10-701 Recitation 2

Naïve Bayes, Logistic Regression, Spectral Clustering

Bayes Rule

- Suppose you observe variables X, and you want to predict a variable Y
  - Want to know P(Y|X)

- Bayes Rule states:

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Why is this useful?

- Suppose we believe "Y generates X" according to some probability distribution
  - We want to construct P(X|Y)
  - We also have a prior on Y, P(Y)

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Notice that P(X|Y) and P(Y) are terms in the Bayes Rule expression

- As for P(X), observe that

$$P(X) \quad = \quad \sum_{y} P(X \mid Y = y) P(Y = y)$$

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Thus, if we could learn P(X|Y) and P(Y) from the data, we can compute P(Y|X)
- From there, we can predict argmax_Y[P(Y|X)] as the most likely value of Y, given data X

Naïve Bayes

- Suppose X is a collection of variables $X_1, \ldots, X_N$
  - Thus, $P(X|Y) = P(X_1, \ldots, X_N|Y)$

- What form should $P(X_1, \ldots, X_N|Y)$ take?

- Naïve Bayes assumes

$$P(X_1, \ldots, X_N|Y) = \prod_{i=1}^{N} P(X_i|Y)$$

Naïve Bayes

$$P(X_1, \ldots, X_N | Y) = \prod_{i=1}^{N} P(X_i | Y)$$

- The Naïve Bayes form assumes that the X_i are <u>conditionally independent</u> given Y

- Not always the “right” assumption, but a good starting point nonetheless

Naïve Bayes

$$P(X_1, \ldots, X_N | Y) = \prod_{i=1}^{N} P(X_i | Y)$$

- Why is this conditional independence assumption useful?
  - Consider binary X and Y. How is learning $P(X_i | Y)$ better than learning $P(X_1, \ldots, X_N | Y)$?

Learning P(X|Y) and P(Y)

- We want to maximize P(X|Y) and P(Y) w.r.t their parameters

- Discrete case: parameters are normalized counts
  - $P(X_i = x_j \mid Y = y_k) := \text{Count}(X_i = x_j \text{ and } Y = y_k) / \text{Count}(Y = y_k)$
  - $P(Y = y_k) := \text{Count}(Y = y_k) / M$

- Gaussian X, Discrete Y: parameters are mean/variance
  - $\mu_{ik} = \text{mean}(X_i \mid Y = y_k)$
  - $\sigma^2_{ik} = \text{var}(X_i \mid Y = y_k)$
  - Then:

$$P(X_i = x \mid Y = y_k) \quad = \quad \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left\{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}\right\}$$

Inference

- Compute:

$$\underset{Y}{argmax}\, P(Y|X) \propto P(Y) \prod_{i=1}^{N} P(X_i|Y)$$

- Do computations in logspace to prevent over/underflow:

$$\underset{Y}{argmax}\, P(Y) \prod_{i=1}^{N} P(X_i|Y) =$$

$$\underset{Y}{argmax}\, \log P(Y) + \sum_{i=1}^{N} \log P(X_i|Y)$$

Logistic Regression

- In Naïve Bayes, we learnt P(X|Y) and P(Y) in order to compute P(Y|X)

- Logistic regression learns P(Y|X) <u>directly</u> for binary Y and real-valued X
  - LR is an example of a <u>discriminative</u> model
  - NB is a <u>generative</u> model

Logistic Regression

- Logistic regression assumes

$$P(Y=0|\mathbf{X},\mathbf{w}) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

- Which implies

$$P(Y=1|\mathbf{X},\mathbf{w}) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

