Classification

Vahid Tarokh
CEE 690/ECE 590, Fall 2019

Introduction

- We will little by little get into logistic regression and the construction of neural Neural Networks.
- Important Note: Source of some of my slides (with great appreciation and acknowledgements)
  - Professor David Carlson Slides
  - Professor Alex Smola's slides (available online)
  - Professor Ruslan Salakhutdinov's slides (available online)
  - Professor Hugo Larochelle's class on Neural Networks

Linear Models for Classification

- So far, we have looked at the linear models for regression that have particularly simple analytical and computational properties.

- We will now look at the analogous class of models for solving classification problems.

- We will also look at the Bayesian treatment of linear models for classification.

Classification

• The goal of classification is to assign an input **x** into one of K discrete classes C_k, where k=1,..,K.

• Typically, each input is assigned only to one class.

• Example: The input vector **x** is the set of pixel intensities, and the output variable t will represent the presence of cancer, class C_1, or absence of cancer, class C_2.

C_1: Cancer present

C_2: Cancer absent

x -- set of pixel intensities

Linear Classification

• The goal of classification is to assign an input **x** into one of K discrete classes C_k, where k=1,..,K.

• The input space is divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

• We will consider linear models for classification. Remember, in the simplest linear regression case, the model is linear in 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 class labels, or posterior probabilities that lie in the range of (0,1), so we use a nonlinear function.

Linear Classification

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

• The decision surfaces correspond to $y(\mathbf{x}, \mathbf{w}) = \text{const},$ so that $\mathbf{x}^T\mathbf{w} + w_0 = \text{const},$ and hence the decision surfaces are linear functions of **x**, even if the activation function is nonlinear.

• This class of models is called generalized linear models.

• Note that these models are no longer linear in parameters, due to the presence of nonlinear activation function.

• This leads to more complex analytical and computational properties, compared to linear regression.

• Note that we can make a fixed nonlinear transformation of the input variables using a vector of basis functions $\phi(\mathbf{x}),$ as we did for regression models.

Notation

• In the case of two-class problems, we can use the binary representation for the target value $t \in \{0,1\}$ such that t=1 represents the positive class and t=0 represents the negative class.

- We can interpret the value of t as the probability of the positive class, and the output of the model can be represented as the probability that the model assigns to the positive class.

• If there are K classes, we use a 1-of-K encoding scheme, in which **t** is a vector of length K containing a single 1 for the correct class and 0 elsewhere.

• For example, if we have K=5 classes, then an input that belongs to class 2 would be given a target vector:

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

- We can interpret a vector **t** as a vector of class probabilities.

Three Approaches to Classification

- First approach: Construct a discriminant function that directly maps each input vector to a specific class.

- Model the conditional probability distribution $p(\mathcal{C}_k|\mathbf{x})$, and then use this distribution to make optimal decisions.

- There are two alternative approaches:
  - Discriminative Approach: Model $p(\mathcal{C}_k|\mathbf{x})$, directly, for example by representing them as parametric models, and optimize for parameters using the training set (e.g. logistic regression).
  - Generative Approach: Model class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ together with the prior probabilities $p(\mathcal{C}_k)$ for the classes. Infer posterior probability using Bayes' rule:

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

  - For example, we could fit multivariate Gaussians to the input vectors of each class. Given a test vector, we see under which Gaussian the test vector is most probable.

