CSC 2515: Probabilistic Classification

Raquel Urtasun & Rich Zemel

University of Toronto

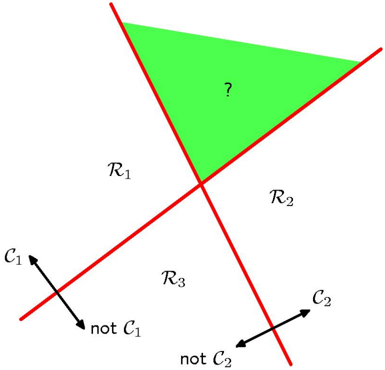
Jan 26, 2015

Today

- Multi-class classification with:
  - Least-squares regression
  - Logistic Regression
  - K-NN
- Classification – Bayes classifier
- Estimate probability densities from data
- Making decisions: Risk
- Classification – Multi-dimensional Bayes classifier
- Naive Bayes

Discriminant Functions for $K > 2$ classes

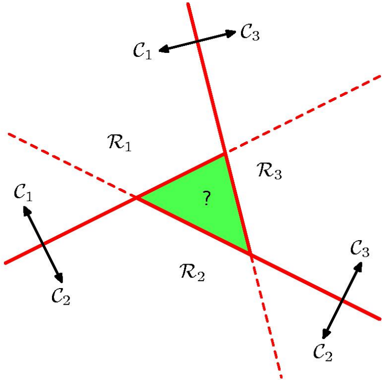
- Use $K - 1$ classifiers, each solving a two class problem of separating point in a class C_k from points not in the class.
- Known as **1 vs all** or **1 vs the rest** classifier



- PROBLEM: More than one good answer!

Discriminant Functions for $K > 2$ classes

- Introduce $K(K-1)/2$ two-way classifiers, one for each possible pair of classes
- Each point is classified according to majority vote amongst the disc. func.
- Known as the **1 vs 1 classifier**



- PROBLEM: Two-way preferences need not be transitive

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

and then assigning a point $\mathbf{x}$ to class C_k if

$$\forall j \neq k \qquad y_k(\mathbf{x}) > y_j(\mathbf{x})$$

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

and then assigning a point $\mathbf{x}$ to class C_k if

$$\forall j \neq k \qquad y_k(\mathbf{x}) > y_j(\mathbf{x})$$

- Note that $\mathbf{w}_k^T$ is now a vector, not the k-th coordinate

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

  and then assigning a point $\mathbf{x}$ to class C_k if

$$\forall j \neq k \qquad y_k(\mathbf{x}) > y_j(\mathbf{x})$$

- Note that $\mathbf{w}_k^T$ is now a vector, not the k-th coordinate
- The decision boundary between class C_j and class C_k is given by $y_j(\mathbf{x}) = y_k(\mathbf{x})$, and thus it's a $(D-1)$ dimensional hyperplane defined as

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

  and then assigning a point $\mathbf{x}$ to class C_k if

$$\forall j \neq k \qquad y_k(\mathbf{x}) > y_j(\mathbf{x})$$

- Note that $\mathbf{w}_k^T$ is now a vector, not the k-th coordinate
- The decision boundary between class C_j and class C_k is given by $y_j(\mathbf{x}) = y_k(\mathbf{x})$, and thus it's a $(D-1)$ dimensional hyperplane defined as

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

- What about the binary case? Is this different?

K-Class Discriminant

- We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}$$

and then assigning a point $\mathbf{x}$ to class C_k if

$$\forall j \neq k \qquad y_k(\mathbf{x}) > y_j(\mathbf{x})$$

- Note that $\mathbf{w}_k^T$ is now a vector, not the k-th coordinate
- The decision boundary between class C_j and class C_k is given by $y_j(\mathbf{x}) = y_k(\mathbf{x})$, and thus it's a $(D-1)$ dimensional hyperplane defined as

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

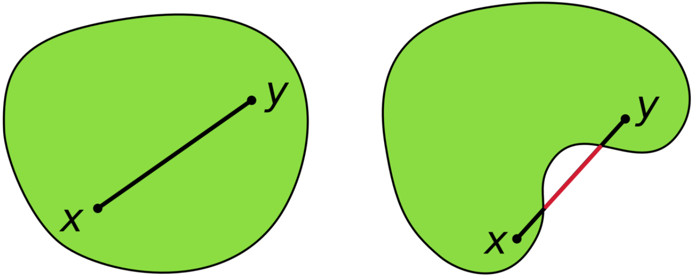
- What about the binary case? Is this different?
- What is the shape of the overall decision boundary?

K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**

K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**
- In Euclidean space, an object is **convex** if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object



- Which object is convex?

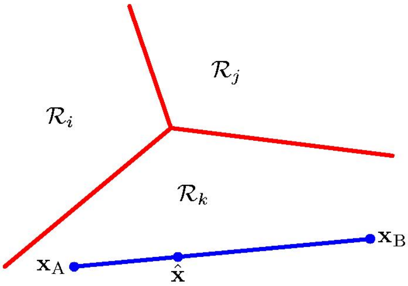
K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**

K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**
- Consider 2 points $\mathbf{x}_A$ and $\mathbf{x}_B$ that lie inside decision region R_k
- Any convex combination $\hat{\mathbf{x}}$ of those points also will be in R_k

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B$$



Multi-class classification via the "softmax"

- Associate a set of weights with each class, then use a normalized exponential output

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where the **activations** are given by

$$z_k = \mathbf{w}_k^T \mathbf{x}$$

Multi-class classification via the "softmax"

- Associate a set of weights with each class, then use a normalized exponential output

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where the **activations** are given by

$$z_k = \mathbf{w}_k^T \mathbf{x}$$

- For the target vector, if there are K classes we often use a 1-of-K encoding, i.e., a vector of K target values containing a single 1 for the correct class and zeros elsewhere

Multi-class classification via the "softmax"

- Associate a set of weights with each class, then use a normalized exponential output

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where the **activations** are given by

$$z_k = \mathbf{w}_k^T \mathbf{x}$$

- For the target vector, if there are K classes we often use a 1-of-K encoding, i.e., a vector of K target values containing a single 1 for the correct class and zeros elsewhere
- Let $\mathbf{T} \in \{0,1\}^{N \times K}$ for N training examples and K classes

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}}$$

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

and

$$z_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

and

$$z_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- What assumptions have I used to derive the likelihood?

