Discriminative classifiers: Logistic Regression, SVMs

CISC 5800
Professor Daniel Leeds

Maximum A Posteriori: a quick review

- Likelihood: $P(D|\theta) = P(D|p) = p^{|H|}(1-p)^{|T|}$
- Prior: $P(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} = P(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}$
- Posterior Likelihood x prior = $P(D|\theta)P(\theta)$

Choose α and β to give the prior belief of Heads bias $p \in [0,1]$

Higher α: Heads more likely
Higher β: Tails more likely

- MAP estimate:

$$\underset{\theta}{\operatorname{argmax}} \log P(D|\theta) + \log P(\theta)$$

$$\underset{p}{\operatorname{argmax}} \log P(D|p) + \log P(p)$$

$$p = \frac{|H| + (\alpha - 1)}{|H| + (\alpha - 1) + |T| + (\beta - 1)}$$

Estimate each P(X_i|Y) through MAP

Incorporating prior for each class β_j

$$P(X_i = x_k | Y = y_j) = \frac{\#D(X_i = x_k \wedge Y = y_j) + (\beta_j - 1)}{\#D(Y = y_j) + \sum_m(\beta_m - 1)}$$

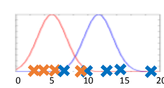
$$P(Y = y_j) = \frac{\#D(Y = y_j) + (\beta_j - 1)}{|D| + \sum_m(\beta_m - 1)}$$

$(\beta_j - 1)$ – "frequency" of class j
$\sum_m(\beta_m - 1)$ – "frequencies" of all classes

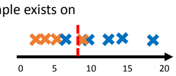
Note: both X and Y can take on multiple values (binary **and** beyond)

3

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



4

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
⋮	⋮	⋮	⋮

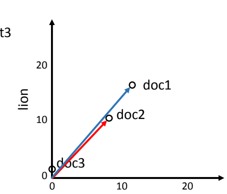
# of word occurrences

5

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
⋮	⋮	⋮	⋮

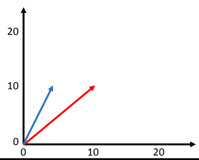


6

The dot product

The dot product compares two vectors:

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



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

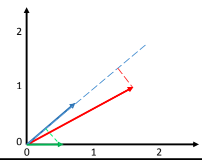
$= 50 + 100 = 150$

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

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

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

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



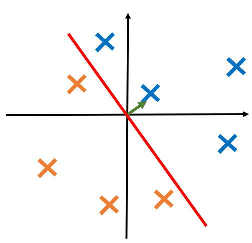
$\begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 1 \end{bmatrix} = .71 \times 1.5 + .71 \times 1$

$\approx 1.07 + .71 = 1.78$

$\begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} = .71 \times 0 + .71 \times 0.5$

$\approx 0 + .35 = 0.35$

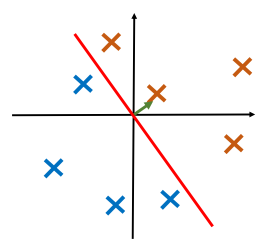
Separating boundary, defined by w



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

Separating boundary, defined by w

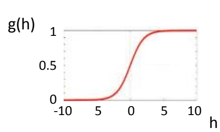
More typically



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

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

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

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 $(1-g(x^i;w))$
  - If y^i in class B, $y^i=1$, multiply $(g(x^i;w))$

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

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

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

Learning parameters for classification

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

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

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

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

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

13

Learning parameters for classification

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

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

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

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

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

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

14

Iterative gradient ascent

y^i – true data label
$g(w^T x^i)$ – computed data label

- Begin with initial guessed weights w
- For each data point (y^i, x^i), update each weight w_j

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

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

Intuition

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

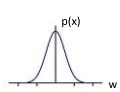
15

MAP for discriminative classifier

- MLE: P(y=1|x;w) ~ $g(w^T x)$
- MAP: P(y=1,w|x) ∝ P(y=1|x;w) P(w) ~ $g(w^T x)$???
  (different from Bayesian posterior)
- P(w) priors
  - L2 regularization – minimize all weights
  - L1 regularization – minimize number of non-zero weights

16

MAP – L2 regularization



- P(y=1,w|x) ∝ P(y=1|x;w) P(w):

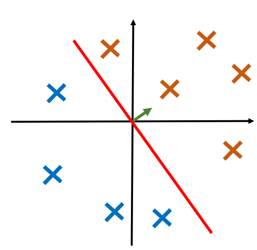
$$L(y,w|x) = \prod_i \left(1-g(x^i;w)\right)^{(1-y^i)} g(x^i;w)^{y^i} \quad \prod_j e^{-\frac{w_j^2}{2\lambda}}$$

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

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

17

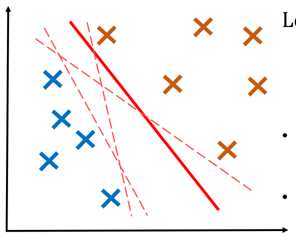
Separating boundary, defined by w



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

19

But, where do we place the boundary?



