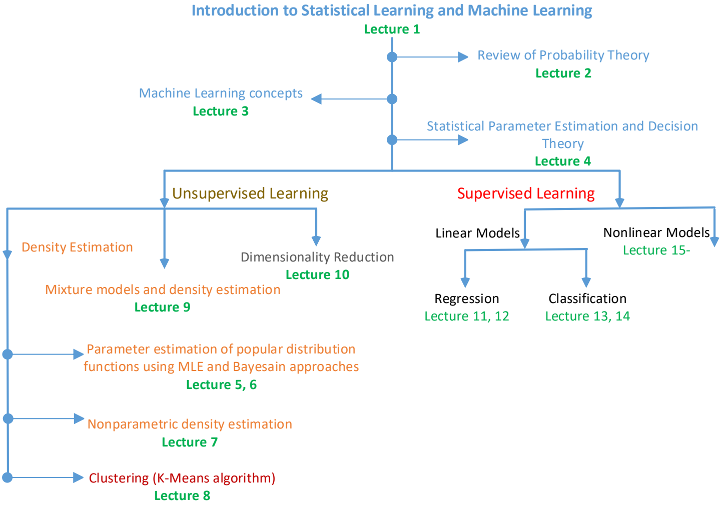
Statistical Learning and Machine Learning

Lecture 14 - Linear Models for Classification 2

October 10, 2021

Course overview and where do we stand



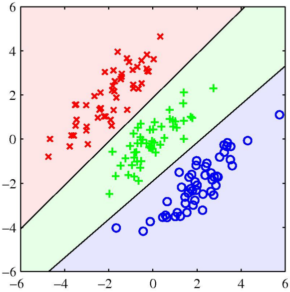
Objectives of the lecture

- Linear models for classification
  - Probabilistic generative models: linear discriminant analysis
  - Probabilistic discriminative models: Logistic regression

Linear classification

The goal of classification is to assign an input vector $\boldsymbol{x}$ to one of the K classes $\mathcal{C}_k,\ k = 1, \ldots, K$:

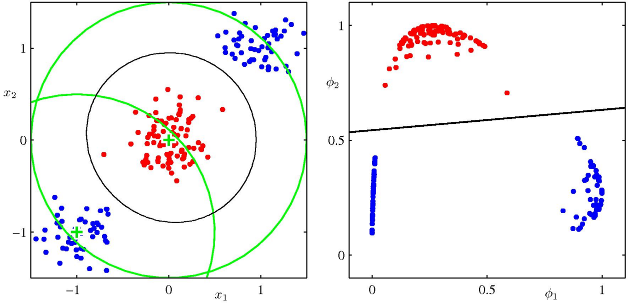
- The input space is divided into K decision regions, separated by decision boundaries or decision surfaces
- **Linear models**: are those for which decision boundaries are linear functions of $\boldsymbol{x}$ i.e., $\boldsymbol{w}^T \boldsymbol{x} + w_0$



Decision boundaries for a linearly separable data set

Generalized linear classification

- Generalized linear classification corresponds to using linear combination of (nonlinearly) transformed vectors $\phi(\boldsymbol{x})$ i.e., $y(\boldsymbol{x}) = f(\boldsymbol{w}^T\phi(\boldsymbol{x}))$.
- Appropriately chosen linear models of basis functions $\phi(\boldsymbol{x})$ can be equivalent to nonlinear models in $\boldsymbol{x}$



Three approaches to classification

1. Finding a function $y(\boldsymbol{x})$ called discriminant function which maps a new input $\boldsymbol{x}_*$ onto a class label.
2. **Generative models:** Determining class-conditional densities $p(\boldsymbol{x}|\mathcal{C}_k)$ or joint distributions $p(\boldsymbol{x}, \mathcal{C}_k)$, followed by the estimation of posterior densities:

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

   Finally, decision theory is used to determine class membership for new $\boldsymbol{x}$.
3. **Discriminative models:** Obtaining posterior class probabilities $p(\mathcal{C}_k|\boldsymbol{x})$ directly, followed by discrimination.

Probabilistic Generative Models

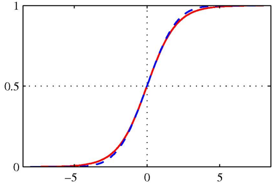
Consider a two-class problem; the posterior probability for $\mathcal{C}_1$ is

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1 + exp(-\alpha)} = \sigma(\alpha) \quad (17)$$

where:

$$\alpha = \ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \quad (18)$$

and $\sigma(\alpha)$ is the *logistic sigmoid* function.