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

and

$$z_k = \mathbf{w}_k^T\mathbf{x} + w_{k0}$$

- What assumptions have I used to derive the likelihood?
- Derive the loss by computing the negative log-likelihood

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

and

$$z_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- What assumptions have I used to derive the likelihood?
- Derive the loss by computing the negative log-likelihood

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- This is known as the **cross-entropy** error for multiclass classification

Multi-class Logistic Regression

- The likelihood

$$p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\mathbf{x}^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_k^{(n)}(\mathbf{x}^{(n)})^{t_k^{(n)}}$$

with

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

and

$$z_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- What assumptions have I used to derive the likelihood?
- Derive the loss by computing the negative log-likelihood

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- This is known as the **cross-entropy** error for multiclass classification
- How do we obtain the weights?

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K)$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} =$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} =$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}}$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}$$

$$\frac{\partial E}{\partial w_{k,j}} =$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}$$

$$\frac{\partial E}{\partial w_{k,j}} = \sum_{n=1}^{N}\sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial w_{k,j}} =$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}$$

$$\frac{\partial E}{\partial w_{k,j}} = \sum_{n=1}^{N}\sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial w_{k,j}} = \sum_{n=1}^{N} (y_k^{(n)} - t_k^{(n)}) \cdot x_j^{(n)}$$

Training Multi-class Logistic Regression

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log y_k^{(n)}(\mathbf{x}^{(n)})$$

- Do gradient descent, where the derivatives are

$$\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k,j) y_j^{(n)} - y_j^{(n)} y_k^{(n)}$$

and

$$\frac{\partial E}{\partial z_k}^{(n)} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}$$

$$\frac{\partial E}{\partial w_{k,j}} = \sum_{n=1}^{N}\sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial w_{k,j}} = \sum_{n=1}^{N} (y_k^{(n)} - t_k^{(n)}) \cdot x_j^{(n)}$$

- The derivative is the error times the input

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

- So the logistic is just a special case that avoids using redundant parameters

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

- So the logistic is just a special case that avoids using redundant parameters
- Rather than having two separate set of weights for the two classes, combine into one

$$z' = z_1 - z_2 =$$

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

- So the logistic is just a special case that avoids using redundant parameters
- Rather than having two separate set of weights for the two classes, combine into one

$$z' = z_1 - z_2 = \mathbf{w}_1^T\mathbf{x} - \mathbf{w}_2^T\mathbf{x} =$$

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

- So the logistic is just a special case that avoids using redundant parameters
- Rather than having two separate set of weights for the two classes, combine into one

$$z' = z_1 - z_2 = \mathbf{w}_1^T \mathbf{x} - \mathbf{w}_2^T \mathbf{x} = \mathbf{w}^T \mathbf{x}$$

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|\mathbf{x}) = y_1(\mathbf{x}) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp\left(-(z_1 - z_2)\right)}$$

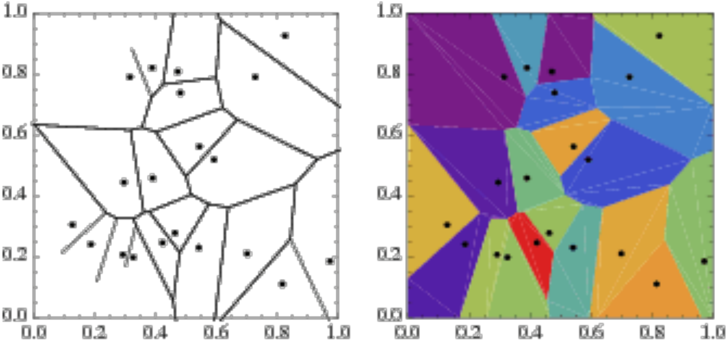
- So the logistic is just a special case that avoids using redundant parameters
- Rather than having two separate set of weights for the two classes, combine into one

$$z' = z_1 - z_2 = \mathbf{w}_1^T\mathbf{x} - \mathbf{w}_2^T\mathbf{x} = \mathbf{w}^T\mathbf{x}$$

- The over-parameterization of the softmax is because the probabilities must add to 1.

Multi-class K-NN

- Can directly handle multi class problems



Generative vs Discriminative

Two approaches to classification:

- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier)
  - Build a model of $p(\mathbf{x}|t_k)$
  - Apply Bayes Rule

Generative vs Discriminative

Two approaches to classification:

- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier)
  - Build a model of $p(\mathbf{x}|t_k)$
  - Apply Bayes Rule
- **Discriminative** classifiers estimate parameters of decision boundary/class separator directly from labeled sample
  - learn boundary parameters directly (logistic regression), or
  - learn mappings from inputs to classes (least-squares, neural nets)

Bayes Classifier

- Aim to diagnose whether patient has diabetes: classify into one of two classes (yes C=1; no C=0)
- Run battery of tests
- Given patient's results: $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$ we want to update class probabilities using Bayes Rule:

$$p(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)p(C)}{p(\mathbf{x})}$$

- More formally

$$\text{posterior} = \frac{\text{Class likelihood} \times \text{prior}}{\text{Evidence}}$$

- How can we compute $p(\mathbf{x})$ for the two class case?

$$p(\mathbf{x}) = p(\mathbf{x}|C=0)p(C=0) + p(\mathbf{x}|C=1)p(C=1)$$

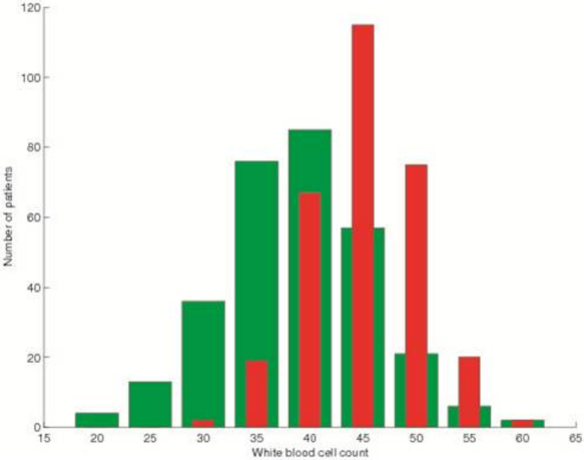
Classification: Diabetes Example

- Start with single input/observation per patient: white blood cell count

$$p(C = 1|x = 50) = \frac{p(x = 50|C = 1)p(C = 1)}{p(x = 50)}$$

- Need class-likelihoods, priors
- Prior: In the absence of any observation, what do I know about the problem?
- What would you use as prior?

