



Machine Learning

Lecture 5: Linear Classification

Prof. Dr. Stephan Günnemann

Data Analytics and Machine Learning
Technical University of Munich

11.11.2019

Reading material

- Bishop: chapters 4, 4.1.1, 4.1.2, 4.1.7, 4.2, 4.3.0 - 4.3.4

Acknowledgements

- Slides are based on an older version by G. Jensen and J. Bayer
- Some figures are from C. Bishop: "Pattern Recognition and Machine Learning"

Notation

Symbol	Meaning
s	scalar is lowercase and not bold
\mathbf{s}	vector is lowercase and bold
\mathbf{S}	matrix is uppercase and bold
\hat{y}	predicted class label
y	actual class label
$\mathbb{I}(a)$	Indicator function; $\mathbb{I}(a) = 1$ if a is true, else 0

There is not a special symbol for vectors or matrices augmented by the bias term, w_0 . Assume it is always included as was done with linear regression.

Section 1

Introduction to linear classification

Regression

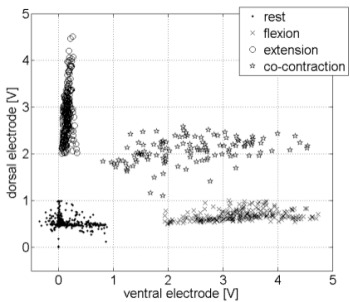
Output y is continuous (i.e. $y \in \mathbb{R}$).
For example, predict the price of a house given its area.

Classification

Output y belongs to one of C predetermined classes (i.e. $y \in \{1, \dots, C\}$).
For example, determine whether the picture shows a cat or a dog.

Given

- observations ¹
 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^D$
- set of possible classes.
 $\mathcal{C} = \{1, \dots, C\}$
- labels
 $\mathbf{y} = \{y_1, y_2, \dots, y_N\}, y_i \in \mathcal{C}$



Find

- function $f : \mathbb{R}^D \rightarrow \mathcal{C}$ that maps observations \mathbf{x}_i to class labels y_i

$$y_i = f(\mathbf{x}_i) \quad \text{for } i \in \{1, \dots, N\}$$

¹Like before, we represent samples as a **data matrix** $\mathbf{X} \in \mathbb{R}^{N \times D}$.

Zero-one loss

How do we measure quality of a prediction $\hat{\mathbf{y}} := f(\mathbf{X})$? ²

Zero-one loss denotes the number of misclassified samples.

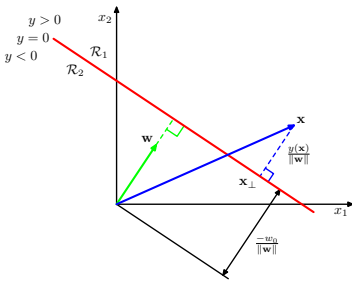
$$\ell_{01}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^N \mathbb{I}(\hat{y}_i \neq y_i).$$

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

²For brevity, we denote the generated prediction $f(\mathbf{x}_i)$ as \hat{y}_i , i.e. $\hat{\mathbf{y}}$ is the vector of predictions for entire \mathbf{X}

Hyperplane as a decision boundary

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



A hyperplane be defined by a normal vector \mathbf{w} and an offset w_0 .

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

Hyperplanes are computationally very convenient: easy to evaluate.

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

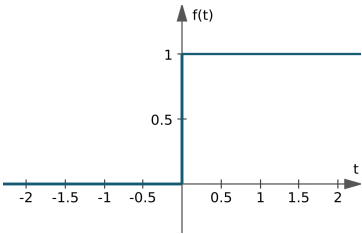
Perceptron algorithm is one of the oldest methods for binary classification.

Decision rule

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

where f is the step function defined as:

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



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

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

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

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

until all samples are classified correctly.

