Machine Learning

Lecture 8: Linear Classification

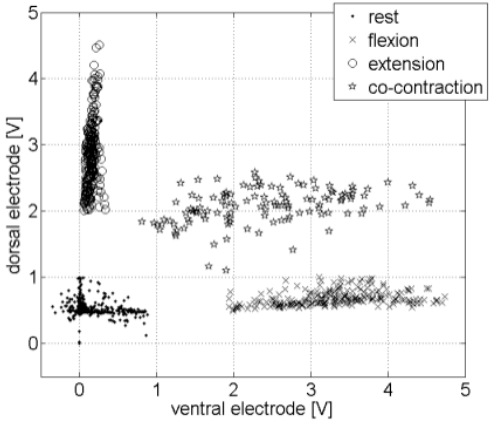
Prof. Dr. Aleksandar Bojchevski

14.05.24

Outline

Classification problem

Given features $\boldsymbol{x}_i \in \mathbb{R}^D$ and labels $y_i \in \mathcal{C} = \{1, \ldots, C\}$, find a mapping $f : \mathbb{R}^D \to \mathcal{C}$ such that $y_i = f(\boldsymbol{x}_i)$ for as many examples as possible.



Zero-one loss

How do we measure the quality of a prediction $\hat{y} = f(\boldsymbol{x})$?

The zero-one loss $l_{01}(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$ equals 1 if we have a misclassification.

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}, y_i)\}_{i=1}^N$, the total number of misclassified examples is

$$\mathcal{L} = \sum_{i=1}^{N} l_{01}(y, \hat{y}) = \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i \neq y_i) = \mathbb{I}(f(\boldsymbol{x})_i \neq y_i)$$

How do we choose a good $f(\cdot)$?

Hyperplane as a decision boundary

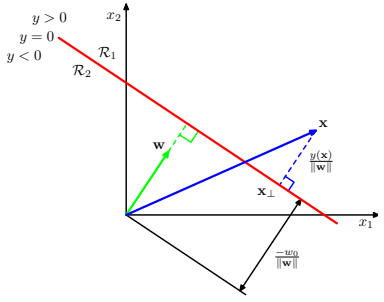
For a 2-class problem, $\mathcal{C} = \{0, 1\}$, we can try to separate the instances from the two classes by a hyperplane.

Hyperplane as a decision boundary

For a 2-class problem, $\mathcal{C} = \{0, 1\}$, we can try to separate the instances from the two classes by a hyperplane.

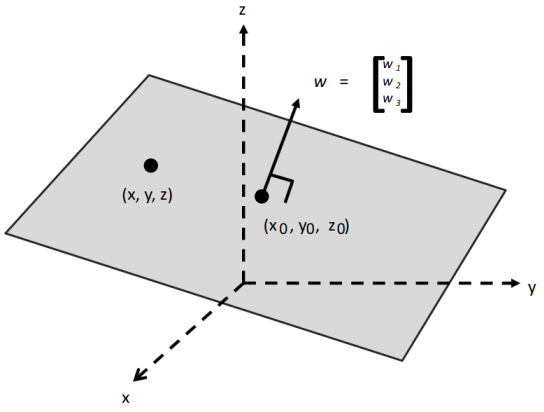
A hyperplane be defined by a normal vector $\boldsymbol{w}$ and an offset w_0.
Hyperplanes are computationally very convenient: easy to evaluate.

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \begin{cases} = 0 & \text{if } \boldsymbol{x} \text{ on the plane} \\ > 0 & \text{if } \boldsymbol{x} \text{ on normal's side} \\ < 0 & \text{else} \end{cases}$$



Hyperplane as a decision boundary

Same idea applies in higher dimensions. We have $\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = \boldsymbol{w}^T\boldsymbol{x} \underbrace{-\boldsymbol{w}^T\boldsymbol{x}_0}_{=w_0} = 0.$



