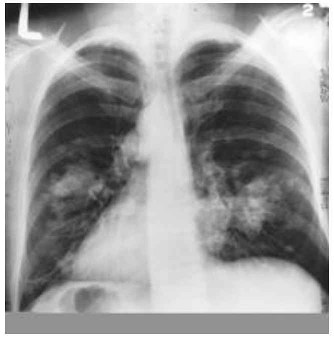
Linear Classification

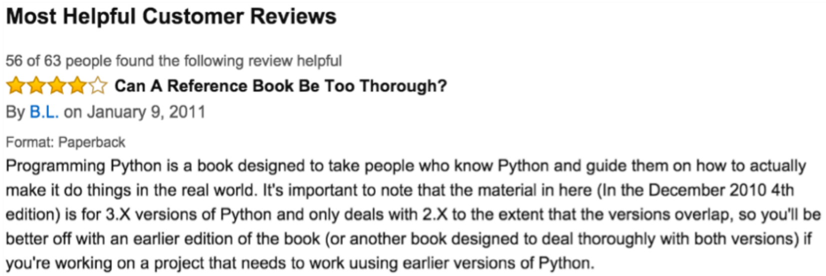
Jian Tang

tangjianpku@gmail.com

Classification

- Assign an input real-valued vector x into K discrete classes $\{C_k\}_{k=1,\ldots,K}$





Most Helpful Customer Reviews

56 of 63 people found the following review helpful

Can A Reference Book Be Too Thorough?

By B.L. on January 9, 2011

Format: Paperback

Programming Python is a book designed to take people who know Python and guide them on how to actually make it do things in the real world. It's important to note that the material in here (In the December 2010 4th edition) is for 3.X versions of Python and only deals with 2.X to the extent that the versions overlap, so you'll be better off with an earlier edition of the book (or another book designed to deal thoroughly with both versions) if you're working on a project that needs to work uusing earlier versions of Python.

X: set of pixel intensities
Y: cancer present/cancer absent

X: user reviews
Y: positive/neural/negative

Linear Classification

- Goal: Assign an input real-valued vector x into K discrete classes $\{C_k\}_{k=1,\dots,K}$
- The input space is divided into different decision regions whose boundaries are called decision boundaries or decision surfaces.
- Linear classification: the model is linear w.r.t. the parameters

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{x}^T\mathbf{w} + w_0. \qquad y(\mathbf{x}, \mathbf{w}) = f(\mathbf{x}^T\mathbf{w} + w_0).$$

adaptive parameters

fixed nonlinear function: activation function

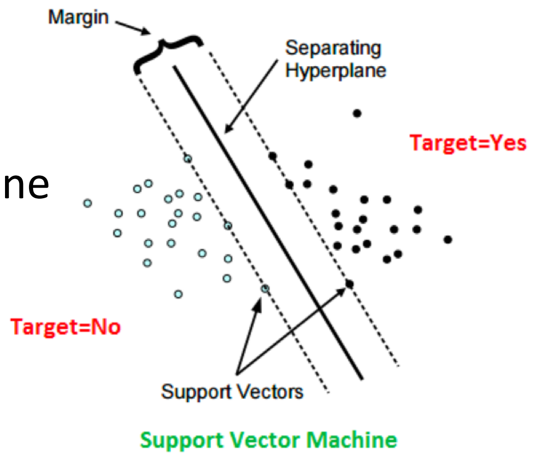
- For classification, we need to predict discrete classes, or posterior probabilities that lie in the range of (0,1), and therefore a nonlinear function f is used.

Linear Classification

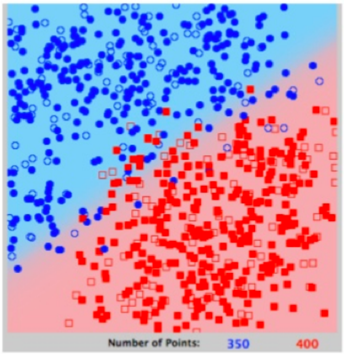
$$y(\mathbf{x}, \mathbf{w}) = f(\mathbf{x}^T \mathbf{w} + w_0).$$

- The decision boundary : $y(\mathbf{x}, \mathbf{w}) = \text{const}$, i.e., $\mathbf{x}^T \mathbf{w} + w_0 = \text{const}$,
  - The decision boundary are linear functions of x
  - Even if f is a nonlinear function
- Note: these models are not linear w.r.t. the parameters any more