This method takes a finite number of steps to converge to a (\mathbf{w}, w_0) discriminating between two classes if it exists.³

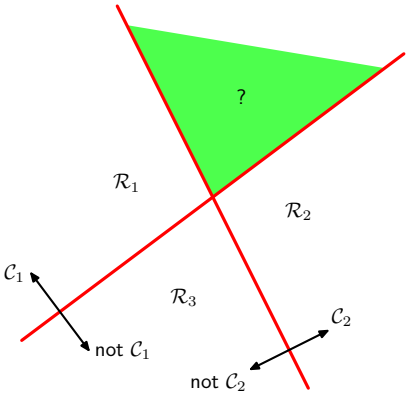
³However, there is no way to determine the number of required iterations in advance.

Does this scale up to multiple classes?

One-versus-rest classifier

Each hyperplane \mathcal{H}_i makes a decision

$$\text{class } \mathcal{C}_i \leftrightarrow \text{not class } \mathcal{C}_i$$

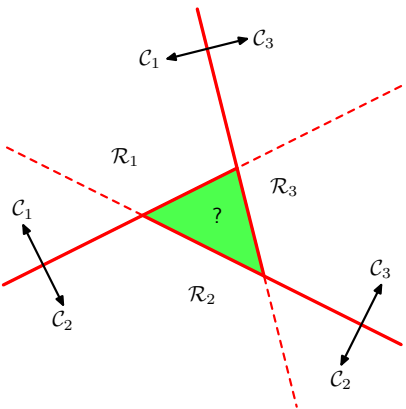


One-versus-one classifier

Hyperplane \mathcal{H}_{ij} makes a decision for each pair of classes

$$\text{class } \mathcal{C}_i \leftrightarrow \text{class } \mathcal{C}_j$$

Use majority vote to classify.



Multiclass discriminant

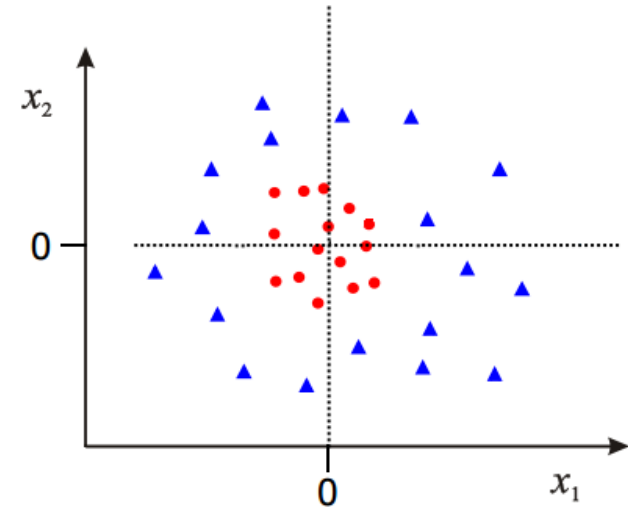
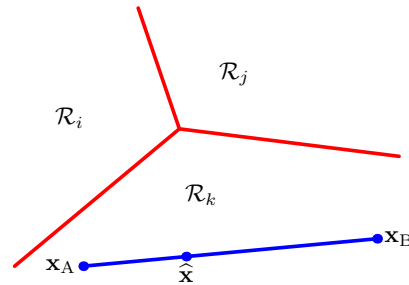
Define C linear functions of the form

$$f_c(\mathbf{x}) = \mathbf{w}_c^T \mathbf{x} + w_{0c}$$

with the decision rule

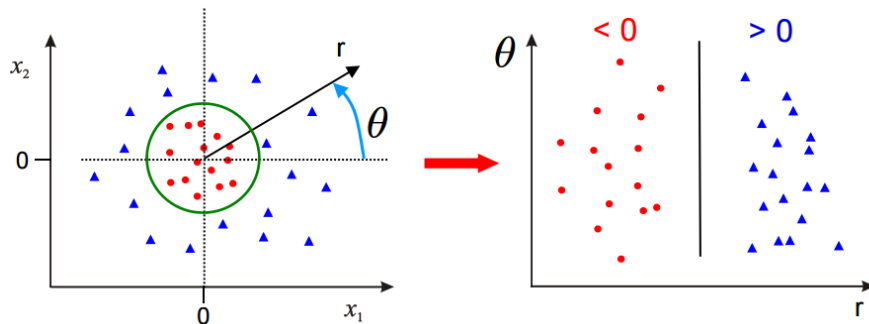
$$\hat{y} = \arg \max_{c \in \mathcal{C}} f_c(\mathbf{x})$$

That is, assign \mathbf{x} to the class c which produces the highest $f_c(\mathbf{x})$.



