Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

6. Probabilistic models for classification

L. Iocchi, F. Patrizi

The goal in classification is to take an input vector x and to assign it to one of K discrete classes Ck where k = 1, . . . , K. In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable. For regression problems, the target variable t was simply the vector of real numbers whose values we wish to predict. In the case of classification, there are various ways of using target values to represent class labels. For probabilistic models, the most convenient, in the case of two-class problems, is the binary representation in which there is a single target variable t ∈ {0, 1} such that t = 1 represents class C1 and t = 0 represents class C2. We can interpret the value of t as the probability that the class is C1, with the values of probability taking only the extreme values of 0 and 1.

Overview

- Probabilistic generative models
- Probabilistic discriminative models
- Logistic regression

References

C. Bishop. Pattern Recognition and Machine Learning. Sect. 4.2, 4.3

The simplest approach to the classification problem involves constructing a discriminant function that directly assigns each vector x to a specific class. A more powerful approach, however, models the conditional probability distribution p(Ck|x) in an inference stage, and then subsequently uses this distribution to make optimal decisions

Probabilistic Models for Classification

Consider a generic classification problem

$f\ X \to C\ \{c_1, \quad c_k\}$

Given $f : X \to C$, $D = \{(x_i, c_i)_{i=1}^n\}$ and $\mathbf{x} \notin D$, estimate

$$P(C_i | \mathbf{x}, D)$$

Simplified notation without D in the formulas.

we are interested in both compute these

$P(C_i|\mathbf{x})$: posterior, $P(\mathbf{x}|C_i)$: class-conditional densities

discriminative generative

Two families of models: for determining P(Ci|x)

posterior probability

- Generative: estimate $P(\mathbf{x}|C_i)$ and then compute $P(C_i|\mathbf{x})$ with Bayes
- Discriminative: estimate $P(C_i|\mathbf{x})$ directly

Probabilistic Generative Models

Consider first the case of two classes.

we model the class-conditional densities P(x|Ck) and the class prior P(Ck), and then we compute P(Ck|x) through Bayes' theorem

Find the conditional probability:

bayes theorem

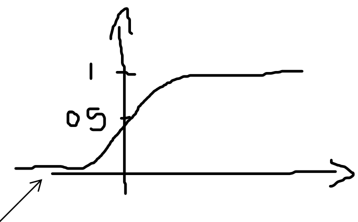
posterior probability

$$P(C_1|\mathbf{x}) = \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x}|C_1)P(C_1) + P(\mathbf{x}|C_2)P(C_2)}$$

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

with:

$$\boxed{a = \ln \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)}}$$



and

this is the definition of the sigmoid function

$$\boxed{\sigma(a) = \frac{1}{1+\exp(-a)}}$$ the *sigmoid function*.

logistic sigmoid function

This type of function is sometimes also called a "squashing function" because it maps the whole real axis into a finite interval.

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

Probabilistic Generative Models

we assume that probability distribution in gaussian, and the posterior prob can be described as a sigmoid func applied to a class conditional prob

gaussian model parameterized with mean and covariance matrix

Assume $p(\mathbf{x}|C_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \mathbf{\Sigma})$ - same covariance matrix

we shall assume that all classes share the same covariance matrix

$$a = \ln \frac{\boxed{p(\mathbf{x}|C_1)}P(C_1)}{\boxed{p(\mathbf{x}|C_2)}P(C_2)} = \ln \frac{\boxed{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \mathbf{\Sigma})}P(C_1)}{\boxed{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_2, \mathbf{\Sigma})}P(C_2)} = \ldots = \boxed{\mathbf{w}^T\mathbf{x} + w_0}$$

with:

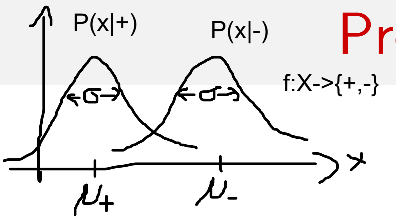
$$\boxed{\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),}$$

$$\boxed{w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{P(C_1)}{P(C_2)}.}$$