Support vector machine



Decision surface is linear



Logistic regression

Notation

- Binary Classification: target $t \in \{0,1\}$, t=1 represents the positive class and t=0 represents the negative class
- Multi-class classification: one-hot encoding
- E.g., if there are K=5 classes, an input belonging to the second class can be encoded as

$$t = (0, 1, 0, 0, 0)^T.$$

- Which can be interpreted as the probabilities belonging to each class

Three Approaches for Classification

- Construct a **discriminant function** that directly maps an input to a class (e.g., support vector machine)
- Model the conditional distribution $p(\mathcal{C}_k|\mathbf{x})$,
- Two alternative approaches
  - **Discriminative model:** directly model the conditional probability $p(\mathcal{C}_k|\mathbf{x})$, (e.g., logistic regression)
  - **Generative model:** model the joint probability $p(\mathrm{x}, \mathcal{C}_k)$. The conditional probability $p(\mathcal{C}_k|\mathbf{x})$, can be calculated as:

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

(e.g. Naïve Bayes).

Outline: Linear Classification

- **Discriminant Function**
- Generative Models
- Discriminative Models

Discriminant Functions

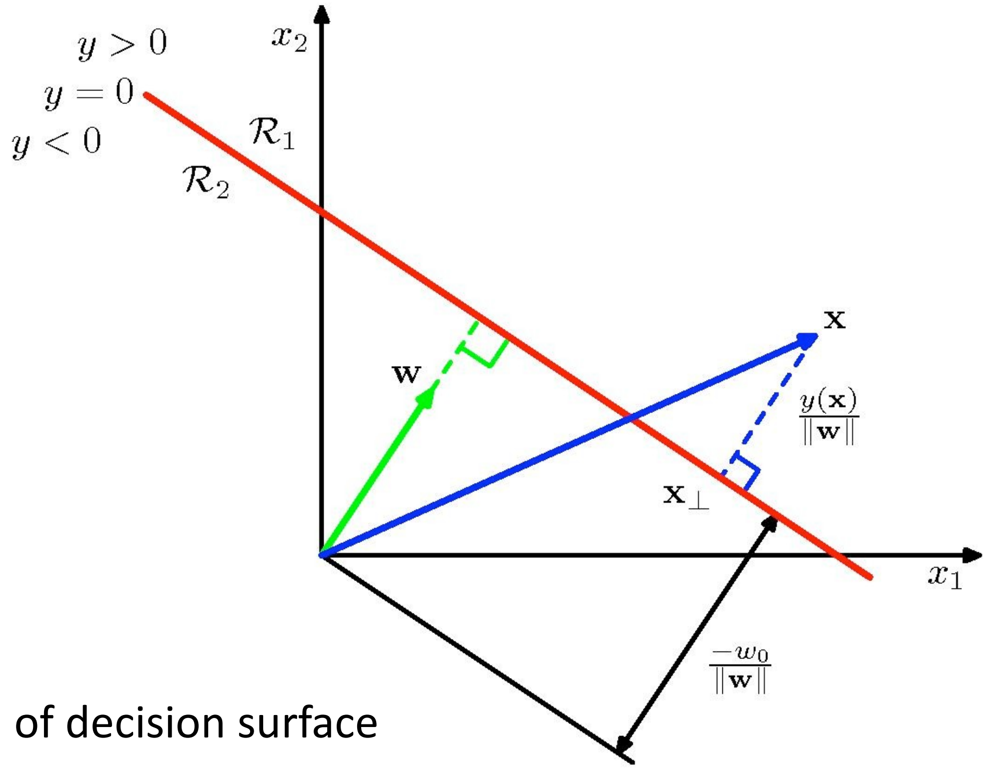
- Consider $y(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + w_0.$
- Assign x to C1 if $y(\mathbf{x}) \geq 0,$ and class C2 otherwise
- Decision boundary: $y(\mathbf{x}) = 0.$
- If two points x_A and x_B lie on the same

decision surface: $$y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0,$$ $$\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0.$$

- w is orthogonal to the decision surface
- If x is on the decision surface

$$\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||} = -\frac{w_0}{||\mathbf{w}||}.$$