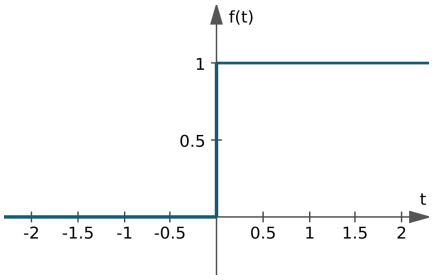
Making predictions

Decision rule

$$\hat{y} = f(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

where f is the step function defined as:

$$f(t) = \begin{cases} 1 & \text{if } t > 0, \\ 0 & \text{otherwise.} \end{cases}$$

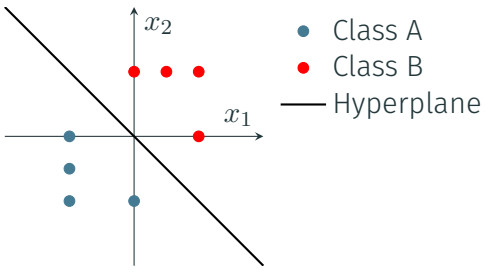


Sometimes the labels are equivalently given as $y_i \in \{1, -1\}$ and the predictions are $\hat{y}_i = \text{sign}(f(\boldsymbol{x}_i))$ where $\text{sign}(\cdot) = 1$ if the input is positive and -1 otherwise.

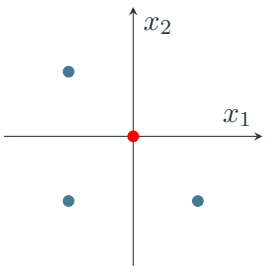
Linear separability

A data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ is linearly separable if there exists a hyperplane for which all $\boldsymbol{x}_i$ with $y_i = 0$ are on one side, and all $\boldsymbol{x}_i$ with $y_i = 1$ on the other side.

linearly separable



not linearly separable



Learning rule for the perceptron

The perceptron algorithm is one of the oldest methods for binary classification.

Initialize parameters to any value, e.g., a zero vector: $\boldsymbol{w}, w_0 \leftarrow \mathbf{0}$.

[1] However, there is no way to determine the number of required iterations in advance.

Learning rule for the perceptron

The perceptron algorithm is one of the oldest methods for binary classification.

Initialize parameters to any value, e.g., a zero vector: $\boldsymbol{w}, w_0 \leftarrow \mathbf{0}$.

For each misclassified sample $\boldsymbol{x}_i$ in the training set update

$$\boldsymbol{w} \leftarrow \begin{cases} \boldsymbol{w} + \boldsymbol{x}_i & \text{if } y_i = 1, \\ \boldsymbol{w} - \boldsymbol{x}_i & \text{if } y_i = 0. \end{cases} \qquad w_0 \leftarrow \begin{cases} w_0 + 1 & \text{if } y_i = 1, \\ w_0 - 1 & \text{if } y_i = 0. \end{cases}$$

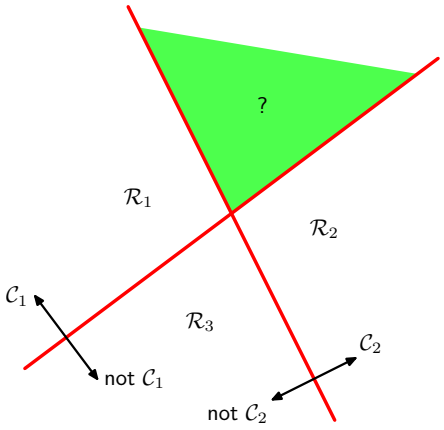
until all samples are classified correctly.

This method takes a finite[1] number of steps to converge to a $(\boldsymbol{w}, w_0)$ discriminating between two classes **if it exists** (if the data is linearly separable).

[1] However, there is no way to determine the number of required iterations in advance.

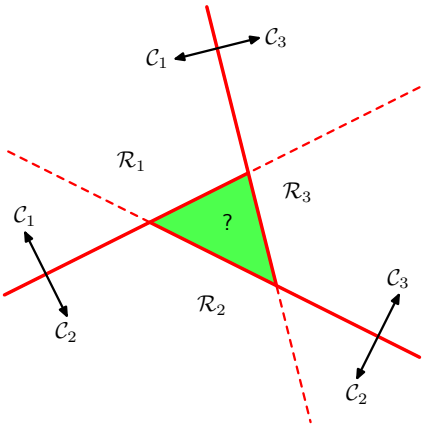
Handling multiple classes

One-versus-rest classifier: Each hyperplane $\mathcal{H}_i$ decides class $\mathcal{C}_i$ vs. not class $\mathcal{C}_i$.