Discriminant Functions

- Consider: $y(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + w_0.$

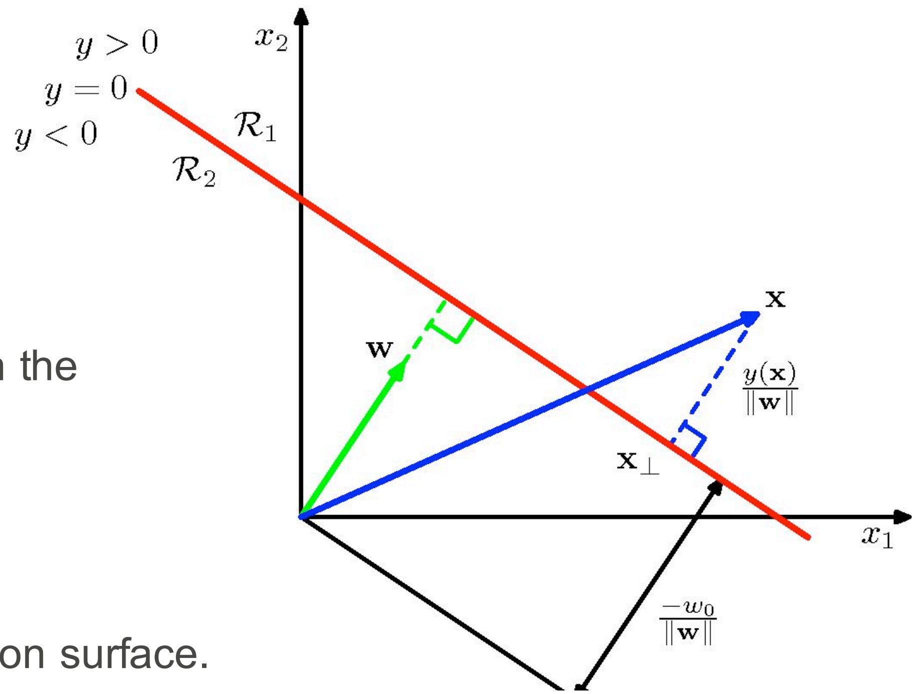
- Assign $\mathbf{x}$ to C_1 if $y(\mathbf{x}) \geq 0,$ and class C_2 otherwise.

- Decision boundary:

$$y(\mathbf{x}) = 0.$$

- If two points $\mathbf{x}_A$ and $\mathbf{x}_B$ lie on the decision surface, then:

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



- $\mathbf{w}$ is orthogonal to the decision surface.

- If $\mathbf{x}$ is a point on the decision surface, then: $\dfrac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||} = -\dfrac{w_0}{||\mathbf{w}||}.$

- Hence w_0 determines the location of the decision surface.

Multiple Classes

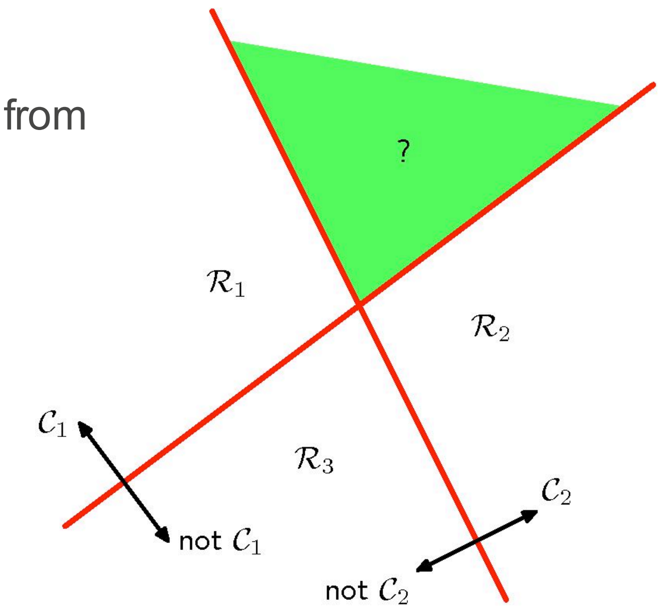
• Consider the extension of linear discriminants to K>2 classes.

• One option is to use K-1 classifiers, each of which solves a two class problem:

- Separate points in class C_k from points not in that class.

• There are regions in input space that are ambiguously classified.