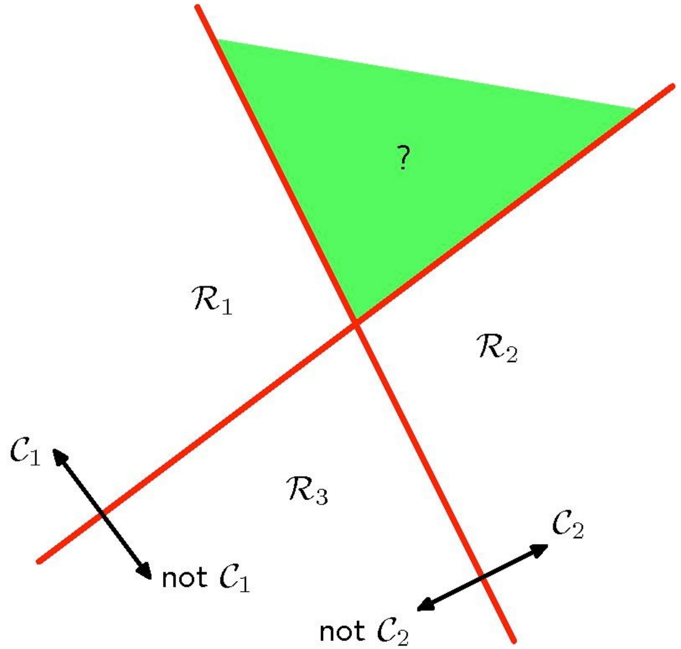
w_0 determines the location of decision surface



Multiple Classes

- How to extend K>2 classes
- One option is to use K-1 classifiers, each of which solves a two-class problem:
  - Separates class $\mathcal{C}_k$ from the rest of the classes
- There are regions in the input space that are ambiguously classified

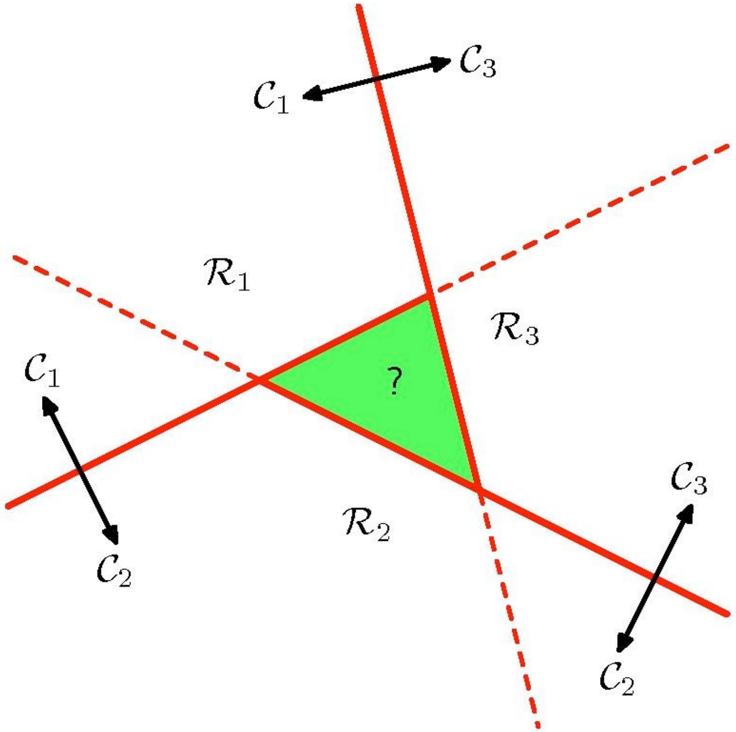
One-versus-the-rest



Multiple Classes

- An alternative solution is to use K(K-1)/2 binary discriminant functions
    - Each function discriminates two classes
- Similar problem of ambiguous regions

One-versus-one



Simple Solution

- Use K discriminant functions of the form:

$$y_k(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_k + w_{k0}, \text{ where } k = 1, ..., K.$$