Thus

$$\boxed{P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0),}$$

Probabilistic Generative Models



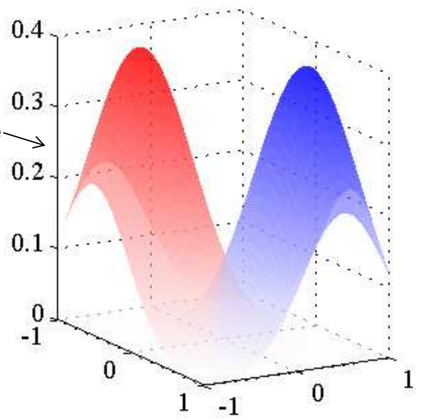
f:X->{+,-}

P(+|x') > 0.5 +

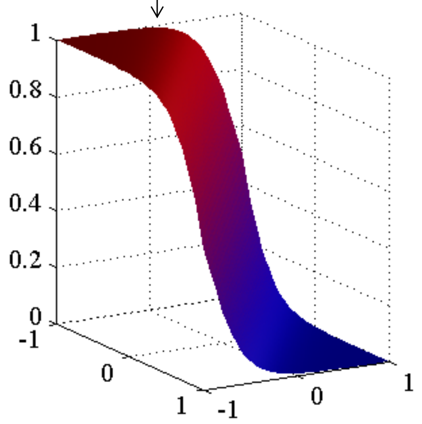
P(+|x') < 0.5 -

this is the corresponding posterior probability p(C1|x), which is given by a logistic sigmoid of a linear function of x. The surface is coloured using a proportion of red ink given by p(C1|x) and a proportion of blue ink given by p(C2|x) = 1 − p(C1|x).

this plot shows the class-conditional densities for two classes, denoted red and blue



$P(\mathbf{x}|C_1), P(\mathbf{x}|C_2)$



$P(C_1|\mathbf{x})$

Decision rule: $c = C_1 \iff P(c = C_1|\mathbf{x}) > 0.5$

Probabilistic Generative Models Multi-class

K classes

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{\sum_j P(\mathbf{x}|C_j)P(C_j)} = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

(normalized exponential or softmax function)

with $a_k = \ln P(\mathbf{x}|C_k)P(C_k)$

Once we have specified a parametric functional form for the class-conditional densities p(x|Ck), we can then determine the values of the parameters, together with the prior class probabilities p(Ck), using maximum likelihood.

Maximum likelihood

if we have k classes -> P(c_k)=1-sum(pi_j)

pi1...pik are not independent but there is a sort of constraint between they

Maximum likelihood solution for 2 classes

prior class probability

Assuming $P(C_1) = \pi$ (thus $P(C_2) = 1 - \pi$), $\boxed{P(\mathbf{x}|C_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma})}$

Given data set $D = \{(\mathbf{x}_n, t_n)_{n=1}^N\}$, $t_n = 1$ if $\mathbf{x}_n$ belongs to class C_1, $t_n = 0$ if $\mathbf{x}_n$ belongs to class C_2

Let N_1 be the number of samples in D belonging to C_1 and N_2 be the number of samples in C_2 ($N_1 + N_2 = N$)

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

Likelihood function

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}).$$

it's a vector of all the output values in the dataset (N components)

$$P(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}, D) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{(1-t_n)}$$

Unknown π, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}$ → argmax P(t|...,D) → argmax log P(t|...,D)

Maximum likelihood

As usual, it is convenient to maximize the log of the likelihood function. Consider first the maximization with respect to π.

Maximum likelihhod solution for 2 classes

Maximizing log likelihood function, we obtain

$$\boxed{\pi = \frac{N_1}{N}}$$

number of ex classified in C1

number of ex

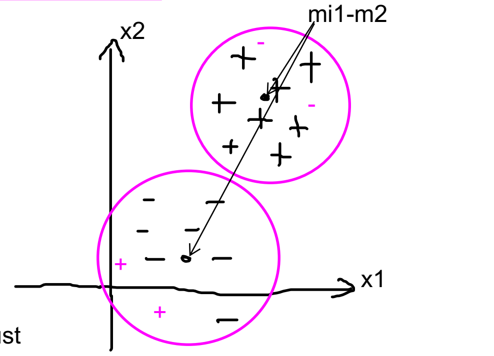
1 if xn belongs to C1

0 if xn belongs to C2

$$\boxed{\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n} \qquad \boxed{\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1-t_n)\mathbf{x}_n}$$

in this way in mi1 we will have only samples from C1 and for mi2 only samples from C2

$$\boxed{\boldsymbol{\Sigma} = \frac{N_1}{N}S_1 + \frac{N_2}{N}S_2}$$



with $S_i = \frac{1}{N_i}\sum_{n\in C_i}(\mathbf{x}_n - \boldsymbol{\mu}_i)(\mathbf{x}_n - \boldsymbol{\mu}_i)^T$, $i = 1, 2$