One-versus-the-rest



Multiple Classes

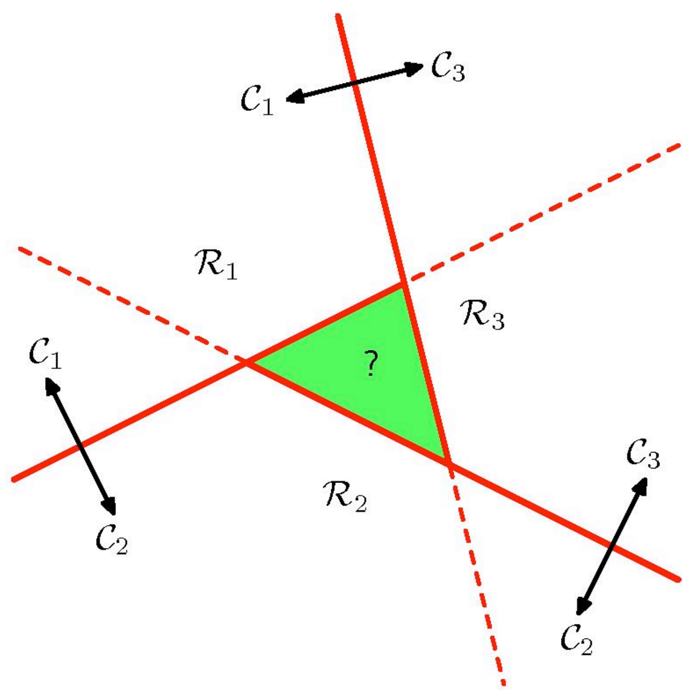
• Consider the extension of linear discriminants to K>2 classes.

• An alternative is to use K(K-1)/2 binary discriminant functions.

- Each function discriminates between two particular classes.

• Similar problem of ambiguous regions.

One-versus-one



Simple Solution

- Use K linear 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 class C_k, if $y_k(\mathbf{x}) > y_j(\mathbf{x}) \ \forall j \neq k$ (pick the max).

- This is guaranteed to give decision boundaries that are singly connected and convex.

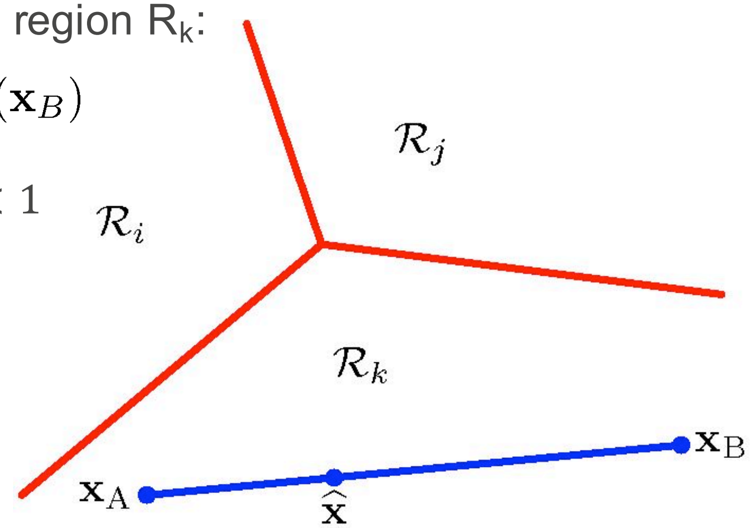
- For any two points that lie inside the region R_k:

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

implies that for any positive $0 < \alpha < 1$

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

due to linearity of the discriminant functions.



The Perceptron Algorithm

- We now consider another example of a linear discriminant model.
- Consider the following generalized linear model of the form

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

where nonlinear activation 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 transformation $\phi(\mathbf{x})$.

- 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).
- We will consider an alternative error function.
- First, note that:
  - Patterns x_n in Class C_1 should satisfy:

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

  - Patterns x_n in Class C_2 should satisfy:

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

- 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 ∈ { -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 the set of all misclassified patterns

• The error function is linear in **w** in regions of **w** space where the example is misclassifies 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 in weight is given by:

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

where 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 η = 1.