Logistic sigmoid function

Properties:

- squashing function: maps the whole real axis into a finite interval.
- $\sigma(-\alpha) = 1 - \sigma(\alpha)$
- $d\sigma/d\alpha = \sigma(1-\sigma)$
- Inverse (known as *logit function*): $\alpha = \ln\left(\frac{\sigma}{1-\sigma}\right)$
- α represents the log of the ratio of conditional probabilities for the two classes:

$$\alpha(\boldsymbol{x}) = \ln\left(\frac{p(\mathcal{C}_1|\boldsymbol{x})}{p(\mathcal{C}_2|\boldsymbol{x})}\right) \tag{19}$$

also known as the *log odds*.

Softmax function

For $K > 2$ we obtain the normalized exponential:

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{j=1}^{K} p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{exp(\alpha_k)}{\sum_{j=1}^{K} exp(\alpha_j)} \tag{20}$$

where:

$$\alpha_k = \ln p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k) \tag{21}$$

The normalized exponential (20) is also called softmax function:

- when $\alpha_k \gg \alpha_j$ for all $j \neq k$, then $p(\mathcal{C}_k|\boldsymbol{x}) \simeq 1$ and $p(\mathcal{C}_j|\boldsymbol{x}) \simeq 0$

Probabilistic Generative Models

Assuming that i) $K = 2$; ii) the class-conditional densities are Gaussian; and iii) there is a common covariance matrix $\Sigma_k = \Sigma,\ k = 1, 2$, the class-conditional density of $\boldsymbol{x}$ is

$$p(\boldsymbol{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\right\} \tag{22}$$

where D is the dimensionality of $\boldsymbol{x}$.

Substituting to the above expression for $p(\mathcal{C}_1|\boldsymbol{x})$:

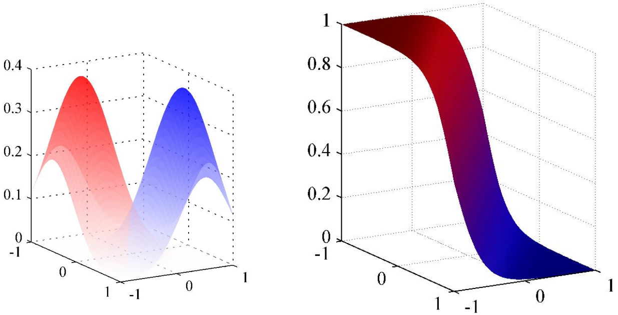
$$p(\mathcal{C}_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0) \tag{23}$$

where:

$$\begin{aligned} \boldsymbol{w} &= \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) && (24) \\ w_0 &= -\frac{1}{2}\left(\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2\right) + \ln\left(\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}\right) && (25) \end{aligned}$$

Probabilistic Generative Models

- The resulting decision boundaries are linear in $\boldsymbol{x}$. Why?
- The prior probabilities $p(\mathcal{C}_k)$ enter through the bias parameter w_0 only, having the effect of making parallel shifts of decision boundaries.



Left: class-conditional densities for two classes (red and blue)
Right: Class posterior probability $p(\mathcal{C}_1|\boldsymbol{x})$

Probabilistic Generative Models

For $K > 2$ and again using a common covariance matrix $\Sigma_k = \Sigma,\ k = 1, \dots, K$:

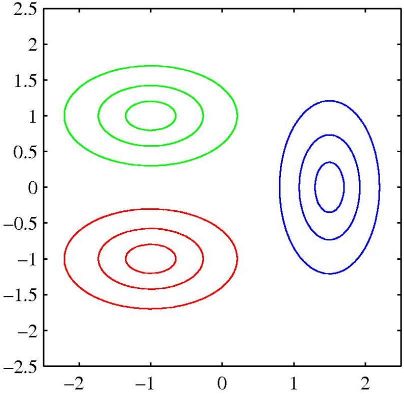
$$\alpha_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0} \tag{26}$$

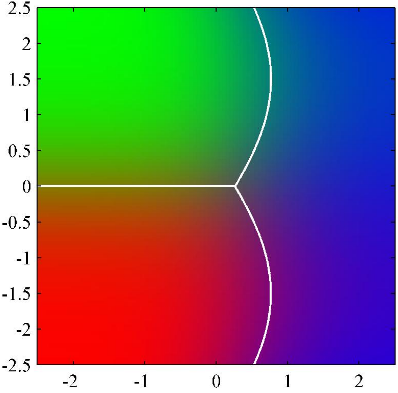
where:

$$\begin{aligned} \boldsymbol{w}_k &= \Sigma^{-1} \boldsymbol{\mu}_k && (27) \\ w_{k0} &= -\frac{1}{2} \boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k + \ln p(\mathcal{C}_k). && (28) \end{aligned}$$

In the above, using different Σ_k for each class will lead to quadratic functions (discriminants) of $\boldsymbol{x}$.

