

Ch.5 - Classification

🕒 Created	@January 16, 2024 2:51 PM
📖 Book	Deep Learning: Foundations and Concepts

[Ch5-Classification.pdf](#)



Classification

Terms

Discriminant Function

Binary Class

Multiple Classes

Least Squares

Decision Theory

Misclassification Rate

Loss Function

ROC Curve

Generative Classifiers

Binary Class

Multiple **Classes**

Maximum Likelihood Solution

Input Features

Continuous

Discrete

Discriminative Classifiers

Activation Function

Fixed Basis Function

Logistic Regression

Binary Class

Multi Class

Terms

- **Decision Surface**
- **Hyperplane**
- **Orthogonal Projection**
- Linearly Separable
- Cross-Entropy

Approaches to solving decision problem can be identified into three categories.

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

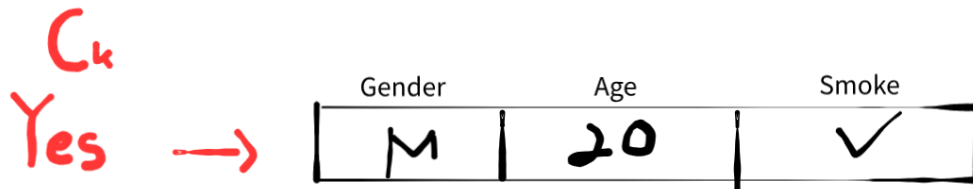
(1) Generative Model

1. Find $p(x|C_k)$ from large data set

- infer

$$p(C_k), p(x)$$

▼ example of class-conditional case



2. Find $p(C_k|x)$ based on info from 1.
3. Use decision theory $\sum_k L_{kj}p(C_k|x)$ to assign class

(2) Discriminative Model

1. Find $p(C_k|x)$
2. Use decision theory to assign class

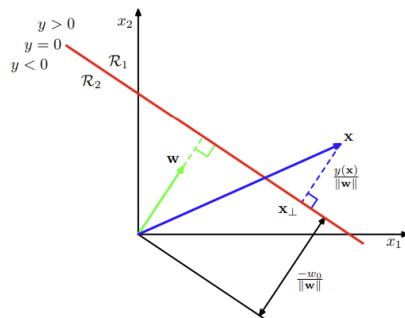
(3) Discriminant Function

Find a boundary function $f(x)$ and map each input directly to a class

The essential difference between generative model and the others is that the generative model knows underlying distributions of $p(x|C_k)$ and $p(C_k)$, $p(x, C_k)$. It means that it knows real-world distribution, not artificially modified training set.

Discriminant Function

Binary Class



Discriminant Function directly decides the class of an input vector without inference stage. It uses D-1 dimensional hyperplane $y(\vec{x}) = \vec{w}^T \vec{x} + w_0 = 0$ to divide areas of input space.

▼ \vec{w} : orthogonal to decision surface

if x_A and x_B sit on the surface:

$$\begin{aligned} w^T x_A + w_0 &= 0 \\ w^T x_B + w_0 &= 0 \\ \rightarrow y(x_A - x_B) &= w^T (x_A - x_B) = 0 \end{aligned}$$

hence, $w \perp (x_A - x_B)$.

▼ $\frac{y(\vec{x})}{||w||}$: perpendicular distance to decision surface

$$x = x^\perp + r \frac{w}{||w||}$$

$$y(x) = w^T (x^\perp + r \frac{w}{||w||}) + w_0$$

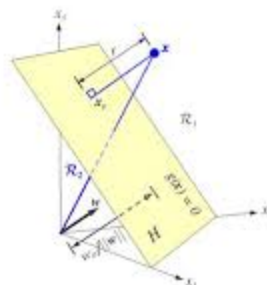
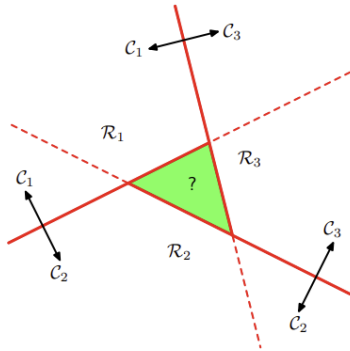


Figure 9.2: The linear decision boundary H separates the feature space into two half-spaces.