• Note that the contribution to the error from a 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 from a 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

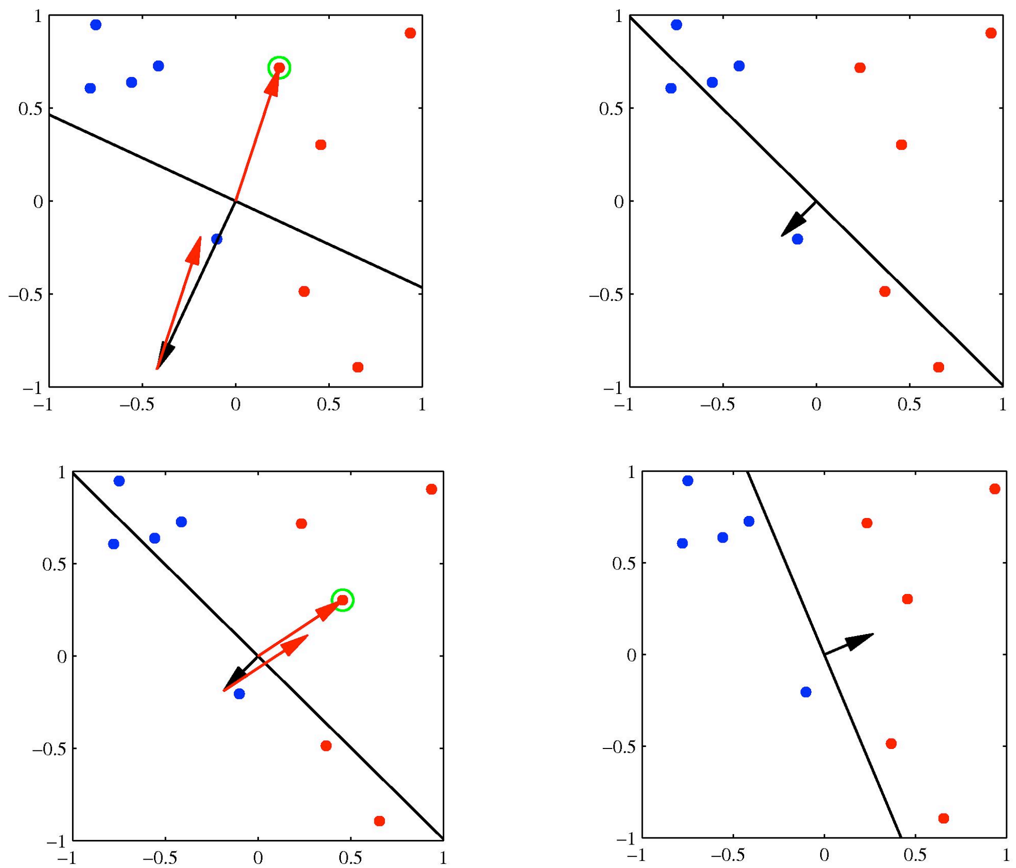
• However, the change in **w** may cause some previously correctly classified points to be misclassified. No convergence guarantees in general.

• If there exists an exact solution (if the training set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in finite number of steps.

• The perceptron does not provide probabilistic outputs, not does it generalize readily to K>2 classes.

Illustration of Convergence

- Convergence of the perceptron learning algorithm



Three Approaches to Classification

- Construct a discriminant function that directly maps each input vector to a specific class.
- Model the conditional probability distribution $p(\mathcal{C}_k|\mathbf{x})$, and then use this distribution to make optimal decisions.
- There are two alternative approaches:
  - Discriminative Approach: Model $p(\mathcal{C}_k|\mathbf{x})$, directly, for example by representing them as parametric models, and optimize for parameters using the training set (e.g. logistic regression).
  - Generative Approach: Model class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ together with the prior probabilities $p(\mathcal{C}_k)$ for the classes. Infer posterior probability using Bayes' rule:

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

We will consider next.