Probabilistic Generative Models





$\Sigma_1 = \Sigma_2$
$\Sigma_3 \neq \Sigma_j,\ j = 1, 2.$

Probabilistic Generative Models: Maximum Likelihood

- Given: Functional form for $p(\boldsymbol{x}, \mathcal{C}_k)$ together with training data $\{\boldsymbol{x}_n, t_n\}$ where $n = 1, \ldots, N$. Here, $t_n = 1$ for class $\mathcal{C}_1$) and $t_n = 0$ for class $\mathcal{C}_2$).
- Aim: Estimating parameters of the $p(\boldsymbol{x}, \mathcal{C}_k)$ along with prior class probabilities $p(\mathcal{C}_k)$ via ML
- Likelihood Function:

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

where $\boldsymbol{t} = (t_1, \ldots, t_n)^T$.

Probabilistic Generative Models: Maximum Likelihood

- The ML estimate of $\boldsymbol{\mu}_k, \ k = 1, 2$ and Σ are:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \boldsymbol{x}_n \tag{29}$$

- The ML estimate of Σ is:

$$\Sigma = \boldsymbol{S} = \frac{N_1}{N} \boldsymbol{S}_1 + \frac{N_2}{N} \boldsymbol{S}_2 \tag{30}$$

where

$$\boldsymbol{S}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T \tag{31}$$

$$\boldsymbol{S}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T \tag{32}$$

- The ML estimate of the prior class probabilities are

$$p(\mathcal{C}_1) = \pi = \frac{N_1}{N_1 + N_2}$$

Generative vs Discriminative Models

Until now we followed a generative approach to estimate posterior class probabilities:

$$\{p(\boldsymbol{x}|\mathcal{C}_k), p(\mathcal{C}_k)\} \Rightarrow p(\mathcal{C}_k|\boldsymbol{x}). \tag{33}$$

where we used the fact that

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

where: where:

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

Linear discriminant functions:

$$y_k(\boldsymbol{x}) = w_k^T \boldsymbol{x} + w_0$$

Generative vs Discriminative Models

When we train discriminative models:

- we use a function form of $p(\mathcal{C}_k|\boldsymbol{x}) = \sigma(\boldsymbol{w}_k^T\boldsymbol{x} + w_0)$ explicitly without considering class conditional densities $p(\boldsymbol{x}|\mathcal{C}_k)$
- we optimize/estimate the parameters of the above function directly using training data.
- Pros and Cons?

Fixed basis functions

Sometimes it is beneficial to use a (fixed) nonlinear transformation of the data $\boldsymbol{x} \rightarrow \phi$ before applying the linear discriminative model.

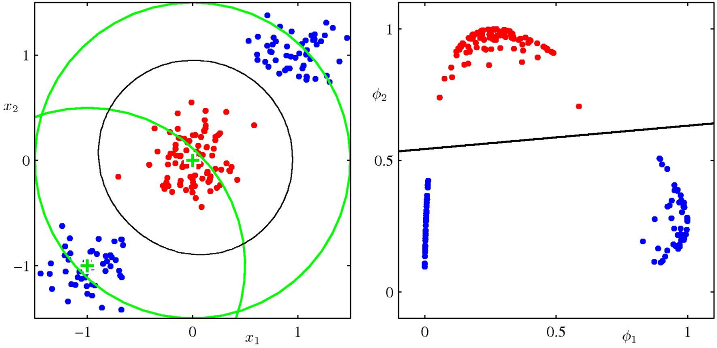


Illustration of the role of nonlinear basis in linear classification models. Use of 2 Gaussian basis functions centered at the green crosses converts the nonlinearly separable data (in $\boldsymbol{x}$) to linearly separable data in $\boldsymbol{\phi}$.

Logistic Regression

- Considering a 2-class classification problem, the posterior probability of class 1 can be written as a logistic sigmoid acting on a linear function of ϕ:

$$\begin{aligned} p(\mathcal{C}_1|\phi(\boldsymbol{x}_n)) &= y(\phi(\boldsymbol{x}_n)) = \sigma(\boldsymbol{w}^T\phi(\boldsymbol{x}_n)) \\ p(\mathcal{C}_2|\phi(\boldsymbol{x}_n)) &= 1 - p(\mathcal{C}_1|\phi(\boldsymbol{x}_n)) \end{aligned}$$

- Goal: For a set of data $\{\phi(\boldsymbol{x}_n), t_n\}$, $n = 1, \ldots, N$ with $t_n \in \{0, 1\}$ and $\boldsymbol{t} = [t_1, \ldots, t_N]^T$, the goal is to estimate the optimal parameters $\boldsymbol{w}$ using ML.
- Likelihood function:

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{w}) &= \prod_{n=1}^{N} (p(\mathcal{C}_1|\phi_n))^{t_n} (p(\mathcal{C}_2|\phi_n))^{1-t_n} \\ &= \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \end{aligned}$$

where $y_n = p(\mathcal{C}_1|\phi(\boldsymbol{x}_n))$

Logistic Regression

- We define the error function as the negative log-likelihood, which is called *cross-entropy* error function (suitable for minimization):

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}|\boldsymbol{w}) = -\sum_{n=1}^{N}\Big(t_n \ln y_n + (1-t_n)\ln(1-y_n)\Big) \quad (34)$$

  where $y_n = \sigma(\alpha_n)$ and $\alpha_n = \boldsymbol{w}^T\phi(\boldsymbol{x}_n)$.
- We optimize the parameters $\boldsymbol{w}$ by applying stochastic gradient descent. But why not obtain $\boldsymbol{w}$ directly as in LS?
- The gradient of error function w.r.t $\boldsymbol{w}$ is

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi(\boldsymbol{x}_n) \quad (35)$$

Iterative Reweighted Least Squares

Reminder: Newton-Raphson method defines the update rule:

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w}^{(\tau)}) \tag{36}$$

where $\boldsymbol{H}$ is the *Hessian matrix* having elements $H_{ij} = \frac{\theta^2 E_n(\boldsymbol{w})}{\theta w_i \theta w_j}$.

Using the cross-entropy error function:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi(\boldsymbol{x}_n) = \Phi^T(\boldsymbol{y} - \boldsymbol{t}) \tag{37}$$

$$\boldsymbol{H} = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^T = \Phi^T\boldsymbol{R}\Phi, \tag{38}$$

where $\boldsymbol{R} = diag(y_n(1-y_n))$ is a $N \times N$ square diagonal depending on $\boldsymbol{w}$.

Iterative Reweighted Least Squares

Using the property $0 < y_n < 1$:

- $\boldsymbol{H}$ is positive definite: for any $\boldsymbol{u}$ the value $\boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} > 0$
- the error function $E(\boldsymbol{w})$ is a *convex* function of $\boldsymbol{w}$
- the error function has a unique (global) minimum.

The iterative update rule (due to dependence of $\boldsymbol{R}$ on $\boldsymbol{w}$) is:

$$\begin{aligned} \boldsymbol{w}^{(\tau+1)} &= \boldsymbol{w}^{(\tau)} - (\Phi^T \boldsymbol{R} \Phi)^{-1} \Phi^T (\boldsymbol{y} - \boldsymbol{t}) \\ &= (\Phi^T \boldsymbol{R} \Phi)^{-1} \Big(\Phi^T \boldsymbol{R} \Phi \boldsymbol{w}^{\tau} - \Phi^T (\boldsymbol{y} - \boldsymbol{t}) \Big) \\ &= (\Phi^T \boldsymbol{R} \Phi)^{-1} \Phi^T \boldsymbol{R} \boldsymbol{z} \end{aligned} \tag{39}$$

where $\boldsymbol{z} \in \mathbb{R}^N$:

$$\boldsymbol{z} = \Phi \boldsymbol{w}^{(\tau)} - \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{t}). \tag{40}$$

Reminder:

$$p(\mathcal{C}_k|\phi(\boldsymbol{x}_n)) = y_k(\phi(\boldsymbol{x}_n)) = \frac{exp(\alpha_k)}{\sum_{j=1}^{K} exp(\alpha_j)} \tag{41}$$

where $\alpha_k = \boldsymbol{w}_k\phi(\boldsymbol{x}_n)$ and $\partial y_k / \partial \alpha_j = y_k(I_{kj} - y_j)$ with I_{kj} being the kj-th element of the identity matrix.

Using 1-of-K coding for the target vectors $\boldsymbol{t}_n$ forming $\boldsymbol{T} \in \mathbb{R}^{N \times K}$, the likelihood function is:

$$p(\boldsymbol{T}|\boldsymbol{w}_1, \dots, \boldsymbol{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p\Big(\mathcal{C}_k|\phi(\boldsymbol{x}_N)\Big)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}} \tag{42}$$

with $y_{nk} = y_k(\phi(\boldsymbol{x}_n))$, and t_{nk} is the $\{n, k\}$-th element of $\boldsymbol{T}$.

Multiclass logistic regression

The negative log-likelihood, which is called *cross-entropy* error function, (suitable for minimization) is:

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\ln p(\boldsymbol{T}|\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk} \qquad (43)$$

Derivatives of $E(\boldsymbol{w})$:

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\phi(\boldsymbol{x}_n) \qquad (44)$$

$$\nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\sum_{n=1}^{N} y_{nk}(I_{kj} - y_{nj})\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^T. \qquad (45)$$

We can use $\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$ for applying SGD-based optimization, or both $\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$ and $\nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$ to apply IRLS-based optimization.