Basis functions

Like in the Linear Regression lecture last week, we can apply a nonlinear transformation $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$. We need to choose a ϕ that maps samples to a space where they are linearly separable.

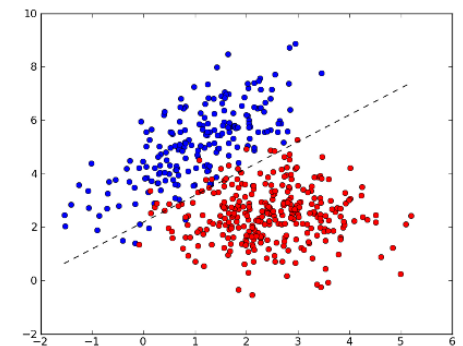


Here, $\phi(\mathbf{x}) = (\theta, r) = (\text{angle}(\mathbf{x}), \sqrt{\mathbf{x}^T \mathbf{x}})$.

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*.

Limitations of hard-decision based classifiers

- No measure of uncertainty
- Can't handle noisy data
- Poor generalization
- Difficult to optimize



What are the alternatives?

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

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

Two types of models:

Generative

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

Discriminative

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

Given $p(y \mid \mathbf{x})$ we can make the prediction \hat{y} based on our problem.

Popular choice is the mode: $\hat{y} = \arg \max_{c \in \mathcal{C}} p(y = c \mid \mathbf{x})$.

Section 2

Probabilistic generative models for linear classification

Generative model

The idea is to obtain the class posterior using the Bayes formula

$$p(y = c \mid \mathbf{x}) \propto \underbrace{p(\mathbf{x} \mid y = c)}_{\text{class conditional}} \cdot \underbrace{p(y = c)}_{\text{class prior}} \quad (1)$$

The model consists of

- **class prior** - a priori probability of a point belonging to a class c
- **class conditional** - probability of generating a point \mathbf{x} , given that it belongs to class c

Applying a generative model

Applying a generative model typically works as following

- Choose a parametric model for the class conditional $p(\mathbf{x} \mid y = c, \boldsymbol{\psi})$ and the class prior $p(y = c \mid \boldsymbol{\theta})$.
- Estimate the parameters of our model $\{\boldsymbol{\psi}, \boldsymbol{\theta}\}$ from the data \mathcal{D} (e.g., using maximum likelihood - obtain estimates $\{\hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\theta}}\}$). This step is called **learning**.

Once fitted, we can perform **inference** - classify a new \mathbf{x} using Bayes rule

$$p(y = c \mid \mathbf{x}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\theta}}) \propto p(\mathbf{x} \mid y = c, \hat{\boldsymbol{\psi}}) p(y = c \mid \hat{\boldsymbol{\theta}}) \quad (2)$$

Additionally, we can **generate** new data - hence the name.

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

Choosing class prior

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 $\sum_{c=1}^C \theta_c = 1$.

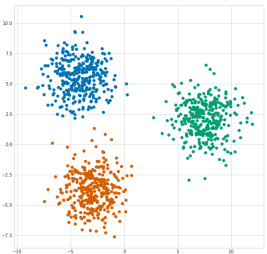
The maximum likelihood estimate for $\boldsymbol{\theta}$ given the data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ is

$$\theta_c^{MLE} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(y_i = c)$$

Class conditionals

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

The feature vector $\mathbf{x} \in \mathbb{R}^D$ is continuous.
 \implies Use a multivariate normal for each class!



$$p(\mathbf{x} \mid y = c) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}) \tag{3}$$

$$= \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_c) \right\} \tag{4}$$

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.