Probabilistic Generative Models

• Model class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ separately for each class, as well as the class priors $p(\mathcal{C}_k)$.

• Consider the case of two classes. The posterior probability of class C_1 is given by:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$= \frac{1}{1 + \exp(-a)} = \sigma(a),$$

Logistic sigmoid function

where we defined:

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

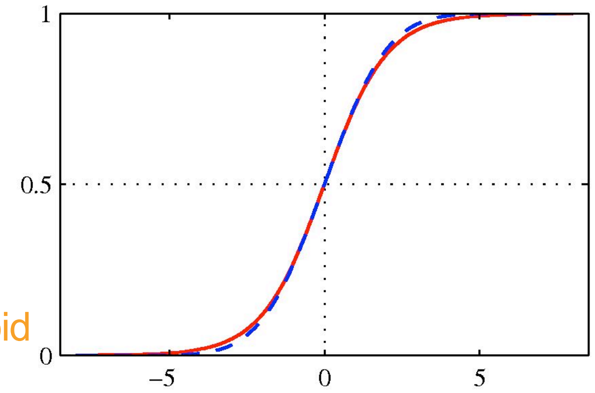
which is known as the logit function. It represents the log of the ratio of probabilities of two classes, also known as the log-odds.

Sigmoid Function

- The posterior probability of class C_1 is given by:

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

Logistic sigmoid function



- The term sigmoid means S-shaped: it maps the whole real axis into (0 1).

- It satisfies:

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

Softmax Function

• For case of K>2 classes, we have the following multi-class generalization:

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

• This normalized exponential is also known as the softmax function, as it represents a smoothed version of the max function:

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

• We now look at some specific forms of class conditional distributions.

Example of Continuous Inputs

- Assume that the input vectors for each 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 the case of two classes, the posterior is logistic function:

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

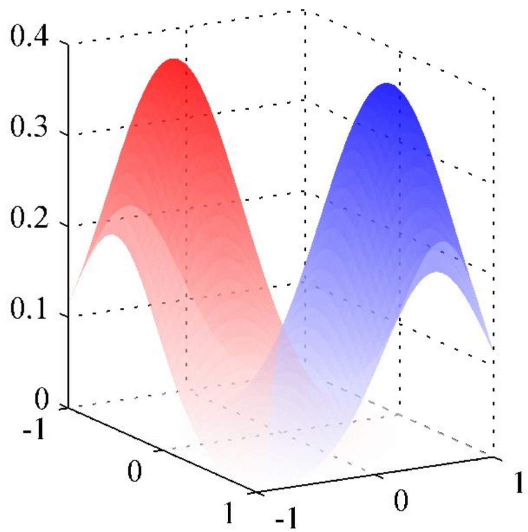
where we have defined:

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

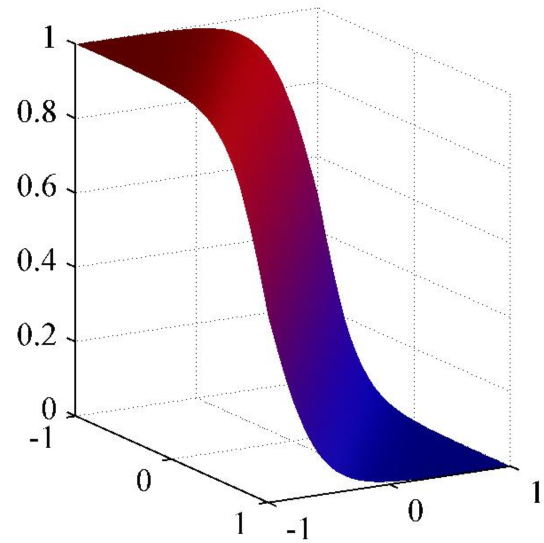
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}.$$