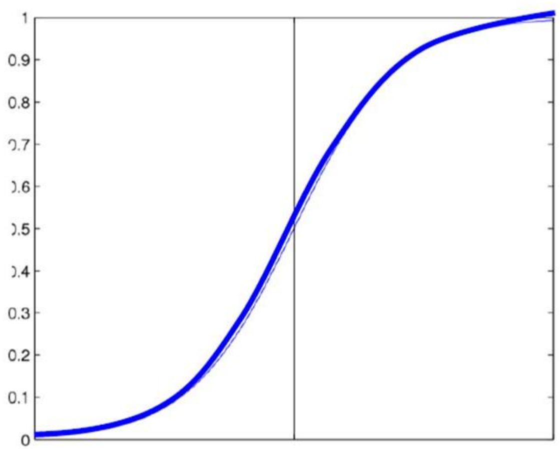
$$(= \frac{1}{1+exp(-w_0-\sum_i w_i X_i)})$$

Logistic Regression

$$P(Y = 0|\mathbf{X}, \mathbf{w}) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

- LR has a linear decision boundary
  - P(Y = 1|X,w) > 0.5 when $w_0 + \sum_i w_i X_i > 0$

- Logistic function $\frac{1}{1 + exp(-z)}$ is sigmoid



Learning parameters w

- Goal: Maximize conditional likelihood P(Y|X,w) w.r.t w

$$\widehat{\mathbf{w}}_{MCLE} = \arg\max_{\mathbf{w}} \prod_{j=1}^{L} P(Y^{(j)} \mid X^{(j)}, \mathbf{w})$$

- Maximizing this is difficult, so we maximize log(P(Y|X,w)) instead:

$$\max_{\mathbf{w}} \; l(\mathbf{w}) \equiv \ln \prod_{j}^{L} P(y^j|\mathbf{x}^j, \mathbf{w})$$
$$= \sum_{j}^{L} y^j(w_0 + \sum_{i}^{n} w_i x_i^j) - \ln(1 + exp(w_0 + \sum_{i}^{n} w_i x_i^j))$$

Learning parameters w

$$\max_{\mathbf{w}} \ l(\mathbf{w}) \equiv \ln \prod_{j}^{L} P(y^j|\mathbf{x}^j, \mathbf{w})$$

$$= \sum_{j}^{L} y^j (w_0 + \sum_{i}^{n} w_i x_i^j) - \ln(1 + exp(w_0 + \sum_{i}^{n} w_i x_i^j))$$

- This function has no closed-form solution for its maximum

- But it is concave, so we can use gradient ascent to converge on the maximum

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i^{(t)}}$$

Multiclass Logistic Regression

- What if Y takes on K > 2 values?

- One solution: K-class classification
  - For each class k < K:

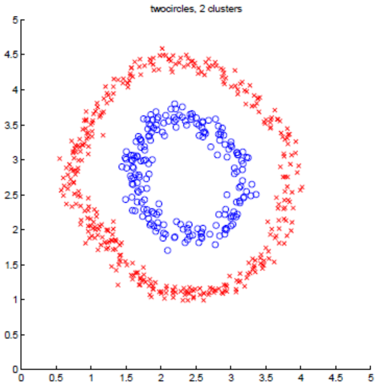
$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{d} w_{ki}X_i)}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{d} w_{ji}X_i)}$$

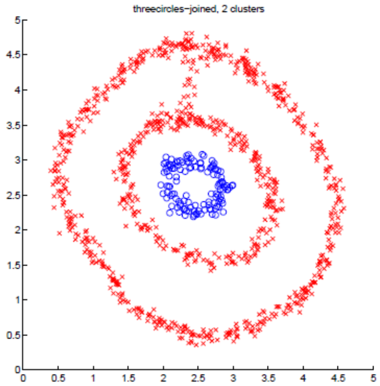
  - For class K

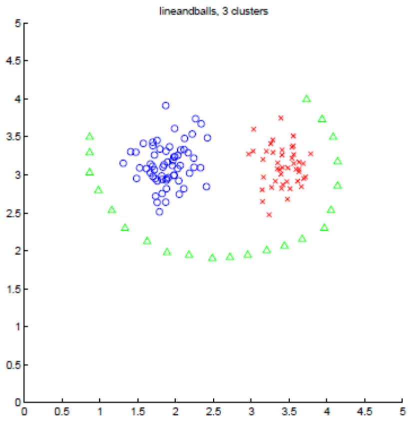
$$P(Y = y_K|X) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{d} w_{ji}X_i)}$$