The MLE estimates for $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_C, \boldsymbol{\Sigma}\}$ will be derived in the tutorial (or see Bishop 4.2.2).

Posterior distribution

Now that we have chosen $p(\mathbf{x} \mid y)$ and $p(y)$, and have estimated their parameters from the training data ⁴, how do we perform classification?

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

$$p(y = 1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y = 1) p(y = 1)}{p(\mathbf{x} \mid y = 1) p(y = 1) + p(\mathbf{x} \mid y = 0) p(y = 0)} \tag{5}$$

$$= \frac{1}{1 + \exp(-a)} =: \sigma(a) \tag{6}$$

where we defined

$$a = \ln \frac{p(\mathbf{x} \mid y = 1) p(y = 1)}{p(\mathbf{x} \mid y = 0) p(y = 0)} \tag{7}$$

and σ is the [sigmoid function](#).

⁴To avoid clutter, we implicitly condition the distributions on their respective parameters $(\boldsymbol{\theta}, \boldsymbol{\mu}_c, \boldsymbol{\Sigma})$

Linear discriminant analysis (LDA)

Let's look at how this function looks for Gaussian class-conditional with the same covariance $\boldsymbol{\Sigma}$

$$a = \ln \frac{p(\mathbf{x} \mid y = 1) p(y = 1)}{p(\mathbf{x} \mid y = 0) p(y = 0)} \tag{8}$$

$$= -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \ln p(y = 1) \tag{9}$$

$$+ \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_0) - \ln p(y = 0) \tag{10}$$

$$= \mathbf{w}^T \mathbf{x} + w_0 \tag{11}$$

where we define

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \tag{12}$$

$$w_0 = -\frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 + \ln \frac{p(y = 1)}{p(y = 0)} \tag{13}$$

LDA for $C = 2$ classes

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

$$p(y = 1 | \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + w_0))} \tag{14}$$

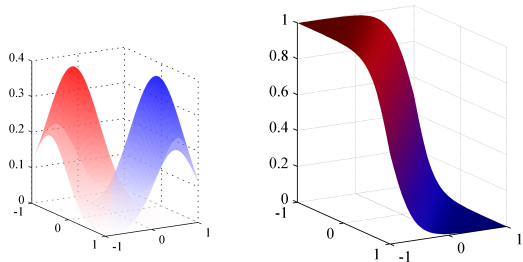
$$= \sigma(\mathbf{w}^T \mathbf{x} + w_0) \tag{15}$$

or equivalently

$$y | \mathbf{x} \sim \text{Bernoulli}(\sigma(\mathbf{w}^T \mathbf{x} + w_0)) \tag{16}$$

This is how this function looks for $D = 2$

- Right:
 $p(\mathbf{x} | y = 1)$ - red
 $p(\mathbf{x} | y = 0)$ - blue
- Left:
 $p(y = 1 | \mathbf{x})$



LDA for $C > 2$ classes

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

$$p(y = c | \mathbf{x}) = \frac{p(\mathbf{x} | y = c)p(y = c)}{\sum_{c'=1}^C p(\mathbf{x} | y = c')p(y = c')} \tag{17}$$

working out through some math we get

$$= \frac{\exp(\mathbf{w}_c^T \mathbf{x} + w_{c0})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x} + w_{c'0})} \tag{18}$$

where

$$\mathbf{w}_c = \Sigma^{-1} \boldsymbol{\mu}_c \tag{19}$$

$$w_{c0} = -\frac{1}{2} \boldsymbol{\mu}_c^T \Sigma^{-1} \boldsymbol{\mu}_c + \ln p(y = c) \tag{20}$$

Softmax function

In the slide above we made use of the [softmax function](#).