Diabetes Data



Which probability distribution makes sense for $p(x|C)$?

MLE for Gaussians

- Let's assume that the class-conditional densities are Gaussian
- How can I fit a Gaussian distribution to my data?

$$p(x|C) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu \in \Re$ and $\sigma^2 \in \Re^+$

MLE for Gaussians

- Let's assume that the class-conditional densities are Gaussian
- How can I fit a Gaussian distribution to my data?

$$p(x|C) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu \in \Re$ and $\sigma^2 \in \Re^+$

- Let's try maximum likelihood estimation (MLE)

MLE for Gaussians

- Let's assume that the class-conditional densities are Gaussian
- How can I fit a Gaussian distribution to my data?

$$p(x|C) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu \in \Re$ and $\sigma^2 \in \Re^+$

- Let's try maximum likelihood estimation (MLE)
- We are given a set of training examples $\{x^{(i)}, t^{(i)}\}_{i=1,\cdots N}$ with $t^{(i)} \in \{0,1\}$ and we want to estimate the model parameters $\{\mu, \sigma\}$ for each class

MLE for Gaussians

- Let's assume that the class-conditional densities are Gaussian
- How can I fit a Gaussian distribution to my data?

$$p(x|C) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu \in \Re$ and $\sigma^2 \in \Re^+$

- Let's try maximum likelihood estimation (MLE)
- We are given a set of training examples $\{x^{(i)}, t^{(i)}\}_{i=1,\cdots N}$ with $t^{(i)} \in \{0, 1\}$ and we want to estimate the model parameters $\{\mu, \sigma\}$ for each class
- First divide the training examples into two classes according to $t^{(i)}$, and for each class take all the examples and fit a Gaussian to model $p(x|C)$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) =$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)$$

- Now we want to maximize the likelihood, or minimize it's negative (if you think in terms of a loss)

$$\ell_{log-loss} \quad = \quad -\ln p(x^{(1)}, \cdots, x^{(N)}|C)$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)$$

- Now we want to maximize the likelihood, or minimize it's negative (if you think in terms of a loss)

$$\ell_{log-loss} \quad = \quad -\ln p(x^{(1)}, \cdots, x^{(N)}|C) = -\ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)\right)$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)$$

- Now we want to maximize the likelihood, or minimize it's negative (if you think in terms of a loss)

$$\begin{aligned} \ell_{log-loss} &= -\ln p(x^{(1)}, \cdots, x^{(N)}|C) = -\ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{N} \ln(\sqrt{2\pi}\sigma) + \sum_{i=1}^{N} \frac{(x^{(i)} - \mu)^2}{2\sigma^2} = \end{aligned}$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)$$

- Now we want to maximize the likelihood, or minimize it's negative (if you think in terms of a loss)

$$\begin{aligned} \ell_{log-loss} &= -\ln p(x^{(1)}, \cdots, x^{(N)}|C) = -\ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{N} \ln(\sqrt{2\pi}\sigma) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} = \frac{N}{2}\ln\left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \end{aligned}$$

MLE for Gaussians II

- We assume that the data points that we have are **independent** and **identically** distributed

$$p(x^{(1)}, \cdots, x^{(N)}|C) = \prod_{i=1}^{N} p(x^{(i)}|C) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)$$

- Now we want to maximize the likelihood, or minimize it's negative (if you think in terms of a loss)

$$\begin{aligned} \ell_{log-loss} &= -\ln p(x^{(1)}, \cdots, x^{(N)}|C) = -\ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{N} \ln(\sqrt{2\pi}\sigma) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} = \frac{N}{2}\ln\left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \end{aligned}$$

- Write $\frac{d\ell_{log-loss}}{d\mu}$ and $\frac{d\ell_{log-loss}}{d\sigma^2}$ and equal it to 0 to find the parameters μ and σ^2

Computing the Mean

$$\frac{\partial \ell_{log-loss}}{\partial \mu} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2 \right) + \sum_{i=1}^{N} \frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right)}{\partial \mu}$$

Computing the Mean

$$\frac{\partial \ell_{log-loss}}{\partial \mu} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^{N} \frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right)}{\partial \mu}$$

Computing the Mean

$$\begin{aligned}\frac{\partial \ell_{log-loss}}{\partial \mu} &= \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} \\ &= \frac{-\sum_{i=1}^{N} 2(x^{(i)} - \mu)}{2\sigma^2}\end{aligned}$$

Computing the Mean

$$\frac{\partial \ell_{log-loss}}{\partial \mu} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2 \right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu}$$

$$= \frac{-\sum_{i=1}^{N} 2(x^{(i)} - \mu)}{2\sigma^2} = -\sum_{i=1}^{N} \frac{(x^{(i)} - \mu)}{\sigma^2}$$

Computing the Mean

$$\begin{aligned}\frac{\partial \ell_{log-loss}}{\partial \mu} &= \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} \\ &= \frac{-\sum_{i=1}^{N} 2(x^{(i)}-\mu)}{2\sigma^2} = -\sum_{i=1}^{N} \frac{(x^{(i)}-\mu)}{\sigma^2} = \frac{N\mu - \sum_{i=1}^{N} x^{(i)}}{\sigma^2}\end{aligned}$$