Multiple Classes



Problem: one-vs-rest classifiers

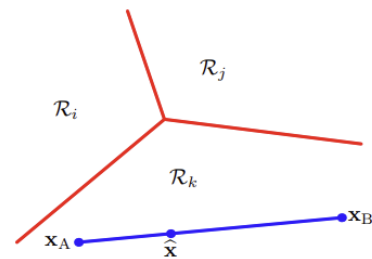
Now we can extend it to multi-class classification where input \vec{x} is determined to be C_k . A model with multiple binary discriminant functions can't simply be a solution since it creates arbitrary boundary where it results in multiple classes.

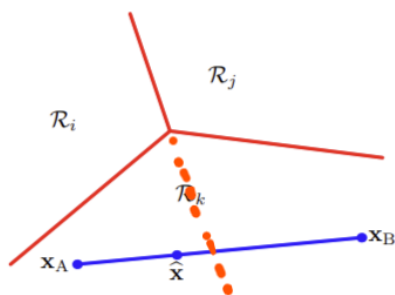
Solution: distance-based classification

$$y_k(\mathbf{x}) = w_k^T \mathbf{x} + w_{k0}$$

This assigns a point \mathbf{x} to class C_k if $y_k(x) > y_j(x)$ for all $j \neq k$, choosing the class that is furthest from the one-vs-rest classifier y_k . Then the boundary between classes is given by $y_k(x) = y_j(x)$, therefore:

$$(w_k - w_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$





Proof of singly connected decision boundary

Let

\hat{x} be any point lying on the line between x_A and x_B in the same class:

$$\hat{x} = \lambda x_A + (1 - \lambda) x_B$$

By the linearity of the discriminant functions:

$$y_k(\hat{x}) = \lambda y_k(x_A) + (1 - \lambda) y_k(x_B)$$

Since $y_k(x_{AB})$ is greater than $y_j(x_{AB})$ for any j , \hat{x} is also classified as the same class, canceling out the half of each decision boundary function.

Least Squares

Each class C_k has its own discriminant function so that $y_k(\mathbf{x}) = w_k^T \mathbf{x} + w_{k0}$, conveniently using vector notation as $\mathbf{y} = \tilde{W} \tilde{\mathbf{x}}$. Consider a training data set $\{x_n, t_n\}$ and define matrices as:

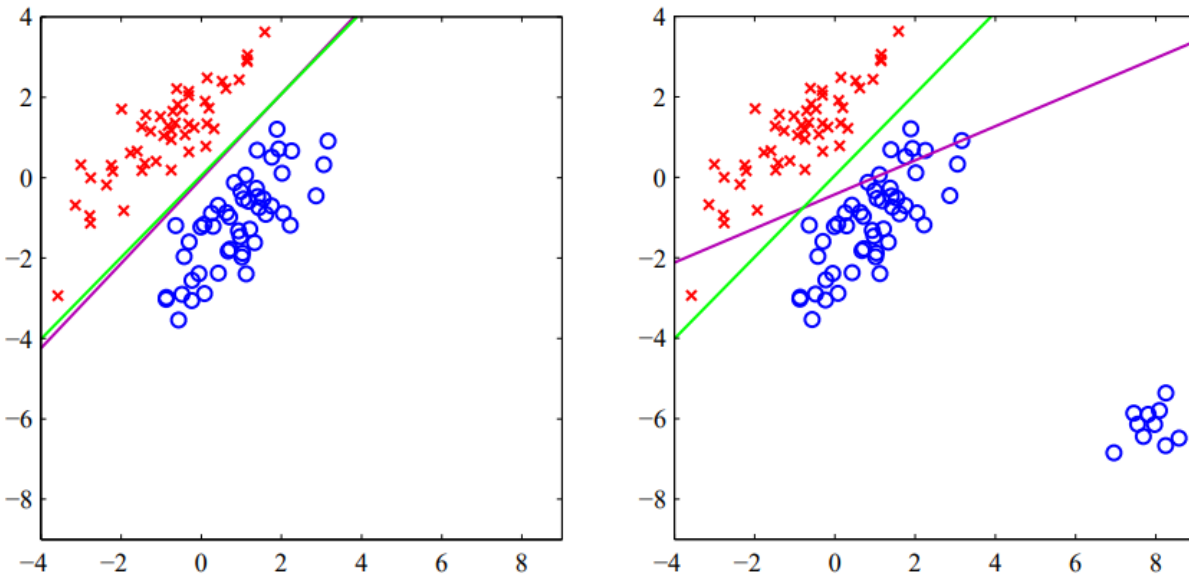
$$\begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \end{bmatrix} = [\mathbf{X} \mid \mathbf{T}]$$

$$\begin{aligned} \mathbf{T} &= (t_1 \ t_2 \ t_3 \ \dots)^T \\ \mathbf{X} &= (\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3 \ \dots)^T \end{aligned}$$