Handling multiple classes

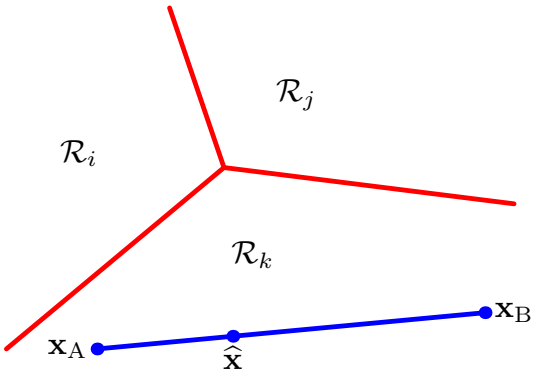
One-versus-one classifier: hyperplane $\mathcal{H}_{ij}$ decides each pair class $\mathcal{C}_i$ vs. class $\mathcal{C}_j$. Use majority vote to classify.



Handling multiple classes

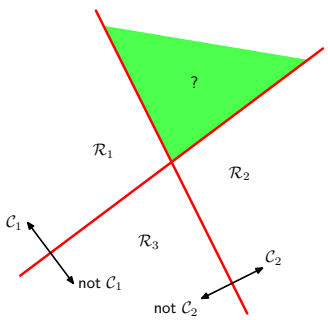
Multiclass discriminant: define C linear functions $f_c(\boldsymbol{x}) = \boldsymbol{w}_c^T \boldsymbol{x} + w_{0c}$ with the decision rule $\hat{y} = \arg\max_{c \in \mathcal{C}} f_c(\boldsymbol{x})$.

That is, assign $\boldsymbol{x}$ to the class c which produces the highest $f_c(\boldsymbol{x})$, dividing the domain into convex decision regions $\mathcal{R}_i$.

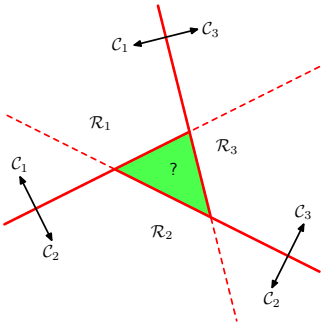


Handling multiple classes

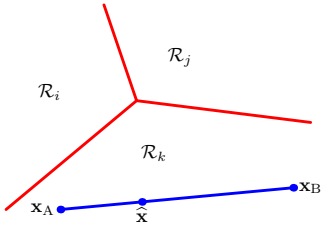
Usually we prefer the multiclass discriminant unless we have no choice.



One-vs-one

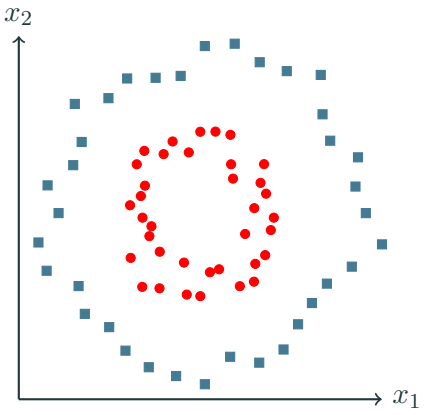


One-vs-rest



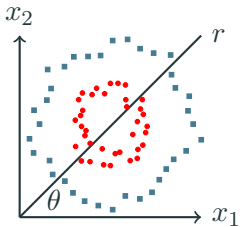
Multiclass

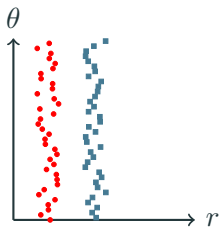
What if the classes are not linearly separable?



Basis functions

Apply a nonlinear transformation $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^M$ that maps samples to a space where they are linearly separable.