Computing the Mean

$$\frac{\partial \ell_{log-loss}}{\partial \mu} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \mu}$$

$$= \frac{-\sum_{i=1}^{N} 2(x^{(i)} - \mu)}{2\sigma^2} = -\sum_{i=1}^{N} \frac{(x^{(i)} - \mu)}{\sigma^2} = \frac{N\mu - \sum_{i=1}^{N} x^{(i)}}{\sigma^2}$$

And equating to zero we have

$$\frac{d\ell_{log-loss}}{d\mu} = 0 = \frac{N\mu - \sum_{i=1}^{N} x^{(i)}}{\sigma^2}$$

Computing the Mean

$$\begin{aligned}\frac{\partial \ell_{log-loss}}{\partial \mu} &= \frac{\partial \left(\frac{N}{2}\ln\left(2\pi\sigma^2\right) + \sum_{i=1}^N \frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)}{\partial \mu} = \frac{\partial \left(\sum_{i=1}^N \frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)}{\partial \mu} \\ &= \frac{-\sum_{i=1}^N 2(x^{(i)}-\mu)}{2\sigma^2} = -\sum_{i=1}^N \frac{(x^{(i)}-\mu)}{\sigma^2} = \frac{N\mu - \sum_{i=1}^N x^{(i)}}{\sigma^2}\end{aligned}$$

And equating to zero we have

$$\frac{d\ell_{log-loss}}{d\mu} = 0 = \frac{N\mu - \sum_{i=1}^N x^{(i)}}{\sigma^2}$$

Thus

$$\boxed{\mu = \frac{1}{N}\sum_{i=1}^N x^{(i)}}$$

Computing the Variance

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2 \right) + \sum_{i=1}^{N} \frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right)}{\partial \sigma^2}$$

Computing the Variance

$$
\begin{aligned}
\frac{\partial \ell_{log-loss}}{\partial \sigma^2} &= \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \sigma^2} \\
&= \frac{N}{2}\frac{1}{2\pi\sigma^2} 2\pi + \frac{\sum_{i=1}^{N}(x^{(i)} - \mu)^2}{2} \left(\frac{-1}{\sigma^4}\right)
\end{aligned}
$$

Computing the Variance

$$\begin{aligned}
\frac{\partial \ell_{log-loss}}{\partial \sigma^2} &= \frac{\partial \left(\frac{N}{2} \ln\left(2\pi\sigma^2\right) + \sum_{i=1}^N \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \sigma^2} \\
&= \frac{N}{2} \frac{1}{2\pi\sigma^2} 2\pi + \frac{\sum_{i=1}^N (x^{(i)} - \mu)^2}{2} \left(\frac{-1}{\sigma^4} \right) \\
&= \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^N (x^{(i)} - \mu)^2}{2\sigma^4}
\end{aligned}$$

Computing the Variance

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^N \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \sigma^2}$$

$$= \frac{N}{2} \frac{1}{2\pi\sigma^2} 2\pi + \frac{\sum_{i=1}^N (x^{(i)} - \mu)^2}{2} \left(\frac{-1}{\sigma^4} \right)$$

$$= \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^N (x^{(i)} - \mu)^2}{2\sigma^4}$$

And equating to zero we have

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = 0 = \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^N (x^{(i)} - \mu)^2}{2\sigma^4}$$

Computing the Variance

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^{N} \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \sigma^2}$$

$$= \frac{N}{2} \frac{1}{2\pi\sigma^2} 2\pi + \frac{\sum_{i=1}^{N}(x^{(i)} - \mu)^2}{2} \left(\frac{-1}{\sigma^4} \right)$$

$$= \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^{N}(x^{(i)} - \mu)^2}{2\sigma^4}$$

And equating to zero we have

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = 0 = \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^{N}(x^{(i)} - \mu)^2}{2\sigma^4} = \frac{N\sigma^2 - \sum_{i=1}^{N}(x^{(i)} - \mu)^2}{2\sigma^4}$$

Computing the Variance

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = \frac{\partial \left(\frac{N}{2} \ln \left(2\pi\sigma^2\right) + \sum_{i=1}^N \frac{(x^{(i)}-\mu)^2}{2\sigma^2} \right)}{\partial \sigma^2}$$

$$= \frac{N}{2}\frac{1}{2\pi\sigma^2}2\pi + \frac{\sum_{i=1}^N (x^{(i)}-\mu)^2}{2}\left(\frac{-1}{\sigma^4}\right)$$

$$= \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^N (x^{(i)}-\mu)^2}{2\sigma^4}$$

And equating to zero we have

$$\frac{\partial \ell_{log-loss}}{\partial \sigma^2} = 0 = \frac{N}{2\sigma^2} - \frac{\sum_{i=1}^N (x^{(i)}-\mu)^2}{2\sigma^4} = \frac{N\sigma^2 - \sum_{i=1}^N (x^{(i)}-\mu)^2}{2\sigma^4}$$

Thus

$$\boxed{\sigma^2 = \frac{1}{N}\sum_{i=1}^N (x^{(i)}-\mu)^2}$$

MLE of a Gaussian

- We can compute the parameters in closed form for each class by taking the training points that belong to that class

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu)^2$$

Inference: Posterior Probability

- Now given a new observation and the estimated class-likelihoods and the prior, we can obtain **posterior probability** for class $C = 1$

$$p(C = 1|x) = \frac{p(x|C = 1)p(C = 1)}{p(x)}$$

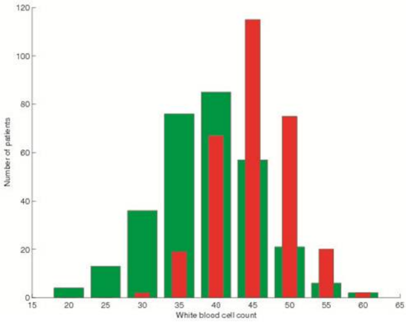
Inference: Posterior Probability

- Now given a new observation and the estimated class-likelihoods and the prior, we can obtain **posterior probability** for class $C = 1$

$$
\begin{aligned}
p(C=1|x) &= \frac{p(x|C=1)p(C=1)}{p(x)} \\
&= \frac{p(x|C=1)p(C=1)}{p(x|C=0)p(C=0) + p(x|C=1)p(C=1)}
\end{aligned}
$$

Diabetes Example

- Doctor has a prior $p(C = 0) = 0.8$, how?
- Example $x = 50$, $p(x = 50|C = 0) = 0.11$, and $p(x = 50|C = 1) = 0.42$
- How were $p(x = 50|C = 0)$ and $p(x = 50|C = 1)$ computed?
- How can I compute $p(C = 1)$?
- Which class is more likely? Do I have diabetes?