Note: details in C. Bishop. PRML. Section 4.2.2

if we have some noise (pink + e -), it doesn't change much the situation -> this method is robust

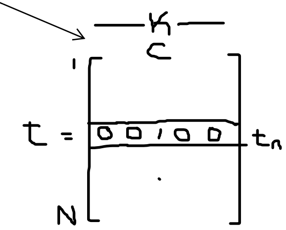
Maximum likelihood for K classes

Gaussian Naive Bayes

$P(C_k) = \pi_k,\ P(\mathbf{x}|C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})$

t is not more a vector but it is a matrix NxK (one hot encoding)

Data set $D = \{(\mathbf{x}_n, \mathbf{t}_n)_{n=1}^N\}$, with $\mathbf{t}_n$ 1-of-K encoding



$$\boxed{\pi_k = \frac{N_k}{N}}$$

$$\boxed{\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk}\, \mathbf{x}_n}$$

$$\boxed{\boldsymbol{\Sigma} = \sum_{k=1}^{K} \frac{N_k}{N} S_k\ ,} \quad \boxed{S_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}$$

weighted average

Probabilistic Models

Represent posterior distributions with parametric models.

For two classes

$$P(C_1|\mathbf{x}) = \sigma(a)$$

For $k \geq 2$ classes

$$P(C_i|\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

$$a_k = \mathbf{w}^T\mathbf{x} + w_0$$

This is valid for all the class-conditional distributions in the exponential family (including Gaussians). [Bishop, Sect. 4.2.4]

Compact notation

$$\boxed{\begin{gathered} \mathbf{w}^T\mathbf{x} + w_0 = (w_0\ \mathbf{w}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \\ \tilde{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}, \tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \\ a_k = \mathbf{w}^T\mathbf{x} + w_0 = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}} \end{gathered}}$$

Maximum Likelihood for parametric models

probability that given D the model generates output t

Likelihood for a parametric model $\mathcal{M}_\Theta$: $P(\mathbf{t}|\Theta, D)$, $D = \langle \mathbf{X}, \mathbf{t} \rangle$

Maximum likelihood solution:

matrix representing the input samples in the data

$$\Theta^* = \operatorname*{argmax}_\Theta \ln P(\mathbf{t}|\Theta, \mathbf{X})$$

maximize the log likelihood

the likelihood can be expressed as a linear combination

When $\mathcal{M}_\Theta$ belongs to the exponential family, likelihood $P(\mathbf{t}|\Theta, \mathbf{X})$ can be expressed in the form $P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$, with maximum likelihood

$$\tilde{\mathbf{w}}^* = \operatorname*{argmax}_{\tilde{\mathbf{w}}} \ln P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$$

Probabilistic Discriminative Models

the goal in discriminative models is to directly estimate the posterior without considering the class conditional densities -> we can ignore what are the params of the distribution and use only w

Estimate directly

However, an alternative approach is to use the functional form of the generalized linear model explicitly and to determine its parameters directly by using maximum likelihood.

$$P(C_{i}|\tilde{\mathbf{x}}, D) = \frac{exp(a_k)}{\sum_j exp(a_j)} \qquad a_k = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

k

with maximum likelihood

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmax}} \ln P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$$

in the next slide we omit X just for simplifing the ntation

without estimating the model parameters.

Simplified notation (dataset omitted): $P(\mathbf{t}|\tilde{\mathbf{w}})$

Logistic regression

Probabilistic discriminative model based on maximum likelihhod.

Two classes

tn=1 xn belongs to C1
tn=0 xn belongs to C2

Given data set $D = \{(\tilde{\mathbf{x}}_n, t_n)_{n=1}^N\}$, with $t_n \in \{0, 1\}$

Likelihood function:

$$\boxed{p(\mathbf{t}|\tilde{\mathbf{w}}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}}$$

product is an assumption of the independence of the data

it's a scalar in this case

with $y_n = p(C_1|\tilde{\mathbf{x}}_n) = \sigma(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_n \cancel{+ w_0})$

actual value in the datset (estimation for xn given w)

Note: t_n: value in the data set corresponding to $\mathbf{x}_n$,
y_n: posterior prediction of the current model $\tilde{\mathbf{w}}$ for $\mathbf{x}_n$.

