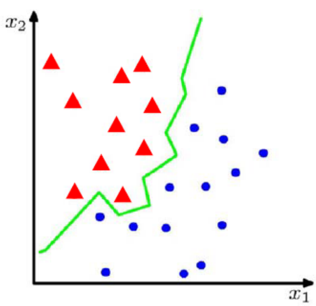
Input: Training examples of unknown target function f

$$\{x_i, y_i\}, \text{for } i = 1, \dots, n$$

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

$$\hat{f}(x_i) \approx y_i$$

Classification



Binary or discrete

- Suppose we are given a training set of N observations

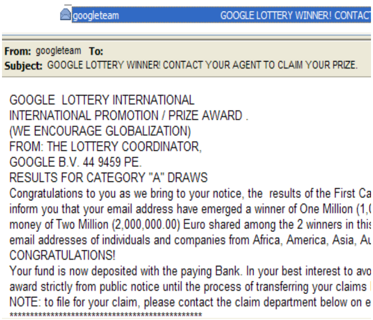
$$\{x_1, \dots, x_N\} \text{ and } \{y_1, \dots, y_N\}, x_i \in R^d, y_i \in \{-1, 1\}$$

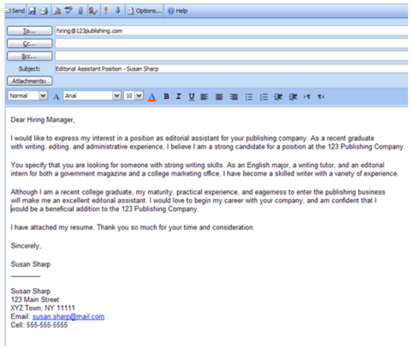
- Classification problem is to estimate f(x) from this data such that

$$f(x_i) = y_i$$

Example 1

Classifying spam email





Content-related features

- Use of certain words
- Word frequencies
- Language
- Sentence

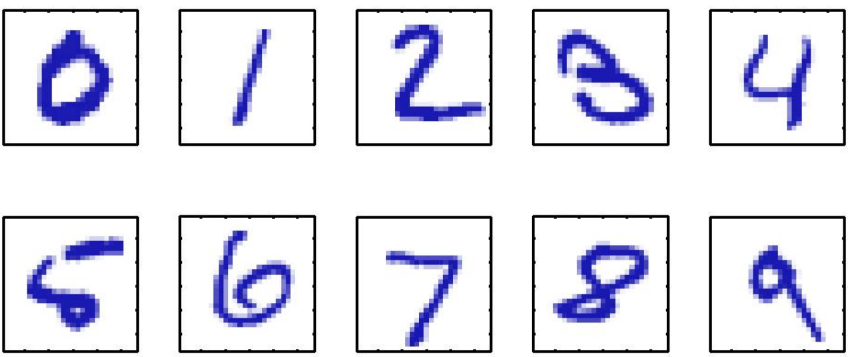
Structural features

- Sender IP address
- IP blacklist
- DNS information
- Email server
- URL links (non-matching)

Binary classification: SPAM or HAM

Example 2

Handwritten Digit Recognition

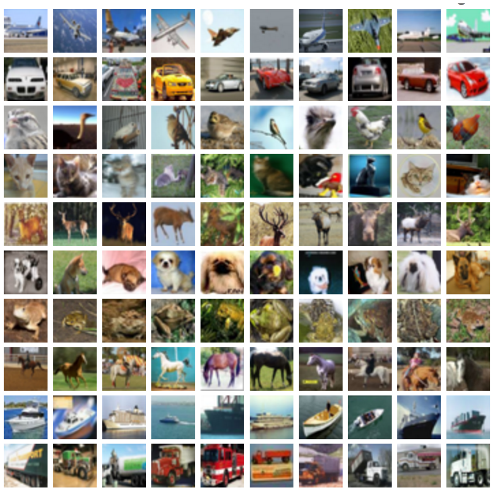


Multi-class classification

Example 3

Image classification

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



Multi-class classification

Supervised Learning Process

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled (Typically)