Bayes Classifier

- Use Bayes classifier to classify new patients (unseen test examples)
- Simple Bayes classifier: estimate posterior probability of each class
- What should the decision criterion be?

Conditional risk of a classifier

$$R(y|\mathbf{x}) = \sum_{c=1}^{C} L(y,t)p(t=c|x)$$

Conditional risk of a classifier

$$
\begin{aligned}
R(y|\mathbf{x}) &= \sum_{c=1}^{C} L(y,t)p(t=c|x) \\
&= 0 \cdot p(t=y|x) + 1 \cdot \sum_{c \neq y} p(t=c|x)
\end{aligned}
$$

Conditional risk of a classifier

$$\begin{aligned} R(y|\mathbf{x}) &= \sum_{c=1}^{C} L(y,t) p(t=c|x) \\ &= 0 \cdot p(t=y|x) + 1 \cdot \sum_{c \neq y} p(t=c|x) \\ &= \sum_{c \neq y} p(t=c|x) = 1 - p(t=y|x) \end{aligned}$$

Conditional risk of a classifier

$$
\begin{aligned}
R(y|\mathbf{x}) &= \sum_{c=1}^{C} L(y,t)p(t=c|x) \\
&= 0 \cdot p(t=y|x) + 1 \cdot \sum_{c \neq y} p(t=c|x) \\
&= \sum_{c \neq y} p(t=c|x) = 1 - p(t=y|x)
\end{aligned}
$$

- To minimize conditional risk given x, the classifier must decide

$$y = arg \max_c p(t=c|x)$$

Conditional risk of a classifier

$$\begin{aligned} R(y|\mathbf{x}) &= \sum_{c=1}^{C} L(y,t)p(t=c|x) \\ &= 0 \cdot p(t=y|x) + 1 \cdot \sum_{c \neq y} p(t=c|x) \\ &= \sum_{c \neq y} p(t=c|x) = 1 - p(t=y|x) \end{aligned}$$

- To minimize conditional risk given x, the classifier must decide

$$y = arg \max_c p(t=c|x)$$

- This is the best possible classifier in terms of generalization, i.e., expected misclassification rate on new examples.

Log-odds ratio

- Optimal rule $y = arg\max_c p(t = c|x)$ is equivalent to

$$y = c \quad \Leftrightarrow \quad \frac{p(t = c|x)}{p(t = j|x)} \geq 1 \quad \forall j \neq c$$

Log-odds ratio

- Optimal rule $y = arg\max_c p(t = c|x)$ is equivalent to

$$
\begin{aligned}
y = c \quad &\Leftrightarrow \quad \frac{p(t = c|x)}{p(t = j|x)} \geq 1 \quad \forall j \neq c \\
&\Leftrightarrow \quad \log \frac{p(t = c|x)}{p(t = j|x)} \geq 0 \quad \forall j \neq c
\end{aligned}
$$

Log-odds ratio

- Optimal rule $y = arg\max_c p(t = c|x)$ is equivalent to

$$y = c \quad \Leftrightarrow \quad \frac{p(t = c|x)}{p(t = j|x)} \geq 1 \quad \forall j \neq c$$
$$\Leftrightarrow \quad \log \frac{p(t = c|x)}{p(t = j|x)} \geq 0 \quad \forall j \neq c$$

- For the binary case

$$y = 1 \quad \Leftrightarrow \quad \log \frac{p(t = 1|x)}{p(t = 0|x)} \geq 0$$

Log-odds ratio

- Optimal rule $y = arg\max_c p(t = c|x)$ is equivalent to

$$
\begin{aligned}
y = c \quad &\Leftrightarrow \quad \frac{p(t = c|x)}{p(t = j|x)} \geq 1 \quad \forall j \neq c \\
&\Leftrightarrow \quad \log \frac{p(t = c|x)}{p(t = j|x)} \geq 0 \quad \forall j \neq c
\end{aligned}
$$

- For the binary case

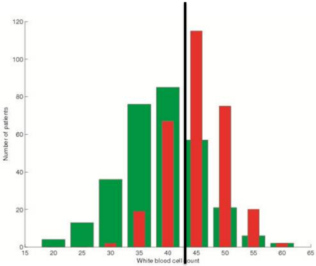
$$
y = 1 \quad \Leftrightarrow \quad \log \frac{p(t = 1|x)}{p(t = 0|x)} \geq 0
$$

- Where have we used this rule before?

Decision Boundary

- The Bayes classifier will construct decision boundary: used to classify new patients (unseen test examples)
- Can be view as a simple linear classifier

$$C = \begin{cases} 1 & \text{if } x \geq T \\ 0 & \text{otherwise} \end{cases}$$



Bayes Classifier

- Aim to diagnose whether patient has diabetes: classify into one of two classes (yes C=1; no C=0)
- Run battery of tests
- Given patient's results: $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$ we want to update class probabilities using Bayes Rule:

$$p(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)p(C)}{p(\mathbf{x})}$$

- More formally

$$\text{posterior} = \frac{\text{Class likelihood} \times \text{prior}}{\text{Evidence}}$$

- How can we compute $p(\mathbf{x})$ for the two class case?