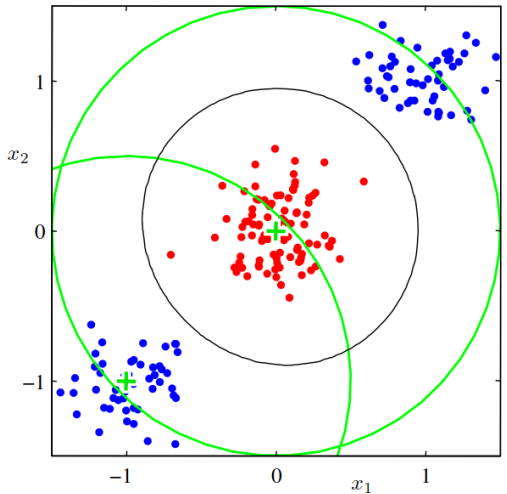


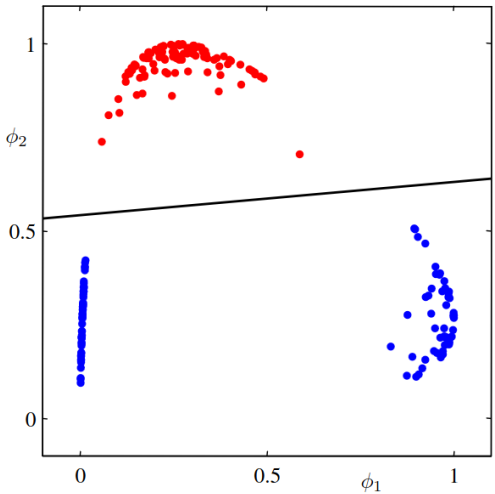
Here, $\boldsymbol{\phi}(\boldsymbol{x}) = (r, \theta) = (\|\boldsymbol{x}\|_2, \text{angle}(\boldsymbol{x}))$.

Here we talk about fixed basis functions. A deeper discussion of this method will take place in *Kernels*. Adaptive basis functions will be covered in *Deep Learning*.

Basis functions

Linear separability using two Gaussian basis functions $\phi_1(\boldsymbol{x}), \phi_1(\boldsymbol{x})$ (green).





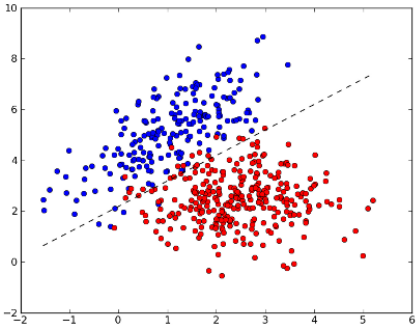
Limitations of hard-decision based classifiers

No measure of uncertainty

Cannot handle noisy data

Poor generalization

Difficult to optimize



What are the alternatives?

Probabilistic models for classification

Solution: model the distribution of the class label y given the data $\boldsymbol{x}$.

$$p(y = c \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid y = c) \cdot p(y = c)}{p(\boldsymbol{x})}$$

Two types of models:

Generative

- Model the joint distribution $p(\boldsymbol{x}, y = c) = p(\boldsymbol{x} \mid y = c) \cdot p(y = c)$

Discriminative

- Directly model the distribution $p(y = c \mid \boldsymbol{x})$

Given $p(y \mid \boldsymbol{x})$ we can make the prediction $\hat{y}$ based on our problem.
Popular choice is the mode: $\hat{y} = \arg\max_{c \in \mathcal{C}} p(y = c \mid \boldsymbol{x})$.

Outline

Hard deterministic classification

Probabilistic generative models for linear classification

Probabilistic discriminative models for linear classification

Generative model

The idea is to obtain the class posterior using Bayes' theorem

$$p(y = c \mid \boldsymbol{x}) \propto \underbrace{p(\boldsymbol{x} \mid y = c)}_{\text{class conditional}} \cdot \underbrace{p(y = c)}_{\text{class prior}}$$

The model consists of:

class prior - a priori probability of a point belonging to a class c,

class conditional - probability of observing $\boldsymbol{x}$, given that it belongs to class c.

