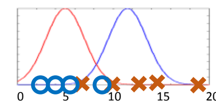
Logistic Classifier

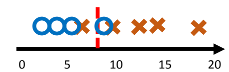
CISC 5800

Professor Daniel Leeds

Classification strategy: generative vs. discriminative



- Generative, e.g., Bayes/Naïve Bayes:
  - Identify probability distribution for each class
  - Determine class with maximum probability for data example
- Discriminative, e.g., Logistic Regression:
  - Identify boundary between classes
  - Determine which side of boundary new data example exists on



2

Linear algebra: data features

- Vector – list of numbers: each number describes a data **feature**
- Matrix – list of lists of numbers: features for each data point

	Document 1	Document 2	Document 3
Wolf	12	8	0
Lion	16	10	2
Monkey	14	11	1
Broker	0	1	14
Analyst	1	0	10
Dividend	1	1	12
⋮	⋮	⋮	⋮

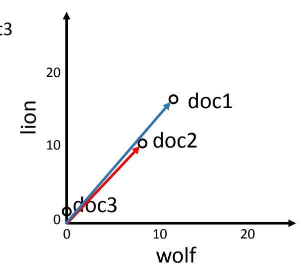
# of word occurrences

3

Feature space

- Each data feature defines a dimension in space

	Document1	Document2	Document3
Wolf	12	8	0
Lion	16	10	2
Monkey	14	11	1
Broker	0	1	14
Analyst	1	0	10
Dividend	1	1	12
⋮	⋮	⋮	⋮

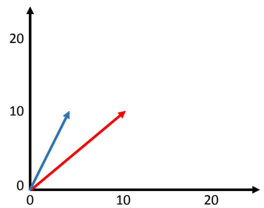


4

The dot product

The dot product compares two vectors:

- $\boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ $\quad \boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i \; = \boldsymbol{a}^T \boldsymbol{b}$



$$\begin{bmatrix} \mathbf{5} \\ \mathbf{10} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{10} \\ \mathbf{10} \end{bmatrix} = \mathbf{5 \times 10 + 10 \times 10}$$

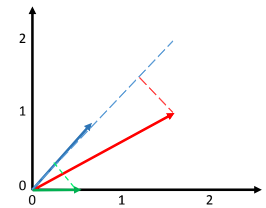
$$= \mathbf{50 + 100 = 150}$$

The dot product, continued $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i$

Magnitude of a vector is sum of squares of the elements

$$|\boldsymbol{a}| = \sqrt{\sum_i a_i^2}$$

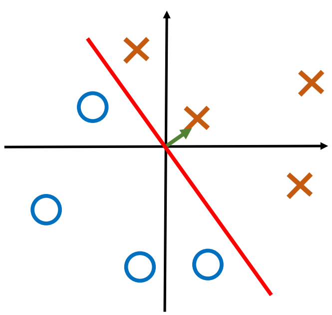
If $\boldsymbol{a}$ has unit magnitude, $\boldsymbol{a} \cdot \boldsymbol{b}$ is “projection” of $\boldsymbol{b}$ onto $\boldsymbol{a}$



$$\begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 1 \end{bmatrix} = 0.6 \times 1.5 + 0.8 \times 1$$

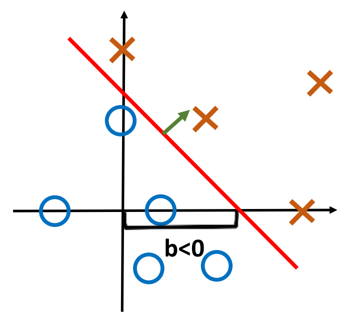
$$= 0.9 + 0.8 = 1.7$$

Separating boundary, defined by **w**



- Separating **hyperplane** splits **class 0** and **class 1**
- Plane is defined by line **w** perpendicular to plane
- Is data point **x** in class 0 or class 1? $\mathbf{w}^T\mathbf{x}$+b > 0 class **1**
  $\mathbf{w}^T\mathbf{x}$+b < 0 class **0**

Separating boundary, defined by **w** and b



Example:

$\mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ b=-4

$\mathbf{x}^1 = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $\mathbf{w}^T\mathbf{x}^1$ +b=**1**

$\mathbf{x}^2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ $\mathbf{w}^T\mathbf{x}^2$ +b=**-2**

Notational simplification

Recall: $\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{w}\cdot\boldsymbol{x} = \sum_{i=1}^{n} w_i x_i$

Define $x'_{1:n} = x_{1:n}$ and $x'_{n+1} = 1$ for all inputs $\boldsymbol{x}$ and $w'_{1:n} = w_{1:n}$ and $w'_{n+1} = b$