- The quadratic terms in **x** cancel (due to the assumption of common covariance matrices).
- 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 Classes

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

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

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

where, similar to the 2-class case, we have defined:

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

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

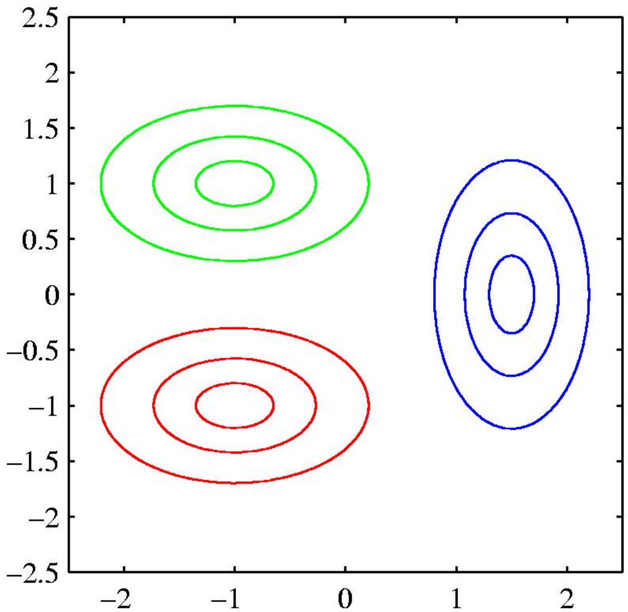
- Again, the decision boundaries are linear in input space.

- If we allow each class-conditional density to have its own covariance, we will obtain quadratic functions of **x**.

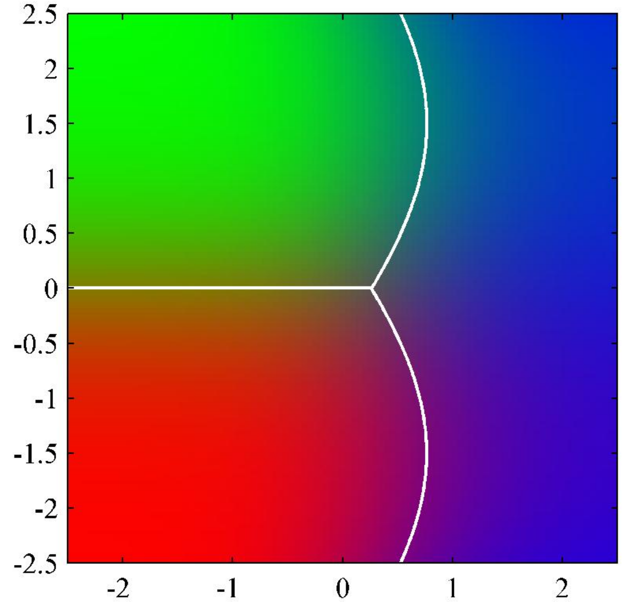
- This leads to a quadratic discriminant.

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

• Consider the case of two classes, each having a Gaussian class-conditional density with shared covariance matrix.

• We observe a dataset $\{\mathbf{x}_n, t_n\}, \;\; n = 1, .., N.$

- Here $t_n=1$ denotes class C_1, and $t_n=0$ denotes class C_2.
- Also denote $p(\mathcal{C}_1) = \pi, \;\; p(\mathcal{C}_2) = 1 - \pi.$

• The likelihood function takes form:

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

Data points from class C_1.

Data points from class C_2.

• As usual, we will maximize the log of the likelihood function.