Applying a generative model

Choose a parametric model for the class conditional $p(\boldsymbol{x} \mid y = c, \boldsymbol{\psi})$ and the class prior $p(y = c \mid \boldsymbol{\theta})$.

Learning equals estimating the parameters of our model $\{\boldsymbol{\psi}, \boldsymbol{\theta}\}$ from the data $\mathcal{D}$, e.g., using maximum likelihood to obtain estimates $\{\hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\theta}}\}$.

Once fitted, we can perform inference - classify a new $\boldsymbol{x}_{\text{new}}$ using Bayes rule

$$p(y_{\text{new}} = c \mid \boldsymbol{x}_{\text{new}}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\theta}}) \propto p(\boldsymbol{x}_{\text{new}} \mid y_{\text{new}} = c, \hat{\boldsymbol{\psi}}) p(y_{\text{new}} = c \mid \hat{\boldsymbol{\theta}})$$

Additionally, we can generate new data - hence the name:

- sample a class label $y_{\text{sample}} \sim p(y \mid \hat{\boldsymbol{\theta}})$
- sample a feature vector $\boldsymbol{x}_{\text{sample}} \sim p(\boldsymbol{x} \mid y = y_{\text{sample}}, \hat{\boldsymbol{\psi}})$

How do we choose the class prior $p(y = c)$?

The label y can take one of C discrete values $\implies$ use categorical distribution.

$$y \sim \text{Categorical}(\boldsymbol{\theta})$$

The parameter $\boldsymbol{\theta} \in \mathbb{R}^C$ specifies the probability of each class

$$p(y = c) = \theta_c \quad \text{or equivalently} \quad p(y) = \prod_{c=1}^{C} \theta_c^{\mathbb{I}(y=c)}$$

and is subject to the constraints $0 \leq \theta_c \leq 1$ and $\sum_{c=1}^{C} \theta_c = 1$.

The maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$ given the data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ is

How do we choose the class prior $p(y = c)$?

The label y can take one of C discrete values $\implies$ use categorical distribution.

$$y \sim \text{Categorical}(\boldsymbol{\theta})$$

The parameter $\boldsymbol{\theta} \in \mathbb{R}^C$ specifies the probability of each class

$$p(y = c) = \theta_c \quad \text{or equivalently} \quad p(y) = \prod_{c=1}^{C} \theta_c^{\mathbb{I}(y=c)}$$

and is subject to the constraints $0 \leq \theta_c \leq 1$ and $\sum_{c=1}^{C} \theta_c = 1$.

The maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$ given the data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ is

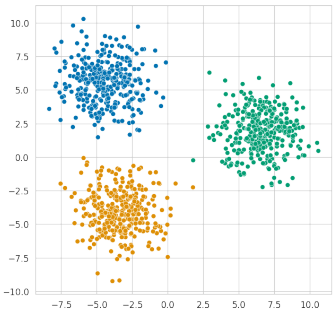
$$\hat{\theta}_c = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i = c)$$

How do we choose the class conditionals $p(\boldsymbol{x} \mid y = c)$?

For continuous features, $\boldsymbol{x} \in \mathbb{R}^D$, we can use a multivariate normal for each class.

$$p(\boldsymbol{x} \mid y = c) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)\right\}$$

We use the same $\boldsymbol{\Sigma}$ for each class, as estimating all $\boldsymbol{\Sigma}_c$'s behaves badly numerically, unless we have **lots** of data.



We will derive MLE estimates for $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_C, \boldsymbol{\Sigma}\}$ in the tutorial (see Bishop 4.2.2).

Posterior distribution

Now that we have chosen $p(\boldsymbol{x} \mid y)$ and $p(y)$, and have estimated their parameters from the training data, how do we perform classification?