Where \mathbf{T} 's n th row is t_n^T (One-Hot) and \mathbf{X} 's n th row is \mathbf{x}_n^T , then a least squares equation can be written as below, with \mathbf{X}^\dagger as pseudo-inverse of \mathbf{X} in case of non square matrix.

$$X\tilde{W} = T \rightarrow \tilde{W} = (X^T X)^{-1} X^T T = X^\dagger T$$

However, it suffers from lacking **robustness**, which means sensitive to a very few outliers. This is because least squares method, as well as MSE, expects training data to be continuous and to follow Gaussian distribution from its regression function, whereas binary-coded targets are clearly far from it.



➡ A different learning model will be used in the later chapter

Decision Theory



Classification can be broken down into two stages: **inference + decision**

- **inference: (main)**

use training data to learn a model for $p(C_k, x)$

- **decision: (simple)**

choose the optimal probability/class based on inference

Let's consider a medical diagnosis problem which determines whether the given image indicates the patient has cancer or not. We denote x as an input vector, t as its classification:

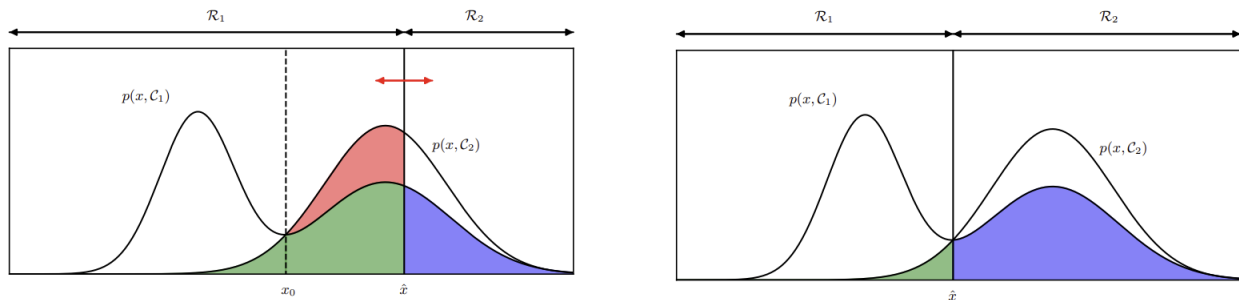
$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \begin{cases} C_1 \text{ or } t = 0 & \rightarrow \text{Presence of cancer} \\ C_2 \text{ or } t = 1 & \rightarrow \text{Absence of cancer} \end{cases}$$

- $P(C_k)$: (Prior) Probability of a person has cancer, regardless of any pictures taken
- $P(C_k|x)$: (Posterior) Probability of a person has cancer based on the given image

Misclassification Rate



Minimize the chance of assigning \times to the wrong class



Regardless of decision rule, we'd like to minimize the probability of making mistake which is:

$$\begin{aligned} p(\text{mistake}) &= \int_{R_1} p(x, C_2) dx + \int_{R_2} p(x, C_1) dx \\ &= \text{Type 1} + \text{Type 2} \end{aligned}$$

It's valid to use joint probability because the evidence $P(x)$ is common to every posterior probability $P(C_k, x) = P(C_k|x)P(x)$. The optimal decision boundary \hat{x} can be found heuristically by minimizing varying cost area of red region.

Loss Function



Penalize some decision choice at the expense of making other mistakes

Figure 5.6 An example of a loss matrix with elements L_{kj} for the cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the assignment of class made by our decision criterion.

	normal	cancer
normal	0	1
cancer	100	0

In real scenario, a case must be avoided where a patient with cancer is diagnosed as normal. When PDF of either $p(x, C_k)$ or $p(C_k|x)$ is given, the class for input x can be found heuristically by minimizing the average of Loss values.

$$E[L] = \sum_k \sum_j \int_{R_j} L_{kj} p(x, C_k) dx$$

Since $P(x)$ is common for all $p(x, C_k) = p(C_k|x)P(x)$, the decision rule can be shortened to minimizing the sum of:

$$\sum_k L_{kj} p(C_k|x)$$



Reject Option

Classification errors arise from the regions of input space where:

- the largest

$p(C_k|x)$ is significantly less than 1

- all the

$p(x, C_k)$ have comparable values

In such cases of ambiguity, the model would reject classification and request further examination.