Spectral Clustering

- Popular clustering method that overcomes some limitations of k-means
  - For instance, k-means fails badly on data like these:



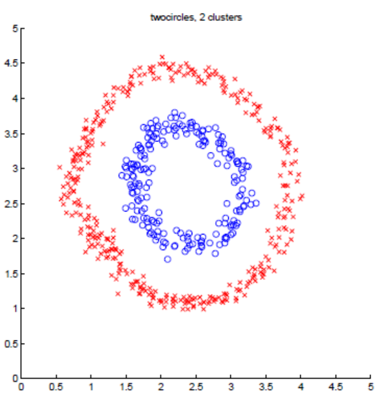


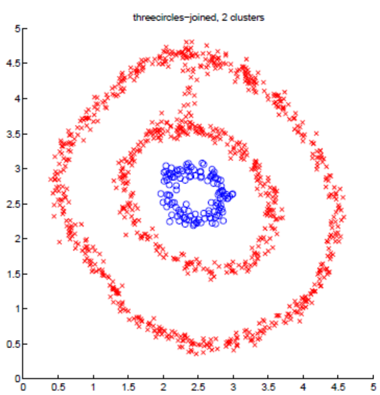


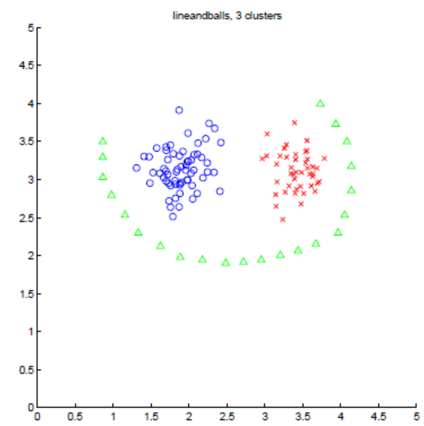
Source: Ng, Jordan, Weiss (2002)

Spectral Clustering

- Here, Euclidean distance does not match intuitive notion of “similarity”
  - Instead, similarity is akin to “connectedness”







Source: Ng, Jordan, Weiss (2002)

Spectral Clustering

- Spectral clustering partitions data points into tightly-connected groups

- This is done in 2 steps:
  1. Construct a graph G from the data X
  2. Solve the <u>relaxed</u> Normalized Cut problem on G

Data Graph G

- The data graph G consists of
  - One vertex v for each data point $X^1, \ldots, X^M$
  - Edges $e(v_i, v_j)$, whose weight is the "similarity" $w(X^i, X^j)$ between X^i and X^j
  - A common similarity function is the Gaussian

$$w\left(X^i, X^j\right) = \exp\left\{-\frac{\left\|X^i - X^j\right\|^2}{2\sigma^2}\right\}$$

where σ is chosen by the user

Normalized Cut Problem

- To partition vertices v into K groups $C_1,\ldots,C_K$ such that:
  - The sum of edge weights between groups is small
  - The K groups are "balanced" in size

- Formally, we want to find the argmin of

$$Ncut\left(C_1,\ldots,C_K\right) \quad := \quad \frac{1}{2}\sum_{k=1}^{K}\frac{W\left(C_k,\bar{C}_k\right)}{vol\left(C_k\right)}$$

where

$$\begin{aligned} W\left(C_k,\bar{C}_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in \bar{C}_k} w\left(v_i,v_j\right) \\ vol\left(C_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in C_k\setminus\{v_i\}} w\left(v_i,v_j\right) \end{aligned}$$

Normalized Cut Problem

$$
\begin{aligned}
Ncut\left(C_1,\ldots,C_K\right) &:= \frac{1}{2}\sum_{k=1}^{K}\frac{W\left(C_k,\bar{C}_k\right)}{vol\left(C_k\right)} \\
W\left(C_k,\bar{C}_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in\bar{C}_k} w\left(v_i,v_j\right) \\
vol\left(C_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in C_k\backslash\{v_i\}} w\left(v_i,v_j\right)
\end{aligned}
$$