Let's assume for simplicity that we have two classes $\mathcal{C} = \{0, 1\}$.

$$
\begin{aligned}
p(y = 1 \mid \boldsymbol{x}) &= \frac{p(\boldsymbol{x} \mid y = 1)\, p(y = 1)}{p(\boldsymbol{x} \mid y = 1)p(y = 1) + p(\boldsymbol{x} \mid y = 0)\, p(y = 0)} \\
&= \frac{1}{1 + \exp(-a)} =: \sigma(a)
\end{aligned}
$$

where σ is the **sigmoid** function and we defined

$$
a = \log \frac{p(\boldsymbol{x} \mid y = 1)p(y = 1)}{p(\boldsymbol{x} \mid y = 0)p(y = 0)}
$$

To avoid clutter, we implicitly condition the distributions on their respective parameters ($\boldsymbol{\theta}$, $\boldsymbol{\mu}_c$, $\boldsymbol{\Sigma}$).

Linear discriminant analysis (LDA)

For Gaussian class-conditionals with the same covariance $\boldsymbol{\Sigma}$ we have

$$\begin{aligned} a &= \log \frac{p(\boldsymbol{x} \mid y=1)p(y=1)}{p(\boldsymbol{x} \mid y=0)p(y=0)} \\ &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \log p(y=1) \\ &\quad + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0) - \log p(y=0) \\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \end{aligned}$$

where we define

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + \log \frac{p(y=1)}{p(y=0)} \end{aligned}$$

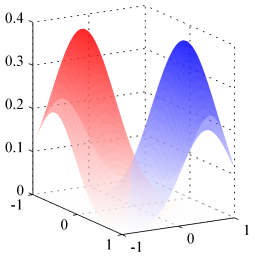
LDA for $C = 2$ classes

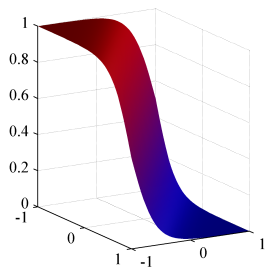
This means, that the posterior distribution is a sigmoid of a linear function of $\boldsymbol{x}$

$$p(y = 1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-(\boldsymbol{w}^T \boldsymbol{x} + w_0))} = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

Or equivalently $y \mid \boldsymbol{x} \sim \text{Bernoulli}(\sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0))$.

For $D = 2$, $p(\boldsymbol{x} \mid y = 1)$ in red, and $p(\boldsymbol{x} \mid y = 0)$ in blue. Right shows $p(y = 1 \mid \boldsymbol{x})$.





LDA for $C > 2$ classes

Using Bayes' theorem, the posterior for the $C > 2$ case is

$$p(y = c \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid y = c)p(y = c)}{\sum_{c'=1}^{C} p(\boldsymbol{x} \mid y = c')p(y = c')}.$$

Working out the math, we get

$$= \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x} + w_{c0})}{\sum_{c'=1}^{C} \exp(\boldsymbol{w}_{c'}^T \boldsymbol{x} + w_{c'0})}$$

where

$$\boldsymbol{w}_c = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c$$
$$w_{c0} = -\frac{1}{2} \boldsymbol{\mu}_c^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \log p(y = c)$$

Softmax function

On the previous slide we arrived at the softmax function defined as

$$\sigma(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{k=1}^{K} \exp(x_k)}$$

Softmax σ is a generalization of sigmoid to multiple dimensions

$$\sigma : \mathbb{R}^K \to \triangle^{K-1}$$

where

$$\triangle^{K-1} = \left\{ \boldsymbol{x} \in \mathbb{R}^K | \sum_{k=1}^{K} x_k = 1 \text{ and } x_k \geq 0, k = 1, ..., K \right\}$$

is the standard **probability simplex**.

Class conditionals: Variant II (Naive Bayes)

Naive Bayes: Assume that the D features of a sample $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ are conditionally independent given the class, i.e.

$$p(x_1, x_2, \ldots, x_d | y = c) = \prod_{i=1}^{d} p(x_i | y = c)$$

In the case of continuous data where the likelihood is assumed to be a normal distribution, this corresponds to **diagonal** covariance matrices, i.e.

$$p(\boldsymbol{x} | y = c) = \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

Important: We use a different (diagonal) covariance matrix $\boldsymbol{\Sigma}_c$ for each class c. Compare to LDA where all class conditionals share the same covariance matrix $\boldsymbol{\Sigma}$.

