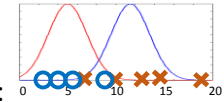


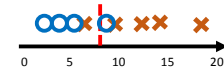
Logistic Classifier

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



Linear algebra: data features

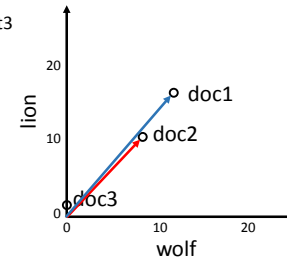
- Vector – list of numbers:
each number describes a data **feature**
- Matrix – list of lists of numbers:
features for each data point

| | 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 |
| ⋮ | ⋮ | ⋮ | ⋮ |

Feature space

- Each data feature defines a dimension in space

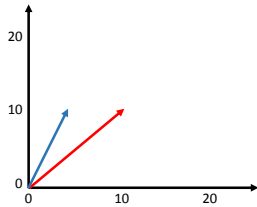
| | Document1 | Document2 | Document3 |
|----------|-----------|-----------|-----------|
| Wolf | 12 | 8 | 0 |
| Lion | 16 | 10 | 2 |
| Monkey | 14 | 11 | 1 |
| Broker | 0 | 1 | 14 |
| Analyst | 1 | 0 | 10 |
| Dividend | 1 | 1 | 12 |
| ⋮ | ⋮ | ⋮ | ⋮ |



The dot product

The dot product compares two vectors:

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



$$\begin{bmatrix} 5 \\ 10 \end{bmatrix} \cdot \begin{bmatrix} 10 \\ 10 \end{bmatrix} = 5 \times 10 + 10 \times 10 \\ = 50 + 100 = 150$$

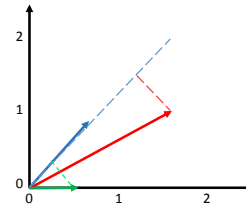
6

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

Magnitude of a vector is the sum of the squares of the elements

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

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

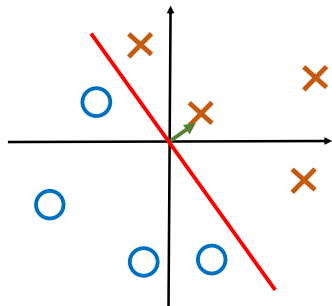


$$\begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 1 \end{bmatrix} = 0.6 \times 1.5 + 0.8 \times 1 \\ = 0.9 + 0.8 = 1.7$$

$$\begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} = 0.6 \times 0 + 0.8 \times 0.5 \\ = 0 + .4 = 0.4$$

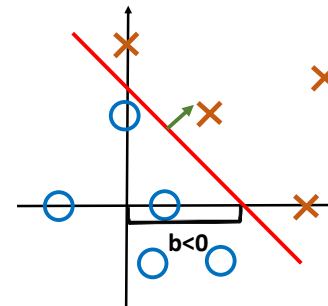
9

Separating boundary, defined by \mathbf{w}



- Separating **hyperplane** splits **class 0** and **class 1**
- Plane is defined by line \mathbf{w} perpendicular to plane
- Is data point \mathbf{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 \mathbf{w} and b



Example:

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

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

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

12

Notational simplification

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

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

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

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

From real-number projection to 0/1 label

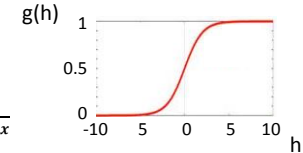
Binary classification: 0 is class A, 1 is class B

Sigmoid function stands in for $p(\mathbf{x}|y)$

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

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

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



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

Learning parameters for classification

Similar to MLE for Bayes classifier

“Likelihood” for data points y^1, \dots, y^n
(different from Bayesian likelihood)

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

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

15

Learning parameters for classification

The long derivation

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

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

$$LL(y|x; w) = \sum_i (1 - y^i) \log(1 - g(\mathbf{x}^i; \mathbf{w})) + y^i \log(g(\mathbf{x}^i; \mathbf{w}))$$

$$LL(y|x; w) = \sum_i y^i \log \frac{g(\mathbf{x}^i; \mathbf{w})}{1 - g(\mathbf{x}^i; \mathbf{w})} + \log(1 - g(\mathbf{x}^i; \mathbf{w}))$$

16

Learning parameters for classification

The long derivation

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

$$LL(y|x; w) = \sum_i y^i \log \frac{g(x^i; w)}{1 - g(x^i; w)} + \log(1 - g(x^i; w))$$

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

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

$$LL(y|x; w) = \sum_i y^i w^T x^i - w^T x^i - \log(1 + e^{-w^T x^i})$$

17

Learning parameters for classification

The long derivation

$$w^T x = \sum_j w_j x_j$$

$$g'(h) = \frac{e^{-h}}{(1 + e^{-h})^2}$$

$$LL(y|x; w) = \sum_i y^i w^T x^i - w^T x^i + \log(g(w^T x^i))$$

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

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

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

18

Iterative gradient ascent

 y^i – true data labelBegin with initial guessed weights w $g(\mathbf{w}^T \mathbf{x}^i)$ – computed data labelFor each data point (y^i, \mathbf{x}^i) , update each weight w_j

$$w_j \leftarrow w_j + \epsilon x_j^i (y^i - g(\mathbf{w}^T \mathbf{x}^i))$$

Choose ϵ so change is not too big or too small – “step size”*Intuition*

$$x_j^i (y^i - g(\mathbf{w}^T \mathbf{x}^i))$$

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

19

Iterative gradient ascent – big picture

Initialize \mathbf{w} with random values

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

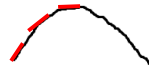
Gradient ascent for $L(y|\mathbf{x};\mathbf{w})$

- Typical gradient ascent can get stuck in local maxima

$$L(y|\mathbf{x};\mathbf{w}) = \prod_i (1 - g(\mathbf{x}^i; \mathbf{w}))^{(1-y^i)} g(\mathbf{x}^i; \mathbf{w})^{y^i}$$



- L is "convex" – it has at most 1 maximum



MAP for discriminative classifier

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

$$\text{MAP: } P(y=1, \mathbf{w}|\mathbf{x}) \propto P(y=1|\mathbf{x};\mathbf{w}) P(\mathbf{w}) \sim g(\mathbf{w}^T \mathbf{x}) ???$$

(different from Bayesian posterior)

$P(\mathbf{w})$ priors

- L2 regularization – minimize all weights
- L1 regularization – minimize number of non-zero weights

22

L1 and L2 update rules

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

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

Thinking about your data:
numeric and non-numeric features

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

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}