- Ncut() is essentially the ratio W()/vol()

- So minimizing Ncut() implies
  - Making W() small
    - groups should be <u>dissimilar</u>
  - … while making vol() large
    - groups should be <u>large</u> and internally <u>well-connected</u>; this effectively balances their sizes

Normalized Cut Problem

$$
\begin{aligned}
Ncut\left(C_1,\ldots,C_K\right) &:= \frac{1}{2}\sum_{k=1}^{K}\frac{W\left(C_k,\bar{C}_k\right)}{vol\left(C_k\right)} \\
W\left(C_k,\bar{C}_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in\bar{C}_k} w\left(v_i,v_j\right) \\
vol\left(C_k\right) &:= \sum_{v_i\in C_k}\sum_{v_j\in C_k\setminus\{v_i\}} w\left(v_i,v_j\right)
\end{aligned}
$$

- Unfortunately, minimizing Ncut() is NP-hard

- Instead, spectral clustering minimizes a <u>relaxed</u> version of Ncut()

Relaxed Ncut()

- The original Ncut() problem makes “hard” (discrete) assignments of vertices to groups C_k

- We can relax this by allowing fractional assignments to groups
  - e.g. vertex v_i could be 0.5 in C_1 and 0.5 in C_2

Relaxed Ncut()

- The optimal solution to relaxed Ncut() can be recovered from the smallest K eigenvectors of the normalized Laplacian matrix

$$L = \mathbb{I} - D^{-1}W$$

where
  - $\mathbb{I}$ is the identity matrix
  - $W_{ij} = w(v_i,v_j)$ is the weight matrix
  - D is the diagonal "degree matrix": D_{ii} equals the sum over the i-th row of W, while $D_{ij} = 0$ for $i \neq j$

- For details on how L relates to relaxed Ncut(), see "A Tutorial on Spectral Clustering" (Ulrike von Luxberg, 2007)
  - http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf

Eigenvectors of L

$$L \quad = \quad \mathbb{I} - D^{-1}W$$

- What does L look like? What about its eigenvectors?

- First, notice that D^{-1} divides row i of W by outdegree(v_i)
  - Edges outgoing from v_i get normalized by v_i's total outgoing weight

- Then, $L = \mathrm{I} - D^{-1}W$ is a matrix with 1's on the diagonal and negative normalized edge weights everywhere else

Eigenvectors of L

$$L \quad = \quad \mathbb{I} - D^{-1}W$$

- Let's see what happens for a simple 2-cluster graph. Define

$$W \quad = \quad \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

  as the weight matrix of a graph with 2 clusters of 3 vertices each. This implies

$$D \quad = \quad \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \qquad L \quad = \quad \begin{bmatrix} 1 & -.5 & -.5 & 0 & 0 & 0 \\ -.5 & 1 & -.5 & 0 & 0 & 0 \\ -.5 & -.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -.5 & -.5 \\ 0 & 0 & 0 & -.5 & 1 & -.5 \\ 0 & 0 & 0 & -.5 & -.5 & 1 \end{bmatrix}$$

Eigenvectors of L

$$L = \begin{bmatrix} 1 & -.5 & -.5 & 0 & 0 & 0 \\ -.5 & 1 & -.5 & 0 & 0 & 0 \\ -.5 & -.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -.5 & -.5 \\ 0 & 0 & 0 & -.5 & 1 & -.5 \\ 0 & 0 & 0 & -.5 & -.5 & 1 \end{bmatrix}$$

- The smallest 2 eigenvectors of L are

$$e_1 = \begin{bmatrix} 0.577 \\ 0.577 \\ 0.577 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad e_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.577 \\ 0.577 \\ 0.577 \end{bmatrix}$$

  and both their associated eigenvalues are 0
  - In other words, $Le_1 = Le_2 = 0$