Assume two classes $\mathcal{C} = \{0, 1\}$. Like in LDA we need to compute

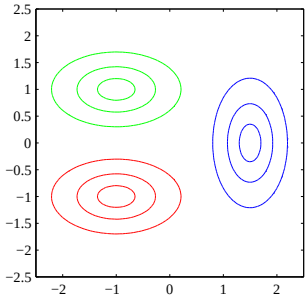
$$\begin{aligned} a &= \log \frac{p(\boldsymbol{x} \mid y=1)p(y=1)}{p(\boldsymbol{x} \mid y=0)p(y=0)} \\ &= \frac{1}{2}\boldsymbol{x}^T[\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}]\boldsymbol{x} + \boldsymbol{x}^T[\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0] \\ &\quad - \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + \log\frac{\pi_1}{\pi_0} + \frac{1}{2}\log\frac{|\boldsymbol{\Sigma}_0|}{|\boldsymbol{\Sigma}_1|} \\ &= \boldsymbol{x}^T\boldsymbol{W}_2\boldsymbol{x} + \boldsymbol{w}_1^T\boldsymbol{x} + w_0 \end{aligned}$$

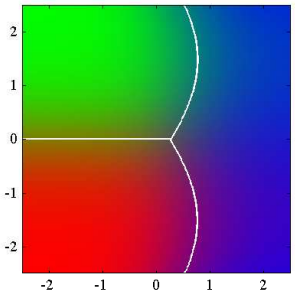
where we define

$$\begin{aligned} \boldsymbol{W}_2 &= \frac{1}{2}[\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}] \\ \boldsymbol{w}_1 &= \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + \log\frac{\pi_1}{\pi_0} + \frac{1}{2}\log\frac{|\boldsymbol{\Sigma}_0|}{|\boldsymbol{\Sigma}_1|} \end{aligned}$$

Naive Bayes: Decision Boundary

Left: Gaussian distributions with diagonal covariance, right: decision boundary.





Naive Bayes results in a quadratic decision boundary.

Compare to LDA which leads to a linear decision boundary.

Naive Bayes: Advantage

Class conditional for Naive Bayes: $p(x_1, x_2, \ldots, x_d|y = c) = \prod_{i=1}^{d} p(x_i|y = c)$.

Advantage: feature independence allows us to easily handle different data types.

Simply choose a suitable (univariate) distribution for each feature:
x_1 – Gaussian, x_2 – Categorical, ..., etc.

Outline

Probabilistic discriminative model

An alternative approach to generative modeling is to model the posterior distribution $p(y \mid \boldsymbol{x})$ directly. Such models are called discriminative.

We saw that a generative approach with Gaussian class-conditionals and shared covariance matrix $\boldsymbol{\Sigma}$ (LDA) leads to the posterior distribution

$$p(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w} + w_0),$$
$$p(y = 0 \mid \boldsymbol{x}) = 1 - \sigma(\boldsymbol{x}^T \boldsymbol{w} + w_0)$$

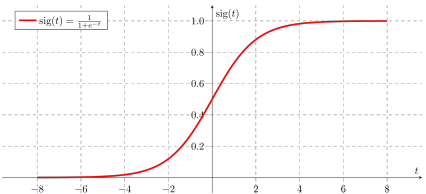
where $\boldsymbol{w}, w_0$ depend on the parameters of class-conditionals $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}$.

Why not just let $\boldsymbol{w}$ and w_0 be free parameters and choose them directly?

Logistic regression

We model the posterior distribution as $y \mid \boldsymbol{x} \sim \text{Bernoulli}(\sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0))$

where $\sigma(a) = \frac{1}{1+\exp(-a)}$ is the sigmoid and $\boldsymbol{w}, w_0$ are free model parameters.



This model is called logistic regression.

We again absorb the bias term by overloading the notation and defining $\boldsymbol{w}^T\boldsymbol{x} := w_0 + w_1x_1 + ... + w_Dx_D$ which is equivalent to defining $x_0 = 1$.