Normalization
Standardization

Feature Selection

Classification
Regression

Testing

New data → Learning model → Predictions

Unlabeled

Malicious
Benign
Classification

Risk score
Regression

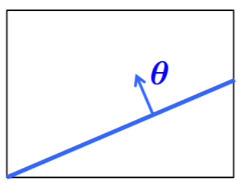
History of Perceptrons

- They were popularised by Frank Rosenblatt in the early 1960's.
  - They appeared to have a very powerful learning algorithm.
  - Lots of grand claims were made for what they could learn to do.
- In 1969, Minsky and Papert published a book called "Perceptrons" that analysed what they could do and showed their limitations.
  - Many people thought these limitations applied to all neural network models.
- The perceptron learning procedure is still widely used today for tasks with enormous feature vectors that contain many millions of features.

They are the basic building blocks for
Deep Neural Networks

Linear classifiers

- A **hyperplane** partitions space into 2 half-spaces
  - Defined by the normal vector $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$
    - $\boldsymbol{\theta}$ is orthogonal to any vector lying on the hyperplane
  - Assumed to pass through the origin
    - This is because we incorporated bias term θ_0 into it by $x_0 = 1$

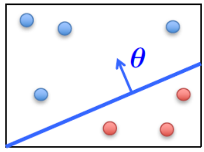


- Consider classification with +1, -1 labels ...

Linear classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

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



$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \text{ where } \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

– Note that: $\boldsymbol{\theta}^\intercal \boldsymbol{x} > 0 \implies y = +1$

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

All the points x on the hyperplane satisfy: $\theta^T x = 0$

Example: Spam

- Imagine 3 features (spam is “positive” class):
  1. free (number of occurrences of “free”)
  2. money (occurrences of “money”)
  3. BIAS (intercept, always has value 1)

$$\sum_{i=0}^{d} x_i \theta_i$$

“free money”

x

BIAS	:	1
free	:	1
money	:	1
...		

θ

BIAS	:	-3
free	:	4
money	:	2
...		

$(1)(-3)\ +$
$(1)(4)\ +$
$(1)(2)\ +$
$\dots$
$= 3$

$\sum_i x_i \theta_i > 0$ ➔ SPAM!!!

The Perceptron

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\theta_j \leftarrow \theta_j - \frac{1}{2}\underbrace{(h_\theta(x_i) - y_i)}_{\text{either 2 or -2}} x_{ij}$$

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

The Perceptron

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

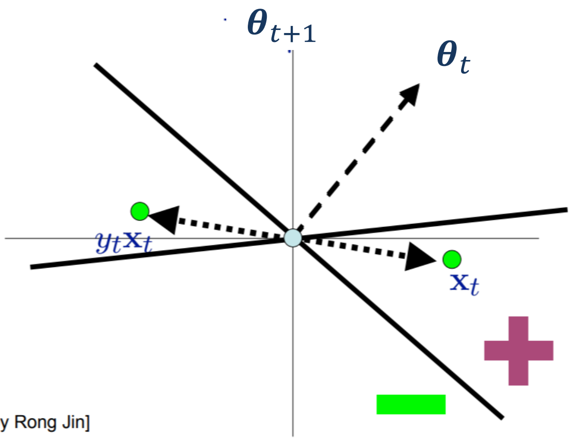
$$\theta_j \leftarrow \theta_j - \frac{1}{2}\underbrace{(h_\theta(x_i) - y_i)}_{\text{either 2 or -2}} x_{ij}$$

- Re-write as $\theta_j \leftarrow \theta_j + y_i x_{ij}$ (only upon misclassification)

Perceptron Rule: If x_i is misclassified, do

$$\theta \leftarrow \theta + y_i \, x_i$$

Geometric interpretation



[Slide by Rong Jin]

Online Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
 Receive training example (x_i, y_i)
 If $y_i \theta^T x_i \leq 0$ // prediction is incorrect
  $\theta \leftarrow \theta + y_i\, x_i$