Logistic regression

maximize the log likelihood is the same of minimize the cross-entropy error because is the negative log likelihood

Cross-entropy error function (negative log likelihood)

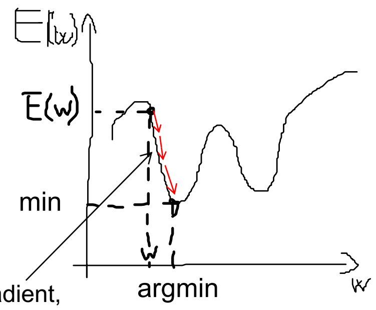
As usual, we can define an error function by taking the negative logarithm of the likelihood, which gives the crossentropy error function in the form

$\sigma(a_n) = \sigma(w^T \tilde{x}_n)$

$$\boxed{E(\tilde{\mathbf{w}}) \stackrel{\text{def}}{\equiv} -\ln p(\mathbf{t}|\tilde{\mathbf{w}}) = -\sum_{n=1}^{N} [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)]}$$

Solution concept: solve the optimization problem

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}} E(\tilde{\mathbf{w}})$$



Many solvers available.

we start from a random value of w and E(w), then we compute the gradient, that is descending, until we find the minimum

Iterative reweighted least squares

Apply *Newton-Raphson* iterative optimization for minimizing $E(\tilde{\mathbf{w}})$.

Gradient of the error with respect to $\tilde{\mathbf{w}}$

Taking the gradient of the error function with respect to w, we obtain:

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

Gradient descent step

this means that we are computing an iterative process

$$\tilde{\mathbf{w}} \boxed{\leftarrow} \tilde{\mathbf{w}} - \boldsymbol{H}(\tilde{\mathbf{w}})^{-1} \nabla E(\tilde{\mathbf{w}})$$

The Newton-Raphson update, for minimizing a function E(w), takes the form

$\boldsymbol{H}(\tilde{\mathbf{w}}) = \nabla\nabla E(\tilde{\mathbf{w}})$ is the Hessian matrix of $E(\tilde{\mathbf{w}})$ (second derivatives with respect to $\tilde{\mathbf{w}}$).

whose elements comprise the second derivatives of E(w) with respect to the components of w.

Iterative reweighted least squares

$\overset{d+1}{N\begin{pmatrix} \cdot & \cdots \\ \vdots & \cdots \end{pmatrix}}\tilde{x}_n$

Given

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_1^T \\ \cdots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} \quad \mathbf{t} = \begin{pmatrix} t_1 \\ \cdots \\ t_N \end{pmatrix},$$

$y_n = \tilde{w}^T \tilde{x}_n$

Nxd
or Nxd+1

$\mathbf{y}(\tilde{\mathbf{w}}) = (y_1, \ldots, y_n)^T$ posterior predictions of model $\tilde{\mathbf{w}}$

NxN

$\boldsymbol{R}(\tilde{\mathbf{w}})$: diagonal matrix with $R_{nn} = y_n(1 - y_n)$

each component on the diagonal is equal to this

n is the dimension of the dataset

we have

The gradient and Hessian of this error function are given by

$$\nabla E(\tilde{\mathbf{w}}) = \tilde{\mathbf{X}}^T(\mathbf{y}(\tilde{\mathbf{w}}) - \mathbf{t})$$

$$\boldsymbol{H}(\tilde{\mathbf{w}}) = \nabla\nabla E(\tilde{\mathbf{w}}) = \sum_{n=1}^{N} y_n(1 - y_n)\tilde{\mathbf{x}}_n\tilde{\mathbf{x}}_n^T = \tilde{\mathbf{X}}^T \boldsymbol{R}(\tilde{\mathbf{w}})\tilde{\mathbf{X}}$$

Iterative reweighted least squares

Iterative method:

1. Initialize $\tilde{\mathbf{w}}$ with a random value
2. Repeat until termination condition

$$\boxed{\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - (\tilde{\mathbf{X}}^T \boldsymbol{R}(\tilde{\mathbf{w}})\, \tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\mathbf{y}(\tilde{\mathbf{w}}) - \mathbf{t})}$$