Maximum Likelihood Solution

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

- Maximizing the respect to ¼, we look at the terms of the log-likelihood functions that depend on ¼:

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

Differentiating, we get:

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

- Maximizing the respect to $^{1}{}_{1}$, we look at the terms of the log-likelihood functions that depend on $^{1}{}_{1}$:

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

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

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

- Maximizing the respect to §:

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

- Here we defined:

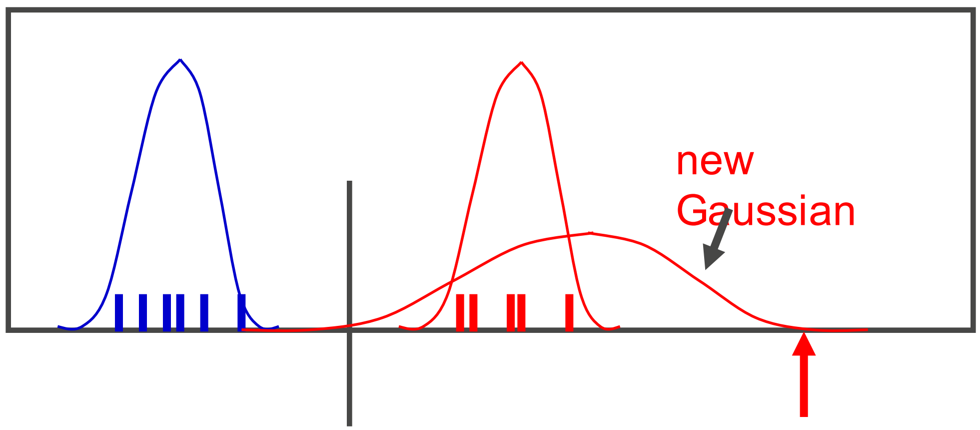
$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2,$$
$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T,$$
$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n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

- 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.

Example



decision boundar y

What happens to the decision boundary if we add a new red point here?

• For generative fitting, the red mean moves rightwards but the decision boundary moves leftwards! If you believe the data is Gaussian, this is reasonable.

• How can we fix this?

Three Approaches to Classification

- Construct a discriminant function that directly maps each input vector to a specific class.
- Model the conditional probability distribution $p(\mathcal{C}_k|\mathbf{x})$, and then use this distribution to make optimal decisions.
- There are two approaches:
  - Discriminative Approach: Model $p(\mathcal{C}_k|\mathbf{x})$, directly, for example by representing them as parametric models, and optimize for parameters using the training set (e.g. logistic regression).
  - Generative Approach: Model class conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ together with the prior probabilities $p(\mathcal{C}_k)$ for the classes. Infer posterior probability using Bayes' rule:

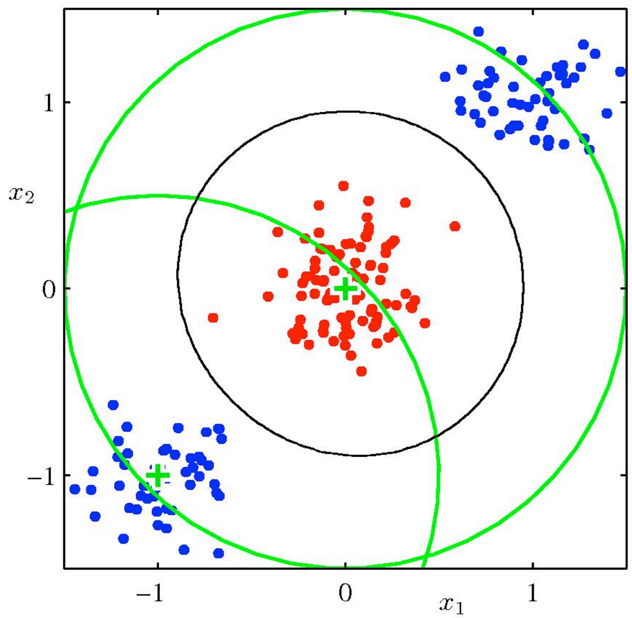
$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}.$$