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

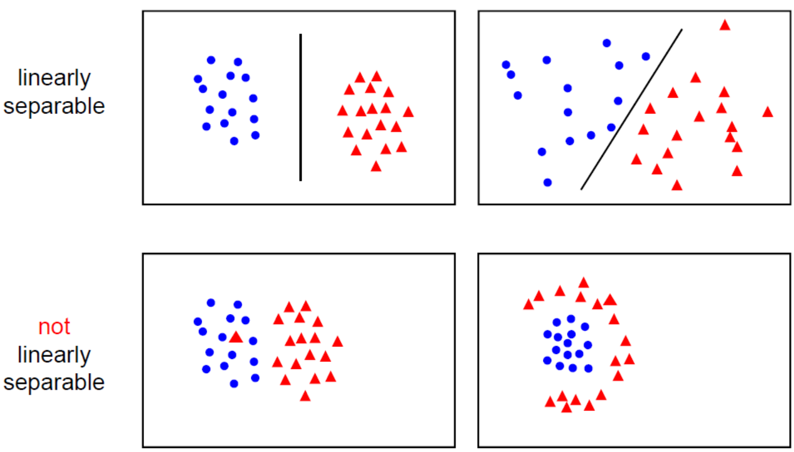
Batch learning – the learning mode where the model update is performed after observing the entire training set

Batch Perceptron

Given training data $\{x_i, y_i\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $y_i \theta^T x_i \leq 0$ // prediction for i^{th} instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y_i x_i$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$ // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$

Guaranteed to find separating hyperplane if data is linearly separable

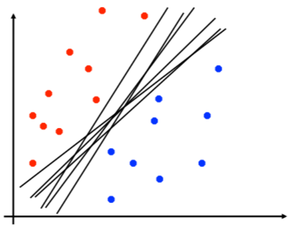
Linear separability



- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  - Move the decision boundary for every example



Which of this is optimal?

Improving the Perceptron

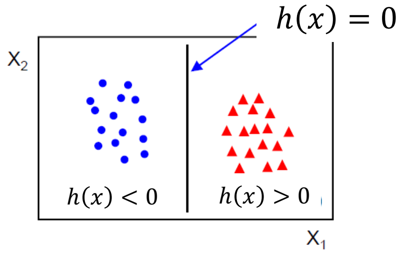
- The Perceptron produces many $\boldsymbol{\theta}$'s during training
- The standard Perceptron simply uses the final $\boldsymbol{\theta}$ at test time
  - This may sometimes not be a good idea!
  - Some other $\boldsymbol{\theta}$ may be correct on 1,000 consecutive examples, but one mistake ruins it!

- **Idea:** Use a combination of multiple perceptrons
  - (i.e., neural networks!)
- **Idea:** Use the intermediate $\boldsymbol{\theta}$'s
  - **Voted Perceptron**: vote on predictions of the intermediate $\boldsymbol{\theta}$'s
  - **Averaged Perceptron**: average the intermediate $\boldsymbol{\theta}$'s

Linear classifiers

A linear classifier has the form

$$h_\theta(x) = f(\theta^T x)$$



- Properties
  - $(\theta_0, \theta_1, \dots, \theta_d)$ = model parameters
  - Perceptron is a special case with $f = sign$
- Pros
  - Very compact model (size d)
  - Perceptron is fast

- Cons
  - Does not work for data that is not linearly separable

LDA

- Classify to one of k classes
- Logistic regression computes directly
  - $P[Y = 1|X = x]$
  - Assume sigmoid function
- LDA uses Bayes Theorem to estimate it
  - $P[Y = k|X = x] = \frac{P[X = x|Y = k]P[Y=k]}{P[X=x]}$
  - Let $\pi_k = P[Y = k]$ be the prior probability of class k and $f_k(x) = P[X = x|Y = k]$