- If we plot each row of $[e_1 \ e_2]$ in 2 dimensions, the first 3 rows are clearly separated from the last 3 rows

Eigenvectors of L

$$L = \begin{bmatrix} 1 & -.5 & -.5 & 0 & 0 & 0 \\ -.5 & 1 & -.5 & 0 & 0 & 0 \\ -.5 & -.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -.5 & -.5 \\ 0 & 0 & 0 & -.5 & 1 & -.5 \\ 0 & 0 & 0 & -.5 & -.5 & 1 \end{bmatrix} \qquad e_1 = \begin{bmatrix} 0.577 \\ 0.577 \\ 0.577 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad e_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.577 \\ 0.577 \\ 0.577 \end{bmatrix}$$

- More generally, if L contains K perfect clusters like the above, then these clusters are well-separated by the smallest K eigenvectors

- To get K spectral clusters, form the matrix $A=[e_1,\ldots,e_K]$ whose columns are the smallest K eigenvectors of L, and run k-means using the rows of A as data points

What about imperfect clusters?

- Real datasets don’t have perfectly separated clusters
- But the general intuition still holds
- Let’s take our W from earlier and add a little noise:

$$W = \begin{bmatrix} 0 & .877 & .871 & .012 & .075 & .055 \\ .832 & 0 & .876 & .050 & .026 & .014 \\ .895 & .819 & 0 & .096 & .051 & .015 \\ .003 & .049 & .068 & 0 & .870 & .826 \\ .044 & .045 & .066 & .859 & 0 & .884 \\ .038 & .065 & .016 & .822 & .896 & 0 \end{bmatrix}$$

What about imperfect clusters?

$$W = \begin{bmatrix} 0 & .877 & .871 & .012 & .075 & .055 \\ .832 & 0 & .876 & .050 & .026 & .014 \\ .895 & .819 & 0 & .096 & .051 & .015 \\ .003 & .049 & .068 & 0 & .870 & .826 \\ .044 & .045 & .066 & .859 & 0 & .884 \\ .038 & .065 & .016 & .822 & .896 & 0 \end{bmatrix}$$

- This gives the normalized Laplacian

$$L = \begin{bmatrix} 1 & -.464 & -.461 & -.006 & -.040 & -.023 \\ -.463 & 1 & -.487 & -.028 & -.014 & -.008 \\ -.477 & -.437 & 1 & -.051 & -.027 & -.008 \\ -.002 & -.027 & -.037 & 1 & -.479 & -.455 \\ -.023 & -.024 & -.035 & -.453 & 1 & -.466 \\ -.021 & -.035 & -.009 & -.448 & -.488 & 1 \end{bmatrix}$$

whose smallest two eigenvalues are 0 and 0.141 with eigenvectors

$$e_1 = \begin{bmatrix} .408 \\ .408 \\ .408 \\ .408 \\ .408 \\ .408 \end{bmatrix} \qquad e_2 = \begin{bmatrix} .404 \\ .419 \\ .397 \\ -.412 \\ -.403 \\ -.413 \end{bmatrix}$$

The first 3 rows of [e_1 e_2] are well-separated from the last 3 rows!

Spectral Clustering Summary

1. Form M-by-M weight matrix $W_{ij} = w(X^i, X^j)$
   – w() is a similarity function, e.g. the Gaussian

$$w\left(X^i, X^j\right) = \exp\left\{-\frac{\left\|X^i - X^j\right\|^2}{2\sigma^2}\right\}$$

2. Construct the normalized Laplacian

$$L \quad = \quad \mathbb{I} - D^{-1}W$$

   where $\mathbb{I}$ is the identity matrix, and D is the diagonal degree matrix such that D_{ii} is the sum over the i-th row of W
3. Compute smallest K eigenvectors $e_1,\ldots,e_K$ of L
4. Use k-means to cluster the rows of $A = [e_1,\ldots,e_K]$, where the i-th row of A represents data point X^i