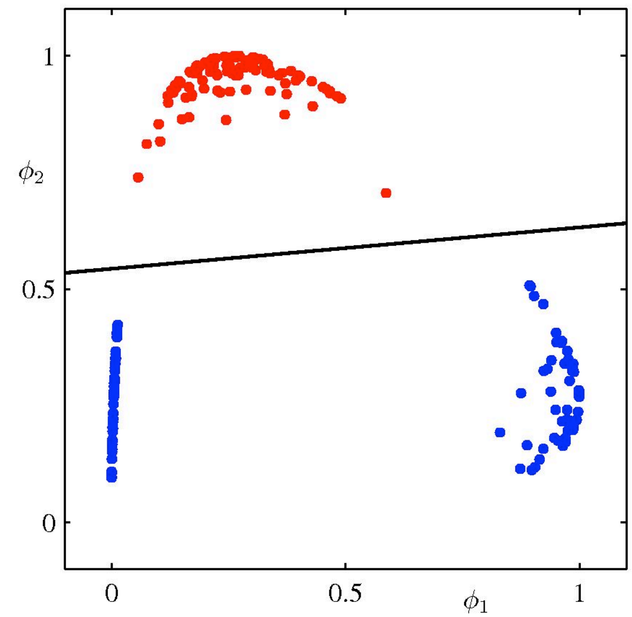
We will consider next.

Fixed Basis Functions

- So far, we have considered classification models that work directly in the input space.
- All considered algorithms are equally applicable if we first make a fixed nonlinear transformation of the input space using vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$.
- Decision boundaries will be linear in the feature space $\boldsymbol{\phi}$, but would correspond to nonlinear boundaries in the original input space $\mathbf{x}$.
- Classes that are linearly separable in the feature space $\boldsymbol{\phi}(\mathbf{x})$ need not be linearly separable in the original input space.

Linear Basis Function Models





- We define two Gaussian basis functions with centers shown by green the crosses, and with contours shown by the green circles.
- Linear decision boundary (right) is obtained using logistic regression, and corresponds to nonlinear decision boundary in the input space (left, black curve).

Logistic Regression

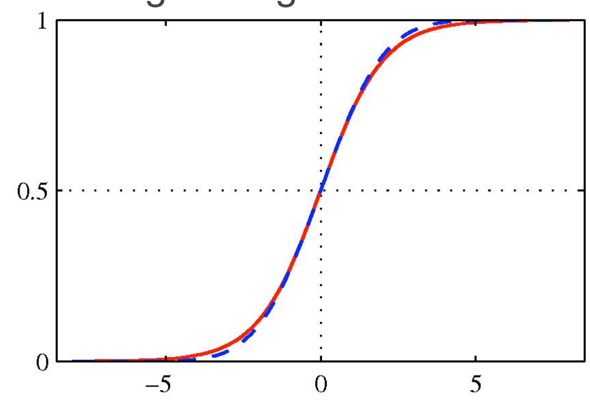
- Let us look at the two-class classification problem.
- We have seen that the posterior probability of class C_1 can be written as a sigmoid function:

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

where $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x}),$ 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).

logistic sigmoid function



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.

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:

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

- 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.

- Easy to adapt to the online learning setting.

Multiclass Logistic Regression

- For the multiclass case, we represent posterior probabilities by a softmax transformation of linear functions of input variables:

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

- Unlike in generative models, here we will use maximum likelihood to determine parameters of this discriminative model directly.

- As usual, we observed a dataset $\{\mathbf{x}_n, t_n\}, \;\; n = 1, .., N,$ where we use 1-of-K encoding for the target vector $\mathbf{t}_n$.

- So if $\mathbf{x}_n$ belongs to class C_k, then $\mathbf{t}$ is a binary vector of length K containing a single 1 for element k (the correct class) and 0 elsewhere.

- For example, if we have K=5 classes, then an input that belongs 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 £ K binary matrix of target variables.

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

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

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

- Taking the gradient:

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

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.

Recap

- 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.