LDA

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

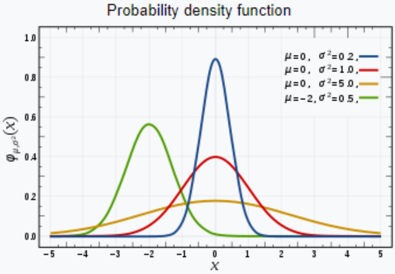
$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Assumption: $\sigma_1 = \ldots \sigma_k = \sigma$

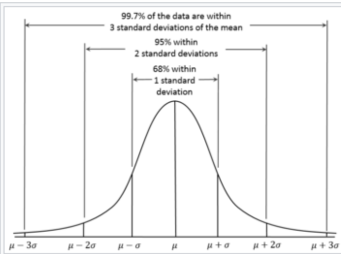
Gaussian Distribution

Normal Distribution

Probability density function



The red curve is the *standard normal distribution*



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Notation	$\mathcal{N}(\mu, \sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 > 0$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

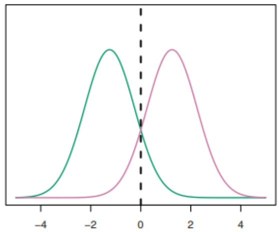
LDA decision boundary

Pick class k to maximize

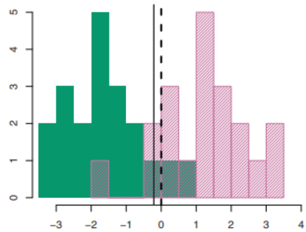
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2\sigma}$



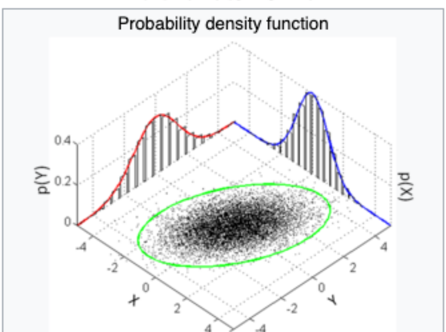
True decision boundary



Estimated decision boundary

Multi-Variate Normal

Multivariate normal



Many sample points from a multivariate normal distribution with $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 3/5 \\ 3/5 & 2 \end{bmatrix}$, shown along with the 3-sigma ellipse, the two marginal distributions, and the two 1-d histograms.

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

with k-dimensional mean vector

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}] = [\mathrm{E}[X_1], \mathrm{E}[X_2], \ldots, \mathrm{E}[X_k]]^{\mathrm{T}},$$

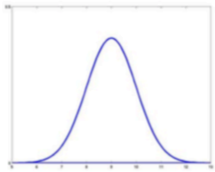
and $k \times k$ covariance matrix

$$\Sigma_{i,j} =: \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)] = \mathrm{Cov}[X_i, X_j]$$

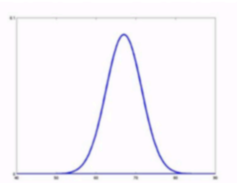
$$\boldsymbol{\Sigma} =: \mathrm{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^{\mathrm{T}}] = [\mathrm{Cov}[X_i, X_j]; 1 \le i, j \le k].$$

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \boldsymbol{\Sigma}_k^{-1}(x-\mu_k)}.$$

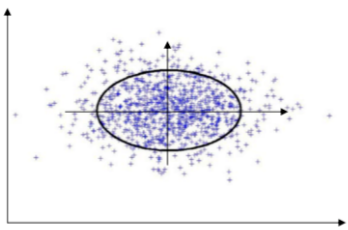
Example: Independent variables



people's heights:
$X \sim N(67, 20)$



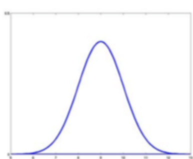
time people woke up this morning: $Y \sim N(9, 1)$