$$p(\mathbf{x}) = p(\mathbf{x}|C=0)p(C=0) + p(\mathbf{x}|C=1)p(C=1)$$

Classification: Diabetes Example

- Before we had a single input/observation per patient: white blood cell count

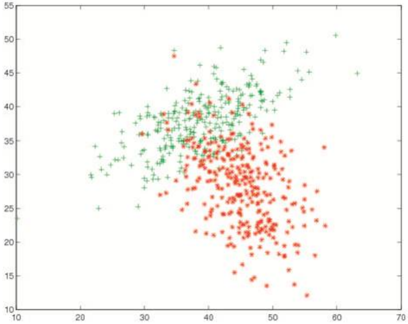
$$p(C = 1|x = 50) = \frac{p(x = 50|C = 1)p(C = 1)}{p(x = 50)}$$

Classification: Diabetes Example

- Before we had a single input/observation per patient: white blood cell count

$$p(C = 1|x = 50) = \frac{p(x = 50|C = 1)p(C = 1)}{p(x = 50)}$$

- Add second observation: Plasma glucose value
- Can construct bivariate normal (Gaussian) distribution of each class



Gaussian Bayes Classifier

- Gaussian (or normal) distribution:

$$p(\mathbf{x}|t=k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)\right]$$

Gaussian Bayes Classifier

- Gaussian (or normal) distribution:

$$p(\mathbf{x}|t = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-(\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k)\right]$$

- Each class k has associated mean vector, but typically the classes share a single covariance matrix

Multivariate Data

- Multiple measurements (sensors)
- d inputs/features/attributes
- N instances/observations/examples

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$$

Multivariate Parameters

- Mean

$$\mathbb{E}[\mathbf{x}] = [\mu_1, \cdots, \mu_d]^T$$

Multivariate Parameters

- Mean

$$\mathbb{E}[\mathbf{x}] = [\mu_1, \cdots, \mu_d]^T$$

- Covariance

$$\Sigma = Cov(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \mu)^T(\mathbf{x} - \mu)] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

Multivariate Parameters

- Mean

$$\mathbb{E}[\mathbf{x}] = [\mu_1, \cdots, \mu_d]^T$$

- Covariance

$$\Sigma = Cov(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \mu)^T(\mathbf{x} - \mu)] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

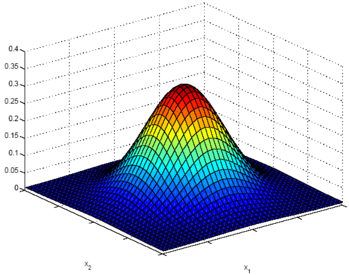
- Correlation $= Corr(\mathbf{x})$ is the covariance divided by the product of standard deviation

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

Multivariate Gaussian Distribution

- $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$, a Gaussian (or normal) distribution defined as

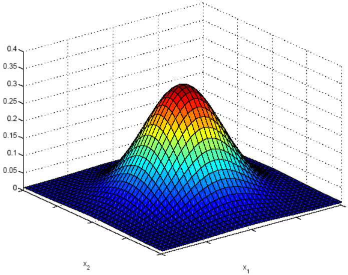
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)\right]$$



Multivariate Gaussian Distribution

- $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$, a Gaussian (or normal) distribution defined as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-(\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k)\right]$$

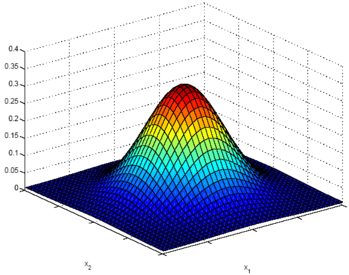


- Mahalanobis distance $(\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k)$ measures the distance from $\mathbf{x}$ to μ in terms of Σ

Multivariate Gaussian Distribution

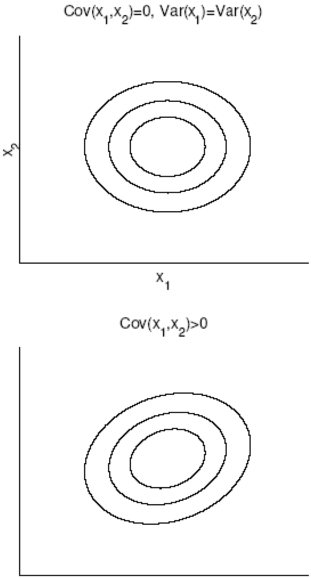
- $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$, a Gaussian (or normal) distribution defined as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)\right]$$

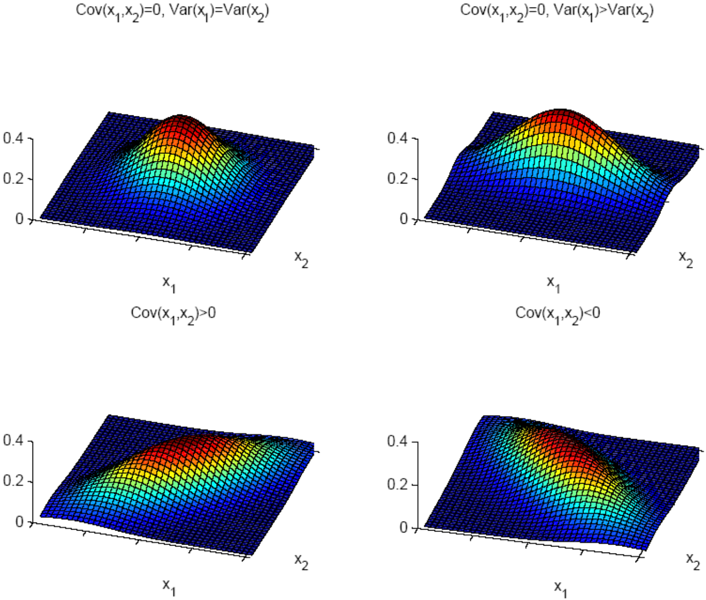


- Mahalanobis distance $(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)$ measures the distance from $\mathbf{x}$ to μ in terms of Σ
- It normalizes for difference in variances and correlations