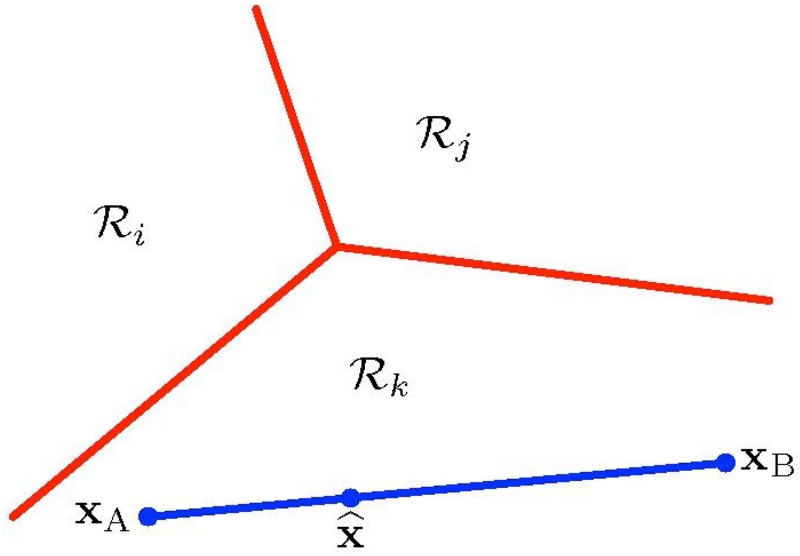
- Assign x to $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x}) \ \forall j \neq k$ (pick the max)
- Can guarantee to give decision boundaries that are singly connected and convex
- For any two points that lie inside region $\mathcal{R}_k$

$$y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A) \text{ and } y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$$

implies that

$$y_k(\alpha\mathbf{x}_A + (1-\alpha)\mathbf{x}_B) > y_j(\alpha\mathbf{x}_A + (1-\alpha)\mathbf{x}_B)$$

due to linearity of the discriminant functions



The Perceptron Algorithm

- Another example of a linear discriminant function
- Consider the following generalized linear model:

$$y(\mathbf{x}) = f\left(\mathbf{w}^T \phi(\mathbf{x})\right)$$

- Where nonlinear function f(.) is given by a step function

$$f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

- and x is transformed using a fixed nonlinear function $\phi(\mathbf{x})$
- Hence we have a two-class model

The Perceptron Algorithm

- A natural choice of error function would be the total number of misclassified examples (but hard to optimize, discontinuous)
- Consider an alternative error function:
- First, note that
  - Patterns x_n in class $\mathcal{C}_1$ should satisfy that：

$$\mathbf{w}^T\phi(\mathbf{x}_n) > 0$$

  - Patterns x_n in class $\mathcal{C}_2$ should satisfy that：

$$\mathbf{w}^T\phi(\mathbf{x}_n) < 0$$

- Using the target coding $t \in \{-1,1\}$, we see that we would like all patterns to satisfy:

$$\mathbf{w}^T\phi(\mathbf{x}_n)t_n > 0$$

Error Function

- Using the target coding $t \in \{-1,1\}$, we see that we would like all patterns to satisfy:

$$\mathbf{w}^T\phi(\mathbf{x}_n)t_n > 0$$

- The error function is therefore given by :

$$E_P(\mathbf{w}) = -\sum_{n\in M} \mathbf{w}^T\phi(\mathbf{x}_n)t_n$$

M denotes all misclassified examples.

- The error function is linear w.r.t. w in regions of w space where the example is misclassified and 0 in regions where it is correctly classified.
- The error function is piece-wise linear

Error Function

- We can use stochastic gradient descent. Given a misclassified example, the change of weight is:

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta \bigtriangledown E_p(\mathbf{w}) = \mathbf{w}^{t} + \eta\phi(\mathbf{x}_n)t_n,$$

η is the learning rate

- Since the perceptron function $y(\mathbf{x}) = f(\mathbf{w}^T\phi(\mathbf{x}))$ is unchanged if we multiple w by a constant, we set ||w||=1
- The contribution to the error function from the misclassified example will be reduced

$$\begin{aligned} -\mathbf{w}^{(t+t)T}\phi(\mathbf{x}_n)t_n &= -\mathbf{w}^{(t)T}\phi(\mathbf{x}_n)t_n - (\phi(\mathbf{x}_n)t_n)^T(\phi)(\mathbf{x}_n)t_n) \\ &< -\mathbf{w}^{(t)T}\phi(\mathbf{x}_n)t_n \end{aligned}$$

Always positive

Error Function

- Note that the contribution to the error function from the misclassified example will be reduced:

$$
\begin{aligned}
-\mathbf{w}^{(t+t)T}\phi(\mathbf{x}_n)t_n &= -\mathbf{w}^{(t)T}\phi(\mathbf{x}_n)t_n - (\phi(\mathbf{x}_n)t_n)^T(\phi)(\mathbf{x}_n)t_n) \\
&< -\mathbf{w}^{(t)T}\phi(\mathbf{x}_n)t_n
\end{aligned}
$$

Always positive

- However, the change in w may cause some previously correctly classified examples to be misclassified. **No convergence guarantees** in general.