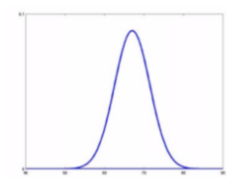
Co-variance matrix

$$\begin{bmatrix} 20 & 0 \\ 0 & 9 \end{bmatrix}$$

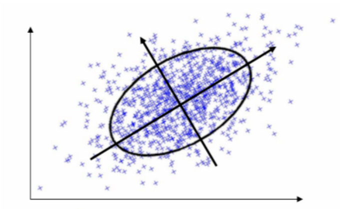
Example: Correlated variables



people's heights:
$X \sim N(67, 20)$



People's weight
$Y \approx N(177,40)$



Co-variance matrix

$$\begin{bmatrix} 20 & 5 \\ 5 & 40 \end{bmatrix}$$

Multi-variate LDA

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Assume $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$

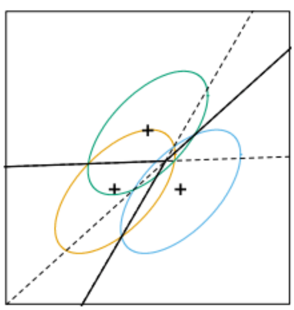
$$\log \frac{\Pr(Y = k|X = x)}{\Pr(Y = l|X = x)z} = \log \frac{f_k(x)}{f_\ell(x)} + \log \frac{\pi_k}{\pi_\ell}$$
$$= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^T \boldsymbol{\Sigma}^{-1}(\mu_k - \mu_\ell) + x^T \boldsymbol{\Sigma}^{-1}(\mu_k - \mu_\ell),$$

Linear decision boundary between classes *k* and *l*

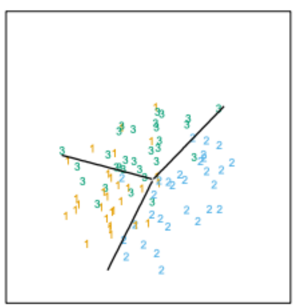
Linear discriminant functions $\delta_k(x) = x^T \boldsymbol{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T \boldsymbol{\Sigma}^{-1}\mu_k + \log \pi_k$

Given x, classify to class k: $argmax_k \delta_k(x)$

Example 3 classes



3 Normal distributions with same co-variance, but different means



LDA decision boundary

LDA in practice

Given training data $(x_i, y_i), i = 1, \dots, n, y_i \in \{1, \dots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k/n.$$

Given testing point x, predict k that maximizes:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

Multi-variate LDA

Given training data $(x_i, y_i), i = 1, \dots, n, y_i \in \{1, \dots, K\}$

1. Estimate mean and variance

- $\hat{\pi}_k = N_k/N$, where N_k is the number of class-k observations;
- $\hat{\mu}_k = \sum_{g_i=k} x_i/N_k$;
- $\hat{\mathbf{\Sigma}} = \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T/(N-K)$.

2. Estimate prior

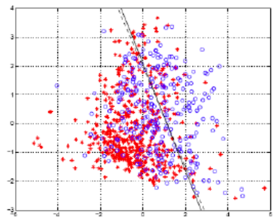
Given testing point x, predict k that maximizes:

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$

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

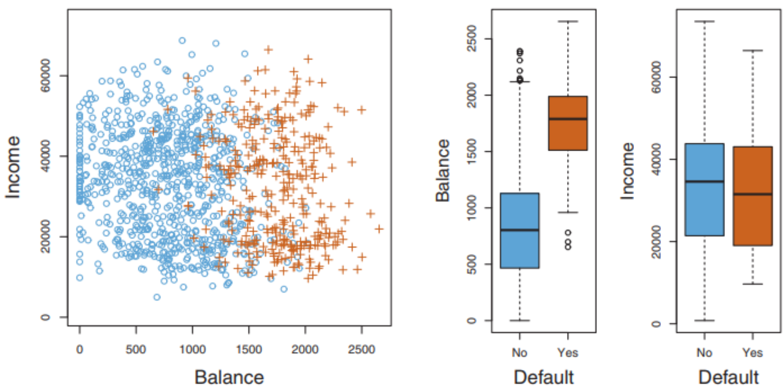


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of*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

Why not linear regression?