Multiclass logistic regression

In our discussion of generative models for multiclass classification, we have seen that for a large class of distributions, the posterior probabilities are given by a softmax transformation of linear functions of the feature variables

K classes

y_{nk} j=1...K

activation

$$\boxed{P(C_k|\tilde{\mathbf{x}}) = \frac{exp(a_k)}{\sum_j exp(a_j)}} \qquad \boxed{a_k = \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}} \qquad k = 1, \ldots, K$$

class k

tn = (0 0 0 1 0 0 0) vector of K components

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_1^T \\ \cdots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} \qquad \underset{N*K}{\mathbf{T}} = \begin{pmatrix} t_1^T \\ \cdots \\ t_N^T \end{pmatrix}$$

1-of-K encoding of labels

$\mathbf{y}_n^T = (y_{n1} \ldots y_{nK})^T$ posterior prediction of $\tilde{\mathbf{x}}_n$ for model $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K$

vector of K components

$$\underset{NxK}{\mathbf{Y}}(\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K) = \begin{pmatrix} \mathbf{y}_1^T \\ \cdots \\ \mathbf{y}_N^T \end{pmatrix}$$

posterior predictions of model $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K$

Multiclass logistic regression

Discriminative model likelihood for K classes

$$P(\mathbf{T}|\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k|\tilde{\mathbf{x}}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

with $y_{nk} = \mathbf{Y}[n, k]$ and $t_{nk} = \mathbf{T}[n, k]$.

Multiclass logistic regression

Cross-entropy error function for the multiclass classification problem

$$E(\tilde{\mathbf{w}}_1, \ldots \tilde{\mathbf{w}}_K) = -\ln P(\mathbf{T}|\tilde{\mathbf{w}}_1, \ldots \tilde{\mathbf{w}}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

Iterative algorithm

gradient $\nabla_{\tilde{\mathbf{w}}_j} E(\tilde{\mathbf{w}}_1, \ldots \tilde{\mathbf{w}}_K) = \ldots$

Hessian matrix $\nabla_{\tilde{\mathbf{w}}_k} \nabla_{\tilde{\mathbf{w}}_j} E(\tilde{\mathbf{w}}_1, \ldots \tilde{\mathbf{w}}_K) = \ldots$

Summary

Given a target function $f : X \to C$, and data set D

assume a parametric model for the posterior probability $P(C_k|\tilde{\mathbf{x}}, \tilde{\mathbf{w}})$
$\sigma(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$ (2 classes) or $\frac{exp(\tilde{\mathbf{w}}_k^T\tilde{\mathbf{x}})}{\sum_{j=1}^{K} exp(\tilde{\mathbf{w}}_j^T\tilde{\mathbf{x}})}$ (k classes)

Define an error function $E(\tilde{\mathbf{w}})$ (negative log likelihood)

Solve the optimization problem

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}}\, E(\tilde{\mathbf{w}})$$

Classify new sample $\tilde{\mathbf{x}}'$ as C_{k^*} where $k^* = \operatorname{argmax}_{k=1,\ldots,K} P(C_k|\tilde{\mathbf{x}}', \tilde{\mathbf{w}}^*)$

Generalization

Given a target function $f : X \to C$, and data set D

assume a prediction parametric model $y(\mathbf{x}; \theta)$, $\quad y(\mathbf{x}; \theta) \approx f(\mathbf{x})$

Define an error function $E(\theta)$

Solve the optimization problem

$$\theta^* = \operatorname*{argmin}_{\theta} E(\theta)$$

Classify new sample $\mathbf{x}'$ as $y(\mathbf{x}'; \theta^*)$

Learning in feature space

All methods described above can be applied in a transformed space of the input (*feature space*).

Given a function $\phi : \tilde{\mathbf{x}} \mapsto \mathbf{\Phi}$ ($\mathbf{\Phi}$ is the *feature space*)
each sample $\tilde{\mathbf{x}}_n$ can be mapped to a feature vector $\boldsymbol{\phi}_n = \phi(\tilde{\mathbf{x}}_n)$

Replacing $\tilde{\mathbf{x}}_n$ with $\boldsymbol{\phi}_n$ in all the equations above, makes the learning system to work in the feature space instead of the input space.

We will see in the next lectures why this trick is useful.