Outline: Linear Classification

- Discriminant Function
- Generative Models
- Discriminative Models

Probabilistic Generative Models

- Model class conditional probability $\mathrm{p}(\mathrm{x}|\mathcal{C}_k)$ and class prior $\mathrm{p}(\mathcal{C}_k)$ separately (e.g., Naïve Bayes)
- Take the binary classification as an example, the posterior probability of class $\mathcal{C}_1$

$$
\begin{aligned}
p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
&= \frac{1}{1 + \exp(-a)} = \sigma(a),
\end{aligned}
$$

Logistic sigmoid function

$$
a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \ln \frac{p(\mathcal{C}_1|\mathbf{x})}{1 - p(\mathcal{C}_1|\mathbf{x})},
$$

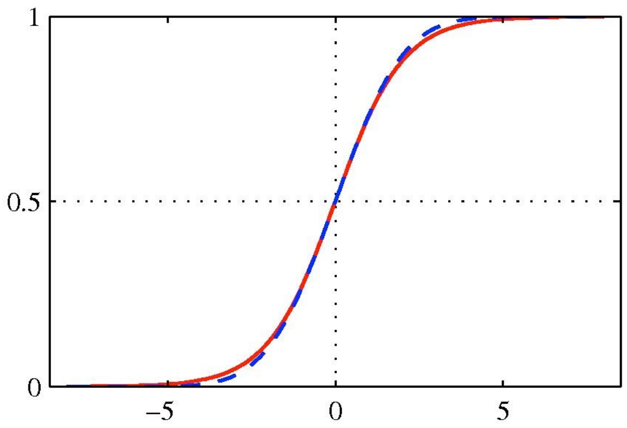
- a is known as the **logit function**, which represents the log or the ration of probabilities of two classes, as known as the **log-odds.**

Sigmoid Function

- The posterior probability of class $\mathcal{C}_1$:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$
$$= \frac{1}{1+\exp(-a)} = \sigma(a),$$

Sigmoid function



- The term sigmoid maps the real space to (0,1), and satisfies:

$$\sigma(-a) = 1 - \sigma(a), \ \frac{\mathrm{d}}{\mathrm{d}a}\sigma(a) = \sigma(a)(1-\sigma(a)).$$

Softmax Function

- For K>2 classes, we generalize the sigmoid function to the **softmax**:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \quad a_k = \ln[p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)].$$

- Softmax function represents a smoothed version of max function

$$\text{if } a_k \gg a_j, \forall j \neq k, \ \text{ then } p(\mathcal{C}_k|\mathbf{x}) \approx 1, \ p(\mathcal{C}_j|\mathbf{x}) \approx 0.$$

Example of Continuous Inputs

- Assuming that the input vectors for reach class are from a Gaussian distribution, and all classes share the same covariance matrix：

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right).$$

- For binary classification, the posterior is the logistic function：

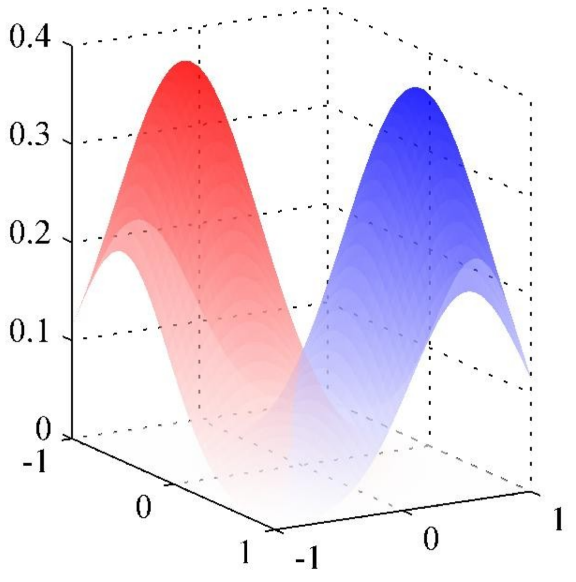
$$p(\mathcal{C}_k|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0),$$