Bivariate Normal



Bivariate Normal



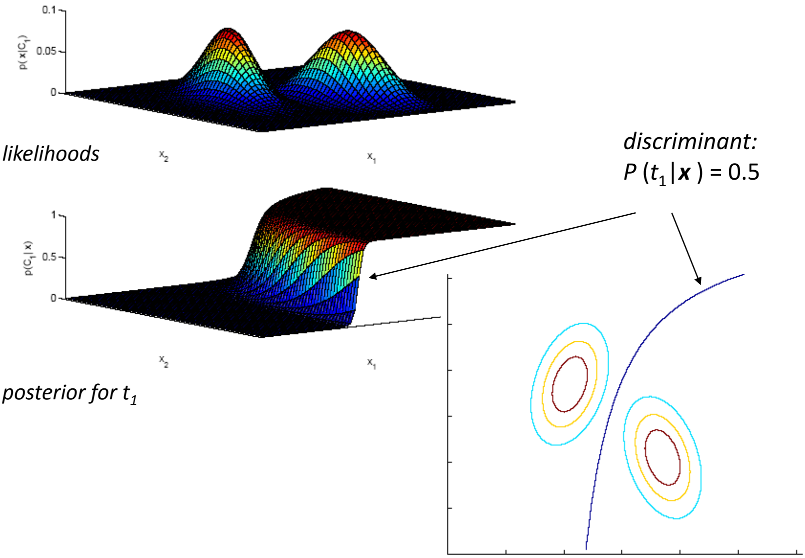
Gaussian Bayes Classifier Decision Boundary

- GBC decision boundary: based on class posterior
- Take the class which has higher posterior probability

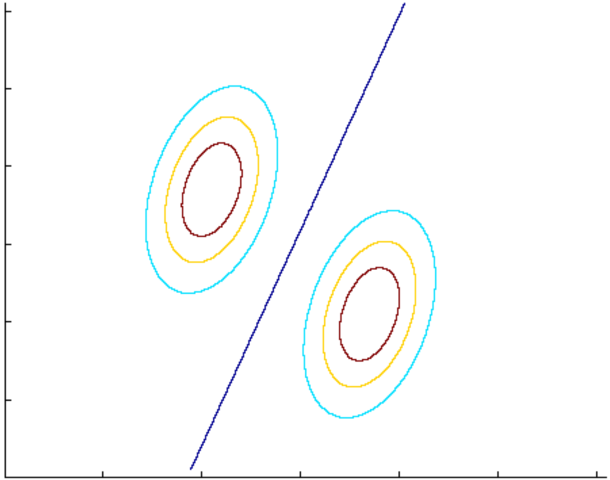
$$
\begin{aligned}
\log p(t_k|\mathbf{x}) &= \log p(\mathbf{x}|t_k) + \log p(t_k) - \log p(\mathbf{x}) \\
&= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_k^{-1}| - \frac{1}{2}(\mathbf{x}-\mu_k)^T\sigma_k^{-1}(\mathbf{x}-\mu_k) + \\
&\quad + \log p(t_k) - \log p(\mathbf{x})
\end{aligned}
$$

- Decision: which class has higher posterior probability

Decision Boundary



Shared Covariance Matrix



Learning Gaussian Bayes Classifier

- Learn the parameters using maximum likelihood

$$
\begin{aligned}
\ell(\phi, \mu_0, \mu_1, \Sigma) &= -\log \prod_{n=1}^{N} p(\mathbf{x}^{(n)}, t^{(n)} | \phi, \mu_0, \mu_1, \Sigma) \\
&= -\log \prod_{n=1}^{N} p(\mathbf{x}^{(n)} | t^{(n)}, \mu_0, \mu_1, \Sigma) p(t^{(n)} | \phi)
\end{aligned}
$$

- What have I assumed?

More on MLE

- Assume the prior is Bernoulli (we have two classes)

$$p(t|\phi) = \phi^t (1-\phi)^{1-t}$$

- You can compute the ML estimate in closed form

$$\begin{aligned}
\phi &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 1] \\
\mu_0 &= \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 0] \cdot \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 0]} \\
\mu_1 &= \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 1] \cdot \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 1]} \\
\Sigma &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \mu_{t^{(n)}})(\mathbf{x}^{(n)} - \mu_{t^{(n)}})^T
\end{aligned}$$

Naive Bayes

- For Gaussian Bayes Classifier, if input $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters

Naive Bayes

- For Gaussian Bayes Classifier, if input $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters
- Save some parameters by using a shared covariance for the classes

Naive Bayes

- For Gaussian Bayes Classifier, if input $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters
- Save some parameters by using a shared covariance for the classes
- Naive Bayes is an alternative Generative model: assumes features independent given the class

$$p(\mathbf{x}|t = k) = \prod_{i=1}^{d} p(x_i|t = k)$$

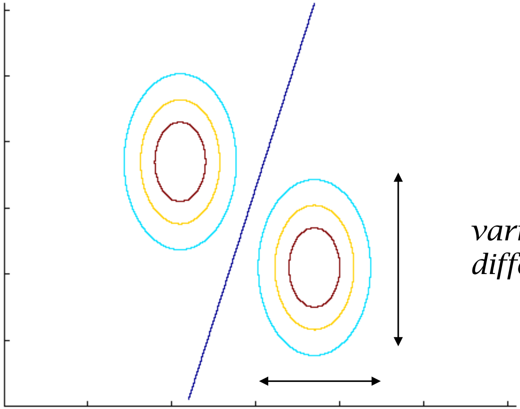
Naive Bayes

- For Gaussian Bayes Classifier, if input $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters
- Save some parameters by using a shared covariance for the classes
- Naive Bayes is an alternative Generative model: assumes features independent given the class

$$p(\mathbf{x}|t = k) = \prod_{i=1}^{d} p(x_i|t = k)$$

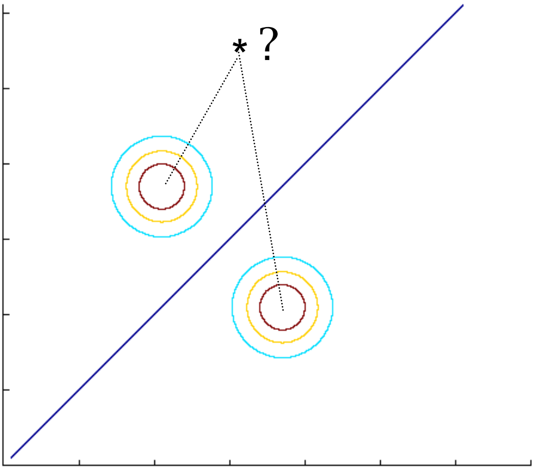
- How many parameters required now? And before?