Now $\boldsymbol{w}'^T\boldsymbol{x}' = \boldsymbol{w}^T\boldsymbol{x} + b$

Let's assume $x_{n+1}=1$ always, and $w_{n+1}=b$ always

From real-number projection to 0/1 label

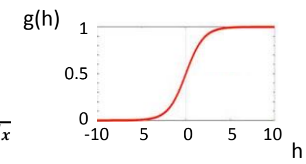
Binary classification: 0 is class A, 1 is class B

Sigmoid function stands in for p(**x**|y)

Sigmoid: $g(h) = \frac{1}{1+e^{-h}}$

$p(y=0|x;\theta) = 1 - g(\boldsymbol{w}^T\boldsymbol{x}) = \frac{e^{-w^T x}}{1+e^{-w^T x}}$

$p(y=1|x;\theta) = g(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1+e^{-w^T x}}$



$$\boldsymbol{w}^T\boldsymbol{x} = \sum_j \boldsymbol{w_j x_j} + b$$

14

Learning parameters for classification

Similar to MLE for Bayes classifier

"Likelihood" for data points $y^1, \ldots, y^n$
(different from Bayesian likelihood)

- If y^i in class A, $y^i=0$, multiply (1-g($\boldsymbol{x}^i$;**w**))
- If y^i in class B, $y^i=1$, multiply (g($\boldsymbol{x}^i$;**w**))

$$\underset{w}{\operatorname{argmax}} L(y|x;w) = \prod_i \left(1 - g(\boldsymbol{x}^i;\boldsymbol{w})\right)^{(1-y^i)} g(\boldsymbol{x}^i;\boldsymbol{w})^{y^i}$$

15

Learning parameters for classification

The long derivation

- Similar to MLE for Bayes classifier
- "Likelihood" for data points $y^1, \ldots, y^n$ (different from Bayesian likelihood)
  - If y^i in class A, $y^i=0$, multiply (1-g(x^i;w))
  - If y^i in class B, $y^i=1$, multiply (g(x^i;w))

$$\underset{w}{\operatorname{argmax}} L(y|x;w) = \prod_i \left(1 - g(x^i;w)\right)^{(1-y^i)} g(x^i;w)^{y^i}$$

$$LL(y|x;w) = \sum_i (1-y^i)\ln\left(1 - g(x^i;w)\right) + y^i \ln\left(g(x^i;w)\right)$$

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i \frac{x_j^i e^{-w^T x}}{\left(1+e^{-w^T x}\right)^2}\left(\frac{-(1-y^i)}{1-g(x^i;w)} + \frac{y^i}{g(x^i;w)}\right)$$

16

Learning parameters for classification

The long derivation

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i \frac{x_j^i e^{-w^T x}}{\left(1+e^{-w^T x}\right)^2}\left(\frac{-(1-y^i)}{1-g(x^i;w)} + \frac{y^i}{g(x^i;w)}\right)$$

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i x_j^i g(x^i;w)\left(1-g(x^i;w)\right)\left(\frac{-(1-y^i)}{1-g(x^i;w)} + \frac{y^i}{g(x^i;w)}\right)$$

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i x_j^i \left(-(1-y^i) g(x^i;w) + y^i\left(1-g(x^i;w)\right)\right)$$

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i x_j^i \left(y^i - g(x^i;w)\right)$$

17

Iterative gradient ascent

y^i – true data label
$g(\mathbf{w}^T\mathbf{x}^i)$ – computed data label

Begin with initial guessed weights w

For each data point $(y^i, \mathbf{x}^i)$, update each weight w_j

$$w_j \leftarrow w_j + \varepsilon x_j^i \left(y^i - g(\boldsymbol{w}^T \boldsymbol{x}^i)\right)$$

Choose ε so change is not too big or too small – "**step size**"

Intuition

$x_j^i \left(y^i - g(\boldsymbol{w}^T \boldsymbol{x}^i)\right)$

- If $y^i=1$ and $g(\mathbf{w}^T\mathbf{x}^i)=0$, and $x^i_j>0$, make w_j larger and push $\mathbf{w}^T\mathbf{x}^i$ to be larger
- If $y^i=0$ and $g(\mathbf{w}^T\mathbf{x}^i)=1$, and $x^i_j>0$, make w_j smaller and push $\mathbf{w}^T\mathbf{x}^i$ to be smaller

21

Iterative gradient ascent – big picture

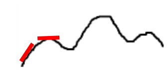
Initialize **w** with random values

- Loop across all training data $\mathbf{x}^i$ for each feature x^i_j
- Repeat this loop many times (100x, or 1000x, etc.)