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),$$

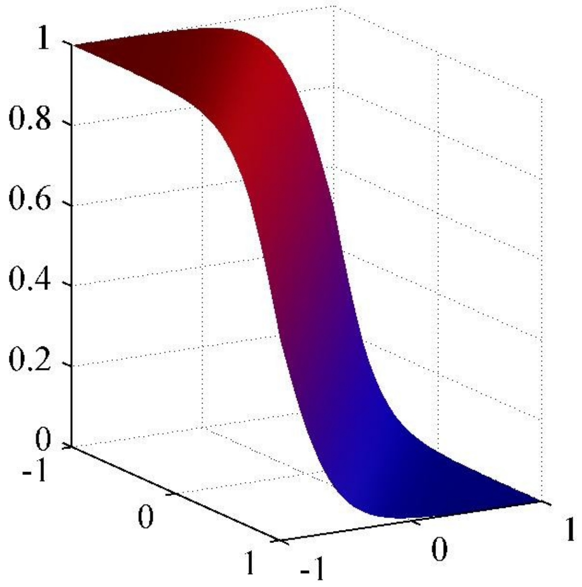
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}.$$

- The quadratic terms in x is cancelled (the same covariance matrix)
- This leads to a linear function of x in the argument of logistic sigmoid. Hence the decision boundaries are linear in input space.

Example of Two Gaussian Models



Class-conditional densities for two classes



The corresponding posterior probability $p(\mathcal{C}_1|\mathbf{x})$, given by the sigmoid function of a linear function of **x**.

Case of K>2 Classes

- For the case of K classes, the posterior is a softmax function:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)},$$

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

- Similar to binary classification, we define:

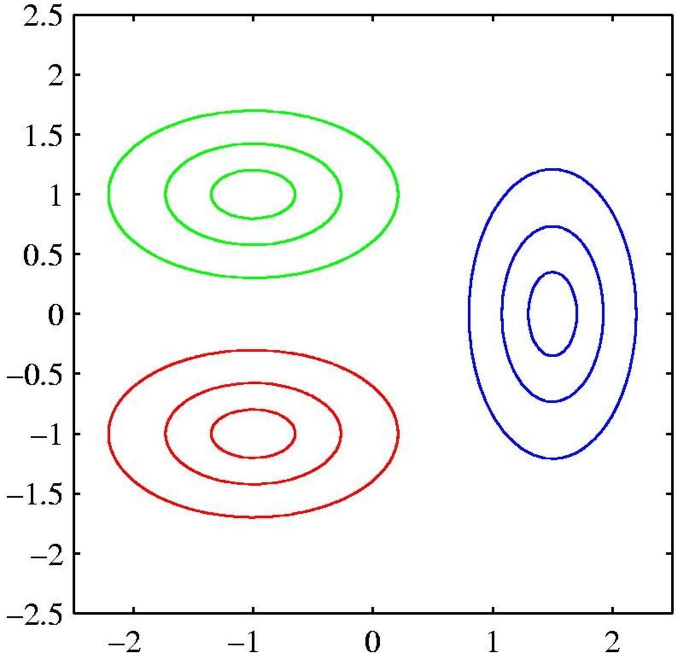
$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k,$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k).$$

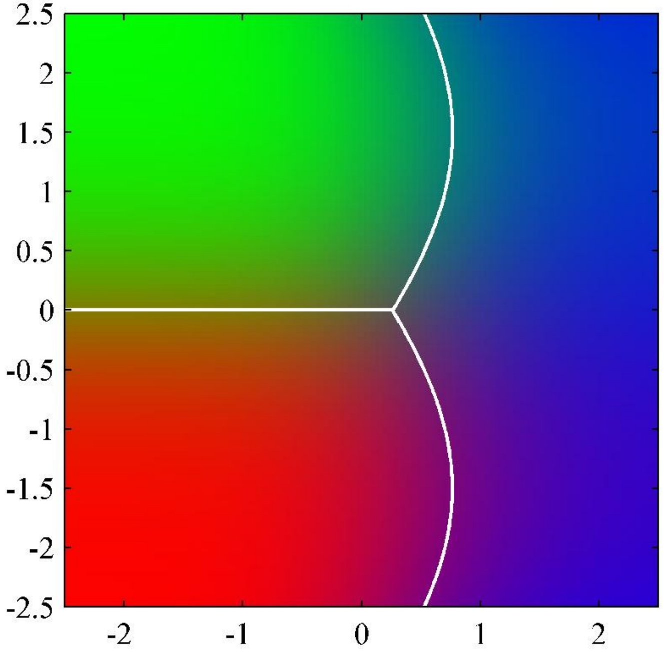
- Again, the decision boundaries are linear in input space.
- If we allow each class-conditional density to have its own covariance, we will obtain quadratic function of x (quadratic discriminant).