Likelihood of the logistic regression

Assuming that all samples $(\boldsymbol{x}_i, y_i)$ are drawn i.i.d., we can write the likelihood as

$$
\begin{aligned}
p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) &= \prod_{i=1}^{N} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \\
&= \prod_{i=1}^{N} \underbrace{p(y = 1 \mid \boldsymbol{x}_i, \boldsymbol{w})^{y_i}}_{=1 \text{ if } y_i = 0} \underbrace{\big(1 - p(y = 1 \mid \boldsymbol{x}_i, \boldsymbol{w})\big)^{1 - y_i}}_{=1 \text{ if } y_i = 1} \\
&= \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)^{y_i} (1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i))^{1 - y_i}
\end{aligned}
$$

Negative log-likelihood

Similar to linear regression, we can define the **negative log-likelihood** as the loss:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) &= -\log p\big(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X}\big) \\ &= -\sum_{i=1}^{N} \left(y_i \log \sigma(\boldsymbol{x}_i) + (1 - y_i) \log(1 - \sigma(\boldsymbol{x}_i))\right)\end{aligned}$$

This loss function is also called the binary cross entropy.

Finding the maximum likelihood estimate for $\boldsymbol{w}$ is equivalent to solving

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$$

Solving the minimization problem

There is **no closed-form** solution for logistic regression.

This means, we cannot represent the optimal $\boldsymbol{w}^*$ directly using standard mathematical operations, such as multiplication, matrix inversion, etc.

However, there is still hope! We can use standard optimization techniques to numerically solve our problem. We will cover this in the upcoming lectures.

For now, just assume that we can find $\boldsymbol{w}^*$.

Logistic regression + weights regularization

Since maximum likelihood estimation can lead to overfitting, just like in linear regression, we can control it by penalizing large weights.

$$\mathcal{L}_{\text{reg}}(\boldsymbol{w}) = -\log p\big(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X}\big) + \lambda ||\boldsymbol{w}||_q^q$$

As before, $q = 2$ corresponds to MAP estimation with a Gaussian prior on $\boldsymbol{w}$.

Again, there is no closed-form solution available.

Multiclass logistic regression

For the binary classification we used the sigmoid function to “squeeze” the unnormalized probability $\boldsymbol{w}^T\boldsymbol{x}$ into the range $(0, 1)$.

The same can be done for multiple classes using the softmax function.

$$p(y = c \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T\boldsymbol{x})}{\sum_{c'} \exp(\boldsymbol{w}_{c'}^T\boldsymbol{x})}$$

How does this relate to multiclass LDA?

Loss for multiclass logistic regression

The negative log-likelihood for multiclass logistic regression can be written as

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) &= -\log p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}) \\ &= -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} \log p(y_i = c \mid \boldsymbol{x}_i, \boldsymbol{w}) \\ &= -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} \log \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{c'} \exp(\boldsymbol{w}_{c'}^T \boldsymbol{x})}\end{aligned}$$

and is also called the cross entropy.

Here we use one-hot encoding: vector of categorical variables $\boldsymbol{y} \in \mathcal{C}^N$ is encoded as a binary matrix $\boldsymbol{Y} \in \{0,1\}^{N \times C}$, where $y_{ic} = \begin{cases} 1 & \text{if sample } i \text{ belongs to class } c \\ 0 & \text{else} \end{cases}$

Generative vs. discriminative models

Discriminative models usually tend to achieve better performance[2] when it comes to pure classification tasks.

While generative models work reasonably well when their assumptions hold, they can be fragile when these assumptions are violated.

Nevertheless, generative models provide the added benefits of better handling missing data, detecting outliers, generating new data and being more appropriate in the semi-supervised setting.

[2] If we measure performance just by the accuracy, however, this is not necessarily true for other metrics such as the calibration error or fairness.

Reading material

Main reading

- ”Pattern Recognition and Machine Learning” by Bishop
  [ch. 4.1.1, 4.1.2, 4.1.7, 4.2, 4.3.0–4.3.4]

Slides are based on an older version by G. Jensen, C. Osendorfer, and S. Günnemann. Some figures are from Bishop’s “Pattern Recognition and Machine Learning”.