θ for reject threshold can be set for this purpose.

ROC Curve

Figure 5.9 The confusion matrix for the cancer treatment problem, in which the rows correspond to the true class and the columns correspond to the assignment of class made by our decision criterion. The elements of the matrix show the numbers of true negatives, false positives, false negatives, and true positives.

	normal	cancer
normal	N_{TN}	N_{FP}
cancer	N_{FN}	N_{TP}

Confusion Matrix

- N_{TP} : (True Positive) prediction of positive is true (actual: positive)

ex) A person diagnosed as cancer actually has cancer

- N_{FP} : (False Positive) prediction of positive is false (actual: negative)

ex) A person diagnosed as cancer is actually healthy

- N_{TN} : (True Negative) prediction of negative is true (actual: negative)

ex) A person diagnosed as healthy is actually healthy

- N_{FN} : (False Negative) prediction of negative is false (actual: positive)

ex) A person diagnosed as healthy actually has cancer

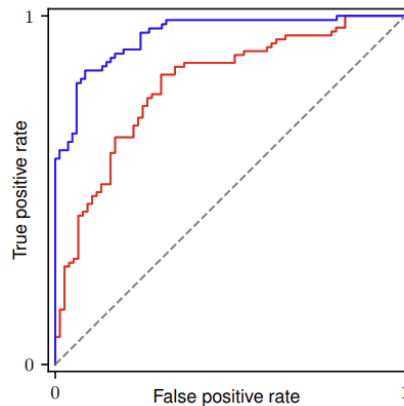
Accuracy

Accuracy, measured by the fraction of correct classifications is:

$$Accuracy = \frac{N_{TP} + N_{TN}}{N}$$

Quantities

- **Precision:** $\frac{N_{TP}}{N_{TP} + N_{FP}}$
- **Recall:** $\frac{N_{TP}}{N_{TP} + N_{FN}}$
- **True Positive Rate:** $\frac{N_{TP}}{N_{TP} + N_{FN}}$
- **False Positive Rate:** $\frac{N_{FP}}{N_{FP} + N_{TN}}$



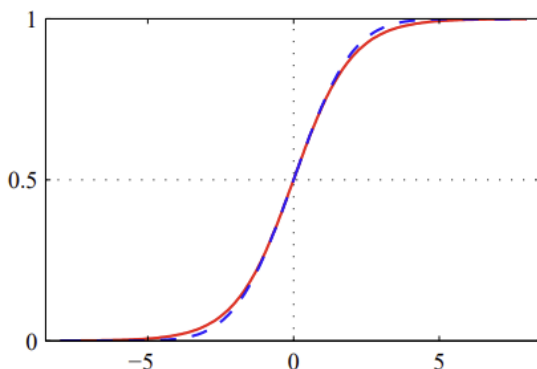
		Actual	
		Positive	Negative
Predict	Positive	TP	FP
	Negative	FN	TN

The goal is to minimize false positive rate and maximize true positive rate. Accordingly, a model which test result locates on the upper left corner is what we want.

Generative Classifiers

Generative classifiers first learn/assume class-conditional densities $p(x|C_k)$ and prior $p(C_k)$ then compute posterior $p(C_k|x)$. The key benefit of such model is that it learns underlying distributions of each class, thus enabling generating new data set that follows more realistic distribution.