Diagonal Covariance



variances may be different

Diagonal Covariance, isotropic



- Classification only depends on distance to the mean

Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each x_i

Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each x_i

The decision rule

$$y = arg \max_k p(t = k) \prod_{i=1}^{d} p(x_i | t = k)$$

Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each x_i

The decision rule

$$y = arg \max_k p(t = k) \prod_{i=1}^{d} p(x_i | t = k)$$

- If the assumption of conditional independence holds, NB is the optimal classifier

Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each x_i

The decision rule

$$y = arg \max_k p(t = k) \prod_{i=1}^{d} p(x_i | t = k)$$

- If the assumption of conditional independence holds, NB is the optimal classifier
- If not, a heavily regularized version of generative classifier

Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each x_i

The decision rule

$$y = arg \max_k p(t = k) \prod_{i=1}^{d} p(x_i | t = k)$$

- If the assumption of conditional independence holds, NB is the optimal classifier
- If not, a heavily regularized version of generative classifier
- What's the regularization?

Gaussian Naive Bayes

- Assume

$$p(x_i|t=k) = \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left[\frac{-(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right]$$

Gaussian Naive Bayes

- Assume

$$p(x_i|t=k) = \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left[\frac{-(x_i-\mu_{ik})^2}{2\sigma_{ik}^2}\right]$$

- Maximum likelihood estimate of parameters

$$\mu_{ik} = \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k]\cdot x_i^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k]}$$

Gaussian Naive Bayes

- Assume

$$p(x_i|t = k) = \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp\left[\frac{-(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right]$$

- Maximum likelihood estimate of parameters

$$\mu_{ik} = \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k] \cdot x_i^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k]}$$

- Similar for the variance

Gaussian Bayes Classifier (GBC) vs Logistic Regression

- If you examine $p(t = 1|\mathbf{x})$ under GBC, you will find that it looks like this:

$$p(t|\mathbf{x}, \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\mathbf{w}(\phi, \mu_0, \mu_1, \Sigma)^T \mathbf{x})}$$

Gaussian Bayes Classifier (GBC) vs Logistic Regression

- If you examine $p(t = 1|\mathbf{x})$ under GBC, you will find that it looks like this:

$$p(t|\mathbf{x}, \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\mathbf{w}(\phi, \mu_0, \mu_1, \Sigma)^T \mathbf{x})}$$

- So the decision boundary has the same form as logistic regression!

Gaussian Bayes Classifier (GBC) vs Logistic Regression

- If you examine $p(t = 1|\mathbf{x})$ under GBC, you will find that it looks like this:

$$p(t|\mathbf{x}, \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\mathbf{w}(\phi, \mu_0, \mu_1, \Sigma)^T \mathbf{x})}$$

- So the decision boundary has the same form as logistic regression!
- When should we prefer GBC to LR, and vice versa?

GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian

GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GBC is asymptotically efficient (best model in limit of large N)

GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GBC is asymptotically efficient (best model in limit of large N)
- But LR is more robust, less sensitive to incorrect modeling assumptions

GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GBC is asymptotically efficient (best model in limit of large N)
- But LR is more robust, less sensitive to incorrect modeling assumptions
- Many class-conditional distributions lead to logistic classifier

GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GBC is asymptotically efficient (best model in limit of large N)
- But LR is more robust, less sensitive to incorrect modeling assumptions
- Many class-conditional distributions lead to logistic classifier
- When these distributions are non-Gaussian, in limit of large N, LR beats GBC

Spam Filter Example

- Naive Bayes also applies to discrete input features (or mixed discrete/continuous)

Spam Filter Example

- Naive Bayes also applies to discrete input features (or mixed discrete/continuous)
- Represent email as feature vector, length equals number of words in vocabulary, binary feature x_i is 1 iff the word i appears in email msg

Spam Filter Example

- Naive Bayes also applies to discrete input features (or mixed discrete/continuous)
- Represent email as feature vector, length equals number of words in vocabulary, binary feature x_i is 1 iff the word i appears in email msg
- Each of these binary conditional probabilities is Bernoulli, with parameter ϕ_i

Spam Filter Example

- Naive Bayes also applies to discrete input features (or mixed discrete/continuous)
- Represent email as feature vector, length equals number of words in vocabulary, binary feature x_i is 1 iff the word i appears in email msg
- Each of these binary conditional probabilities is Bernoulli, with parameter ϕ_i
- When we estimate parameters by maximizing joint likelihood of data, get sensible updates: $\phi_{i|t=1}$ is fraction of the spam emails in which word i appears

Laplace Smoothing

- What happens when some word appears in the test set but never in the training set?

Laplace Smoothing

- What happens when some word appears in the test set but never in the training set?
- Counts = 0, so $\phi_{i|t=1} = \phi_{i|t=0} = 0$

Laplace Smoothing

- What happens when some word appears in the test set but never in the training set?
- Counts $= 0$, so $\phi_{i|t=1} = \phi_{i|t=0} = 0$
- Class posterior probabilities $= 0/0$

Laplace Smoothing

- What happens when some word appears in the test set but never in the training set?
- Counts = 0, so $\phi_{i|t=1} = \phi_{i|t=0} = 0$
- Class posterior probabilities = 0/0
- Instead use this parameter estimate:

$$\phi_{i|t=1} = \frac{1}{N} \sum_{n=1}^{N} \frac{\mathbb{1}[t^{(n)} = 1 \wedge x_i^{(n)} = 1]}{\mathbb{1}[t^{(n)} = 1] + \alpha K}$$

- K is number of classes, parameter α acts like "pseudo-count": prior observations of words