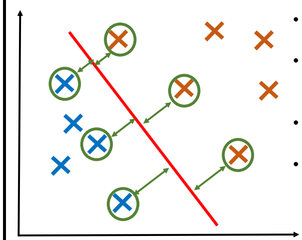
Logistic regression:

$$LL(y|x;w):$$

$$\sum_i (y^i - 1)\mathbf{w}^T x^i - \log\left(1 + e^{-\mathbf{w}^T x^i}\right)$$

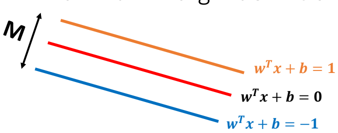
- Each data point $\boldsymbol{x}^i$ considered for boundary $\boldsymbol{w}$
- Outlier data pulls boundary towards it

Max margin classifiers



- Focus on boundary points
- Find largest margin between boundary points on both sides
- Works well in practice
- We can call the boundary points **<u>"support vectors"</u>**

Maximum margin definitions

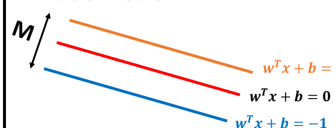


Classify as +1 if $\boldsymbol{w^Tx + b \geq 1}$
Classify as -1 if $\boldsymbol{w^Tx + b \leq -1}$
Undefined if $-1 \leq \boldsymbol{w^Tx + b} \leq \mathbf{1}$

- M is the margin width
- x^+ is a +1 point closest to boundary, x^- is a -1 point closest to boundary

- $x^+ = \lambda w + x^-$
- $|x^+ - x^-| = M$

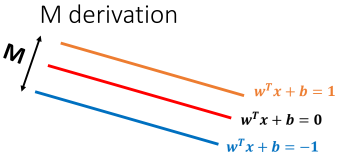
λ derivation



- $\boldsymbol{w^Tx^- + b = -1}$
- $\boldsymbol{w^Tx^+ + b = +1}$
- $\boldsymbol{x^+} = \lambda w + \boldsymbol{x^-}$

- $\boldsymbol{w^Tx^+ + b = +1}$
- $\boldsymbol{w^T}(\lambda w + \boldsymbol{x^-}) + \boldsymbol{b} = +\mathbf{1}$
- $\lambda \boldsymbol{w^T} w + \boldsymbol{w^Tx^-} + \boldsymbol{b} = +\mathbf{1}$
- $\lambda \boldsymbol{w^T} w - \mathbf{1} - \boldsymbol{b} + \boldsymbol{b} = +\mathbf{1}$
- $\lambda = \frac{2}{\boldsymbol{w^T}w}$

M derivation



- $\boldsymbol{w^Tx^- + b = -1}$
- $\boldsymbol{w^Tx^+ + b = +1}$
- $\boldsymbol{x^+} = \lambda w + \boldsymbol{x^-}$
- $|x^+ - x^-| = M$

- $M = |\lambda w + \boldsymbol{x^-} - x^-| = |\lambda w| = \lambda |w|$
- $M = \lambda \sqrt{w^T w}$
- $M = \frac{2}{w^T w}\sqrt{w^T w} = \frac{2}{\sqrt{w^T w}}$

maximize M minimize $w^T w$

Support vector machine (SVM) optimization

- $\max_w M = \frac{2}{\sqrt{w^T w}}$

$\min_w w^T w$

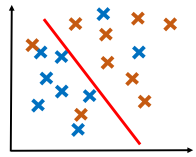
subject to

$\boldsymbol{w^Tx + b \geq 1}$ **for x in class 1**

$\boldsymbol{w^Tx + b \leq -1}$ **for x in class -1**

Support vector machine (SVM) optimization *with slack variables*

What if data not linearly separable?



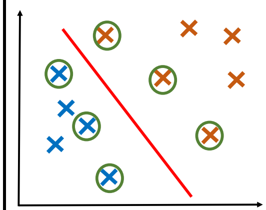
$\min_w w^T w + C \sum_i \varepsilon_i$

subject to

$\boldsymbol{w^T x + b \geq 1 - \varepsilon_i}$ **for x in class 1**

$\boldsymbol{w^T x + b \leq -1 + \varepsilon_i}$ **for x in class -1**

Alternate SVM formulation



$$w = \sum_i \alpha_i x_i y_i$$

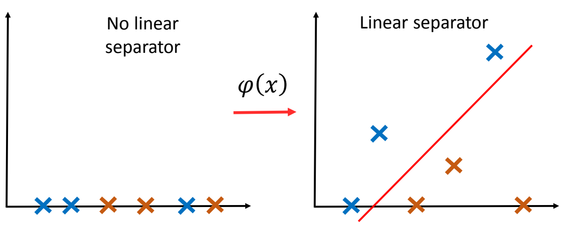
Support vectors x_i have $\alpha_i > 0$

y_i are the data labels +1 or -1

To classify sample x_j, compute:

$$w^T x_j + b = \sum_i \alpha_i y_i x_i x_j + b$$

Classifying with additional dimensions



Mapping function(s)

- Map from low-dimensional space $x = (x_1, x_2)$ to higher dimensional space $\varphi(x) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$
- N data points guaranteed to be separable in space of N-1 dimensions or more

$$\boldsymbol{w = \sum_i \alpha_i \varphi(x_i) y_i}$$

Classifying x_j:

$$\sum_i \alpha_i y_i \varphi(x_i) \varphi(x_j) + b$$

Kernels

Classifying x_j:

$$\sum_i \alpha_i y_i \varphi(x_i) \varphi(x_j) + b$$

Kernel trick:

- Estimate high-dimensional dot product with function
- $K(x_i, x_j) = \varphi(x_i) \varphi(x_j)$
- E.g., $K(x_i, x_j) = \exp\left(-\frac{(x_i - x_j)^2}{2\sigma^2}\right)$