Binary Class



Using Bayes' formula and the sum rule, the posterior probability can be expressed as sigmoid function:

$$\begin{aligned}
 p(C_1|\mathbf{x}) &= \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \\
 &= \frac{1}{1 + \frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1)}} \\
 &= \frac{1}{1 + e^{-a}} = \sigma(a) \quad (a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)})
 \end{aligned}$$



Definition of **logit** function

The inverse of a sigmoid function $\sigma(a)$ is called **logit** : $\ln \frac{\sigma}{1-\sigma}$

Multiple Classes

If there are $K > 2$ classes, the posterior probability $p(C_k|x)$ can be regarded as **softmax** function, which turns inputs of any range into between 0 and 1 so that they can be interpreted as probabilities.

$$\begin{aligned}
 p(C_k|x) &= \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} \\
 &= \frac{e^{a_k}}{\sum_j e^{a_j}} \quad (a_k = \ln(p(x|C_k)p(C_k)))
 \end{aligned}$$

Maximum Likelihood Solution

Suppose we have binary class problem, each having Gaussian class-conditional density with shared covariance for simplification. Then variables are: prior $p(C_1) = \pi$, means μ_1, μ_2 and covariance Σ . We'll derive those variables which maximizes the likelihood function as follows:

$$p(t, X|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^N p(\mathbf{x}_n, C_1)^{t_n} p(\mathbf{x}_n, C_2)^{1-t_n}$$

- $p(x_n, C_1) = \pi N(x_n|\mu_1, \Sigma)$
- $p(x_n, C_2) = (1 - \pi) N(x_n|\mu_2, \Sigma)$

< $t=1$ for C_1 , $t=0$ for C_2 >

As a result of each partial derivatives with respect to π , μ , Σ and setting the equation to zero, solutions are given as follows. These corresponds to general calculation of probabilities, means and variances.

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

$$\mu_1 = \frac{1}{N} \sum_{n=1}^N t_n \mathbf{x}_n$$

$$\mu_2 = \frac{1}{N} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n$$

$$\Sigma = \frac{1}{N} \sum_{n \in C_1} (x_n - \mu_1)^2 + \frac{1}{N} \sum_{n \in C_2} (x_n - \mu_2)^2$$

Input Features

Generative models first start with learning distributions $p(x|C_k)$ and $p(C_k)$, by assuming ideal density function of $p(x|C_k)$. Secondly, it derives a linear function of x from `logit` function which will be fed into a **sigmoid**, if binary class, or a **softmax**, if multi classes. These are examples of deriving linear function of x when distributions $p(x|C_k)$ and $p(C_k)$ are given.

Continuous

As for inputs with only continuous features, let assume the class-conditional distribution follows Gaussian with covariance matrix Σ .

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

Substituting $p(x|C_k)$ in $a_k = \ln(p(x|C_k)p(C_k))$ with the density function above becomes:

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

in which we have defined

$$\begin{aligned} \mathbf{w}_k &= \Sigma^{-1} \boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2} \boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k + \ln p(C_k). \end{aligned}$$

Discrete

Suppose we have an input \mathbf{x} which features are **independent** and discrete (0 or 1). When μ_{ki} is a probability of observing x_i given that class is C_k , by its independence, the likelihood term $p(\mathbf{x}|C_k)$ can be expressed as $\prod p(x_i|C_k)$:

$$p(\mathbf{x}|C_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i},$$

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(C_k),$$

Discriminative Classifiers

Generative classifiers require distributions over $p(\mathbf{x}|C_k)$ and $p(C_k)$, which are often under ideal assumption. Alternative approach is to determine $p(C_k|\mathbf{x})$'s parameters directly by maximizing a likelihood function. Since it's more empirical way than relying on assumptions, it may lead to improved predictive performance.

Activation Function

| Softmax, Sigmoid, ReLU, ...

Activation function $f(\cdot)$ takes linear combination of an input and returns a result which then can be interpreted as inference for decision. The inverse of activation function $f^{-1}(\cdot)$ is called **link function**.

$$y(x, w) = f(w^T x + w_0) \cdots \text{activation}$$

$$f^{-1}(y(x, w)) = w^T x + w_0 \cdots \text{link}$$

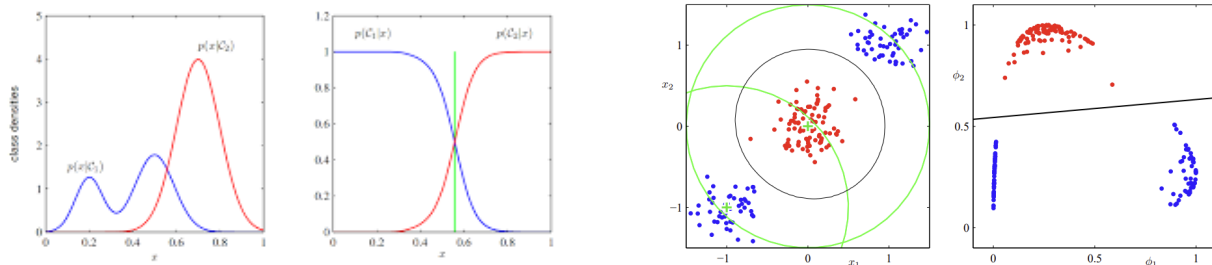
▼ Example: Independent Discrete Input Features