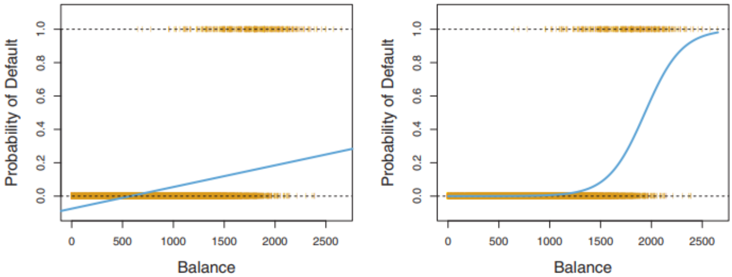


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default`*(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using 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 $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

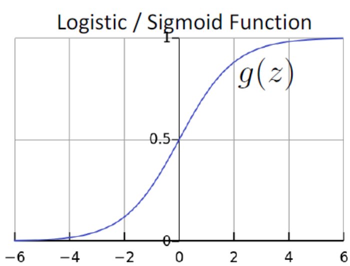
Can't just use linear regression with a threshold

- Logistic regression model:

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

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

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



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:

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

$$\mathrm{P}[y = 1|x;\theta] > ½$$

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

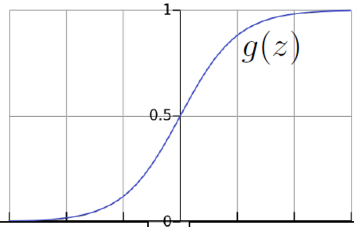
- Equivalent to:
  - $e^{\theta_0 + \sum_{i=1}^{d} \theta_j x_j} > 1$
  - $\theta_0 + \sum_{i=1}^{d} \theta_j x_j > 0$

Logistic Regression is a linear classifier!

Logistic Regression

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

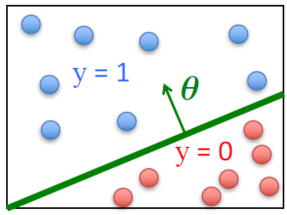
$$g(z) = \frac{1}{1+e^{-z}}$$



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

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large positive 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 $\left\{ (x_1, y_1) \quad , \quad (x_2, y_2) \quad , \ldots , \quad (x_N, y_N) \right\}$

  where $x_i \in R^d, y_i \in \{0,1\}$

- 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:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2$$

  - Using the logistic regression model

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

    results in a non-convex optimization

Maximum Likelihood Estimation (MLE)

Given training data $X = \{x_1, \dots, x_N\}$ with labels $Y = \{y_1, \dots, y_N\}$

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

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

General probabilistic method for classifier training

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

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

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i |, \theta)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)$$

- Substitute in model, and take negative to yield

Logistic regression objective:

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

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)$$

Objective for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i)\log\left(1 - h_\theta(x_i)\right)$$

- 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

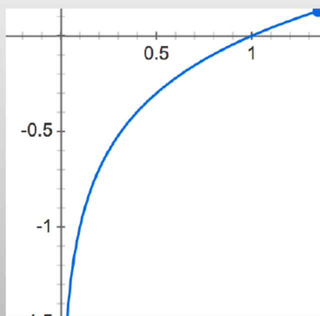
$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_\theta(x_i), y_i \right)$$

Cross-entropy loss

Intuition

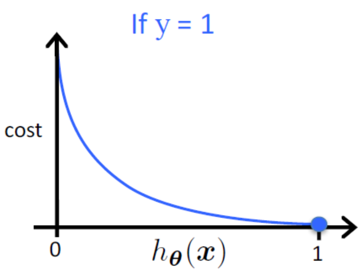
$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



Intuition

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

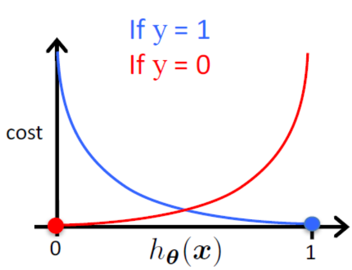


If y = 1

- 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

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



If y = 0

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

Acknowledgements

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