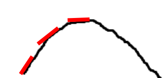
```
All w's = 0 or rand
for iter in range(??):     # This is PSEUDOCODE
  All w's = 0 or rand
  for dataPt i :
    for feature j :
      updateJ += updateJ + eps xIJ (yI - g)
   wJ <- wJOld + updateJ
```

Gradient ascent for L(y|x;w)

- Typical gradient ascent can get stuck in local maxima



- $L(y|\boldsymbol{x};\boldsymbol{w}) = \prod_i \left(1-g(\boldsymbol{x}^i;\boldsymbol{w})\right)^{(1-y^i)} g(\boldsymbol{x}^i;\boldsymbol{w})^{y^i}$
- L is "convex" – it has at most 1 maximum



MAP for discriminative classifier

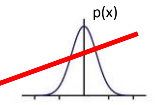
MLE: P(y=1|**x**;**w**) ~ g($\mathbf{w}^T\mathbf{x}$)

MAP: P(y=1,**w**|**x**) ∝ P(y=1|**x**;**w**) P(**w**) ~ g($\mathbf{w}^T\mathbf{x}$) ?Prior?
(different from Bayesian posterior)

P(**w**) priors
- L2 regularization – minimize all weights
- L1 regularization – minimize number of non-zero weights

24

MAP – L2 regularization



- P(y=1,w|x) ∝ P(y=1|x,w) P(w):

$$L(y,w|x) = \prod_i \left(1 - g(\boldsymbol{x}^i;\boldsymbol{w})\right)^{(1-y^i)} g(\boldsymbol{x}^i;\boldsymbol{w})^{y^i} \quad \prod_j e^{-\frac{w_j^2}{2\lambda}}$$

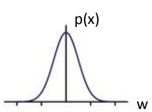
$$LL(y,w|x) = \sum_i y^i w^T x^i - w^T x^i + \log\left(g(w^T x^i)\right) - \sum_j \frac{w_j^2}{2\lambda}$$

$$\frac{\partial}{\partial w_j} LL(y,w|x) = \sum_i x_j^i \left(y^i - g(w^T x^i)\right) - \frac{w_j}{\lambda}$$

This slide is correct but uses slightly different notation from past slides. See next slide for consistent notation

25

MAP – L2 regularization



- P(y=1,w|x) ∝ P(y=1|x;w) P(w):

$$L(y|x;w) = \prod_i \left(1 - g(\boldsymbol{x}^i;\boldsymbol{w})\right)^{(1-y^i)} g(\boldsymbol{x}^i;\boldsymbol{w})^{y^i} \quad \prod_j e^{-\frac{w_j^2}{2\lambda}}$$

$$LL(y|x;w) = \sum_i \left((1-y^i)\ln\left(1 - g(x^i;w)\right) + y^i \ln\left(g(x^i;w)\right)\right) - \sum_j \frac{w_j^2}{2\lambda}$$

$$\frac{\partial}{\partial w_j} LL(y|x;w) = \sum_i x_j^i \left(y^i - g(w^T x^i)\right) - \frac{w_j}{\lambda}$$

26

L1 and L2 update rules

L1: $w_j \leftarrow w_j + \varepsilon x_j^i\left(y^i - g(\boldsymbol{w}^T\boldsymbol{x}^i)\right) - \frac{\text{sign}(w_j)}{\lambda}$ $\quad \text{sign}(w_j) = \begin{cases} +1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \end{cases}$

L2: $w_j \leftarrow w_j + \varepsilon x_j^i\left(y^i - g(\boldsymbol{w}^T\boldsymbol{x}^i)\right) - \frac{w_j}{\lambda}$

Note, L1 comes from "Laplacian distribution" P(w_j)=$e^{-\frac{|w_j|}{\lambda}}$

Thinking about your data: numeric and non-numeric features

Data to be classified can have multiple features $\boldsymbol{x}^i = \begin{bmatrix} x_1^i \\ \vdots \\ x_n^i \end{bmatrix}$

Each feature could be:

- Numeric: Loudness of music, from 0 to 30 decibels
- Non-numeric: Action, including Laugh, Cry, Jump, Dance

Classifier choice

Logistic regression only makes sense for numeric data

Gaussian Bayes only makes sense for numeric data

Multinomial Bayes makes sense for non-numeric data

Non-numeric features -> numeric

You may map non-numeric features to continuous space

Example:

- Mood={Depressed, Disappointed, Neutral, Happy, Excited}
- Switch to: HappinessLevel = {-2, -1, 0, 1, 2}
- One-hot coding: $x_{Depressed}$ – 0 or 1; $x_{Disappointed}$ – 0 or 1; $x_{Neutral}$ – 0 or 1
  - Inflates dimensions, works better if large training data set