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(C_k),$$

$$p(C_k|x) = \frac{e^{a_k}}{\sum_j e^{a_j}}$$

< Linear combination of an input goes through **softmax** activation function for decision rule >

Fixed Basis Function



< basis function doesn't remove overlap between class-conditional probabilities >

< impact of basis function >

Basis function $\phi(\mathbf{x})$ changes the basis of the original space so that it makes the process of modelling the posterior probabilities easier. The picture on the right side shows the non-linear decision boundary becomes linear by changing the basis.

❌ It can't remove overlap between class-conditional probabilities. If not, it may worsen

Logistic Regression

In generative approach, we need to first figure out class-conditional densities $p(x|C_1)$ and $p(x|C_2)$ (for binary class). Given that feature space ϕ is an M-dimensional, using maximum likelihood to fit the two distributions, it requires $O(2M)$ for the means and $O(\frac{M(M+1)}{2})$ for covariance matrix Σ . Such quadratic growth of complexity is better to be avoided thus using discriminative approach has a clear advantage.


Binary Class

For a data set $\{\phi_n, t_n\}$, where $\phi_n = \phi(\mathbf{x})$ and $t_n \in \{0, 1\}$, posterior probability can be expressed as **sigmoid** $\sigma(a_n) = \frac{1}{1+e^{-a_n}}$.

$$p(C_1|x) = \frac{1}{1 + e^{-a_n}} = \sigma(a_n), \text{ where } a_n = w^T \phi_n(x)$$

The **likelihood function** is then given as the product of all the right predictions:

$$p(t|w) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n} \dots (y_n = \sigma_n)$$

parameter matrix w should maximize the likelihood. we can get a computable cost function by taking log to the likelihood, which derivative enables us to explore optimal  through gradient descent.

$$\ln(p(t|w)) = \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} = E(w)$$

Since $\nabla a_n = \phi_n$ and $\nabla y_n = y_n(1 - y_n)$, gradient of $E(w)$ becomes identical to derivative of the cost function $\frac{\partial J}{\partial w}$ in linear regression.

$$\begin{aligned} -\nabla E(w) &= -\nabla \ln p(t|w) = \sum_{n=1}^N (y_n - t_n) \phi_n \\ \rightarrow w &= w - \alpha \nabla E(w) \text{ for gradient-descent} \end{aligned}$$



Issue of Severe Overfitting

However, MLE can exhibit severe overfitting to the training samples because the decision boundary occurs when the hyperplane is sliced to

$\sigma = \frac{1}{1+e^{-a}} = 0.5$, equivalent to $a = w^T \phi \rightarrow \infty$, thus $w = \|(w_0 \ w_1 \ \dots)^T\| \rightarrow \infty$, meaning that the linear function in feature space ϕ has infinitely steep gradient.

? How is it convex ?

Multi Class

For a data set $\{t_n, \phi_n\}$, where a target vector has notation of 1-of-K coding scheme $t_n = \{0, 1, 0, \dots\} = C_2$, we can directly use the **softmax** form of posterior probability and its derivative for gradient descent.

$$p(C_k|\phi) = y_k(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}} \text{ where } a_k = w_k^T \phi$$

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \text{ where } I_{kj} = \text{identity}$$

The likelihood function is then given as the product of all the right predictions

$$p(T|w_1, \dots, w_K) = \prod_{n=1}^N \prod_{k=1}^K p(C_k|\phi_n)^{t_{nk}}$$

$$E(w_1, \dots, w_K) = -\ln p(T|w_1, \dots, w_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

By taking gradient with respect to w_j , now we can apply gradient descent to find the optimal linear combination of w and x .

$$\nabla_{w_j} E(w_1, \dots, w_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n$$

$$\rightarrow w_j = w_j - \alpha \nabla_{w_j} E(w_1, \dots, w_K)$$