Quadratic Discriminant

- The decision boundary is linear when the covariance matrices are the same and quadratic when they are not.



Class-conditional densities for three classes



The corresponding posterior probabilities for three classes.

Maximum Likelihood Solution

- Take the binary classification as an example, each having a Gaussian class-conditional density with the same covariance matrix
- We observe a dataset: $\{\mathbf{x}_n, t_n\},\ n = 1, .., N.$
  - tn=1 denotes class C1, tn=0 denotes class C2
  - And also $p(\mathcal{C}_1) = \pi,\ p(\mathcal{C}_2) = 1 - \pi.$
- The likelihood function：

$$p(\mathbf{t}, \mathbf{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\underbrace{\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})}_{\text{Data points from class } C_1.}\right]^{t_n} \left[\underbrace{(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})}_{\text{Data points from class } C_2.}\right]^{1-t_n}.$$

- Maximize the likelihood function

Maximum Likelihood Solution

$$p(\mathbf{t}, \mathbf{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}.$$

- Maximize w.r.t. π. The terms of the log-likelihood functions depends on π:

$$\sum_n \left[t_n \ln \pi + (1-t_n)\ln(1-\pi)\right] + \text{const.}$$

Differentiating, we have

$$\pi = \frac{1}{N}\sum_{n=1}^{N} t_n = \frac{N_1}{N_1+N_2}.$$

- Maximize w.r.t. μ_1: the terms depending on μ_1:

$$\sum_n t_n \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2}\sum_n t_n(\mathbf{x}_n - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.}$$

Differentiating, we get:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n.$$

And similarly:

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1-t_n)\mathbf{x}_n.$$

Maximum Likelihood Solution

$$p(\mathbf{t}, \mathbf{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}.$$

- Maximize w.r.t. Σ:

$$\begin{aligned}
&-\frac{1}{2}\sum_n t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_n t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1) \\
&-\frac{1}{2}\sum_n (1-t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_n (1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_2) \\
&= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{Tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}).
\end{aligned}$$

- Here:

$$\begin{aligned}
\mathbf{S} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2, \\
\mathbf{S}_1 &= \frac{1}{N_1}\sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \\
\mathbf{S}_2 &= \frac{1}{N_2}\sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T.
\end{aligned}$$

•Using standard results for a Gaussian distribution we have:

$$\boldsymbol{\Sigma} = \mathbf{S}.$$

•Maximum likelihood solution represents a weighted average of the covariance matrices associated with each of the two classes.

Outline: Linear Classification

- Discriminant Function
- Generative Models
- Discriminative Models

Logistic Regression

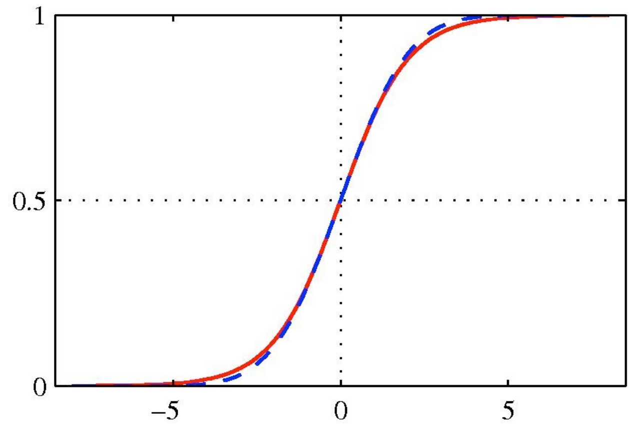
- For binary classification, the posterior probability of class $\mathcal{C}_1$ can be written as sigmoid function

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})} = \sigma(\mathbf{w}^T\mathbf{x}),$$

- and $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$, and we omit the bias term for clarity.
- This model is known as logistic regression (although this is a model for classification rather than regression).

Note that for generative models, we would first determine the class conditional densities and class-specific priors, and then use Bayes' rule to obtain the posterior probabilities.

Here we model $p(\mathcal{C}_k|\mathbf{x})$ directly.



logistic sigmoid function

ML for Logistic Regression

- We observed a training dataset $\{\mathbf{x}_n, t_n\},\ n = 1, .., N;\ t_n \in \{0, 1\}.$
- Maximize the probability of getting the label right, so the likelihood function takes form:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} \left[y_n^{t_n} (1 - y_n)^{1 - t_n} \right], \quad y_n = \sigma(\mathbf{w}^T \mathbf{x}_n).$$

- Taking the negative log of the likelihood, we can define the cross-entropy error function (that we want to minimize):

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right] = \sum_{n=1}^{N} E_n.$$

- Differentiating and using the chain rule:

$$\frac{\mathrm{d}}{\mathrm{d}y_n} E_n = \frac{y_n - t_n}{y_n(1 - y_n)}, \quad \frac{\mathrm{d}}{\mathrm{d}\mathbf{w}} y_n = y_n(1 - y_n)\mathbf{x}_n, \quad \boxed{\frac{\mathrm{d}}{\mathrm{d}a}\sigma(a) = \sigma(a)(1 - \sigma(a)).}$$

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{w}} E_n = \frac{\mathrm{d}E_n}{\mathrm{d}y_n} \frac{\mathrm{d}y_n}{\mathrm{d}\mathbf{w}} = (y_n - t_n)\mathbf{x}_n.$$

- Note that the factor involving the derivative of the logistic function cancelled.

ML for Logistic Regression

- We therefore obtain:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\mathbf{x}_n.$$

prediction target

- This takes exactly the same form as the gradient of the sum-of- squares error function for the linear regression model.
- Unlike in linear regression, there is no closed form solution, due to nonlinearity of the logistic sigmoid function.
- The error function is convex and can be optimized using standard gradient-based (or more advanced) optimization techniques.

Multiclass Logistic Regression

- For multiclass case, the posterior probability is represented by a *softmax transformation* of linear functions of input variables:
- $$p(\mathcal{C}_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})}.$$
- Maximum likelihood is used to determine the parameters of this discriminative model directly.
- Suppose we observe a data set $\{\mathbf{x}_n, t_n\},\ n = 1, .., N$, where we use 1-of-K encoding for the target vector t_n.
- So if $\mathbf{x}_n$ belongs to class C_k, then **t** is a binary vector of length K containing a single 1 for element k (the correct class) and 0 elsewhere.
- For example, K=5, an input belonging to class 2 would be given a target vector:

$$t = (0, 1, 0, 0, 0)^T$$

Multiclass Logistic Regression

- We can write down the likelihood function:

$$p(\mathbf{T}|\mathbf{X}, \mathbf{w}_1, ..., \mathbf{w}_K) = \prod_{n=1}^{N} \underbrace{\left[\prod_{k=1}^{K} p(\mathcal{C}_k|\mathbf{x}_n)^{t_{nk}}\right]}_{\text{Only one term corresponding to correct class contributes.}} = \prod_{n=1}^{N} \left[\prod_{k=1}^{K} y_{nk}^{t_{nk}}\right]$$

N x K binary matrix of target variables.

- Where

$$y_{nk} = p(\mathcal{C}_k|\mathbf{x}_n) = \frac{\exp(\mathbf{w}_k^T \mathbf{x}_n)}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x}_n)}.$$

- Taking the negative logarithm gives the cross-entropy entropy function for multi-class classification problem:

$$E(\mathbf{w}_1, ..., \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{X}, \mathbf{w}_1, ..., \mathbf{w}_K) = -\sum_{n=1}^{N} \left[\sum_{k=1}^{K} t_{nk} \ln y_{nk}\right].$$

- Take the gradient:

$$\nabla E_{\mathbf{w}_j}(\mathbf{w}_1, ...\mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\mathbf{x}_n.$$

Special Case of Softmax

- If we consider a softmax function for two classes

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{\exp(a_1)}{\exp(a_1) + \exp(a_2)} = \frac{1}{1 + \exp(-(a_1 - a_2))} = \sigma(a_1 - a_2).$$

- So the logistic sigmoid is just a special case of the softmax function that avoids using redundant parameters:
  - Adding the same constant to both a_1 and a_2 has no effect.
  - The over-parameterization of the softmax is because probabilities must add up to one.

Summary

- Generative approach: Determine the class conditional densities and class-specific priors, and then use Bayes' rule to obtain the posterior probabilities.
  - Different models can be trained separately on different machines.
  - It is easy to add a new class without retraining all the other classes.

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

- Discriminative approach: Train all of the model parameters to maximize the probability of getting the labels right.
- Model $p(\mathcal{C}_k|\mathbf{x})$ directly.

References

- Chapter 5, Deep Learning Book.