Softmax σ is a generalization of sigmoid to multiple dimensions

$$\sigma : \mathbb{R}^K \rightarrow \Delta^{K-1} \tag{21}$$

where

$$\Delta^{K-1} = \left\{ \mathbf{x} \in \mathbb{R}^K \mid \sum_{k=1}^K x_k = 1 \text{ and } x_k \geq 0, k = 1, \dots, K \right\} \tag{22}$$

is the standard [probability simplex](#).

Softmax is defined as

$$\sigma(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{k=1}^K \exp(x_k)} \tag{23}$$

Class conditionals: Variant II (Naive Bayes)

[Naive Bayes](#): Assume that the d features of a sample $\mathbf{x} = (x_1, x_2, \dots, x_d)$ are conditionally independent given the class, i.e.

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

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

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

Important: We use a different covariance matrix Σ_c for each class c !

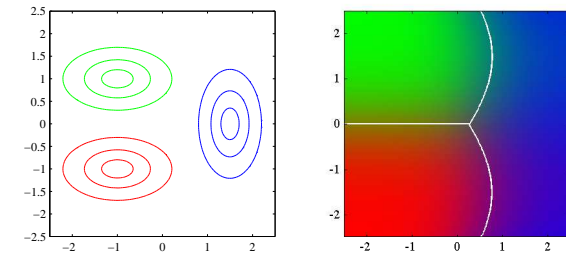
[LDA](#): All class conditionals share the same covariance matrix Σ .

Let's assume for simplicity that we have two classes $\mathcal{C} = \{0, 1\}$. Like in LDA we need to compute

$$\begin{aligned} a &= \ln \frac{p(\mathbf{x} | y = 1)p(y = 1)}{p(\mathbf{x} | y = 0)p(y = 0)} \\ &= \frac{1}{2} \mathbf{x}^T [\Sigma_0^{-1} - \Sigma_1^{-1}] \mathbf{x} + \mathbf{x}^T [\Sigma_1^{-1} \boldsymbol{\mu}_1 - \Sigma_0^{-1} \boldsymbol{\mu}_0] \\ &\quad - \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma_1^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_0^T \Sigma_0^{-1} \boldsymbol{\mu}_0 + \log \frac{\pi_1}{\pi_0} + \frac{1}{2} \ln \frac{|\Sigma_0|}{|\Sigma_1|} \\ &= \mathbf{x}^T \mathbf{W}_2 \mathbf{x} + \mathbf{w}_1^T \mathbf{x} + w_0 \end{aligned}$$

where we define

$$\begin{aligned} \mathbf{W}_2 &= \frac{1}{2} [\Sigma_0^{-1} - \Sigma_1^{-1}] \\ \mathbf{w}_1 &= \Sigma_1^{-1} \boldsymbol{\mu}_1 - \Sigma_0^{-1} \boldsymbol{\mu}_0 \\ w_0 &= -\frac{1}{2} \boldsymbol{\mu}_1^T \Sigma_1^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_0^T \Sigma_0^{-1} \boldsymbol{\mu}_0 + \log \frac{\pi_1}{\pi_0} + \frac{1}{2} \ln \frac{|\Sigma_0|}{|\Sigma_1|} \end{aligned}$$



Naive Bayes results in a quadratic decision boundary.

Recap: LDA leads to a linear decision boundary

Class conditional for Naive Bayes:

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

Advantage: Independence of the features allows to easily handle different data types/mixed data types.

Simply choose a suitable (univariate) distribution for each feature:

x_1 – Gaussian, x_2 – Categorical, ...

Section 3

Probabilistic discriminative models for linear classification

Probabilistic discriminative model

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

We saw in the previous section that a generative approach with Gaussian class-conditional distributions with same covariance matrix Σ (i.e. LDA) produces the following decision rule

$$p(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{x}^T \mathbf{w} + w_0), \tag{24}$$

$$p(y = 0 \mid \mathbf{x}) = 1 - \sigma(\mathbf{x}^T \mathbf{w} + w_0) \tag{25}$$

where \mathbf{w}, w_0 depend on the parameters of class-conditionals μ_0, μ_1, Σ .

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

Logistic regression

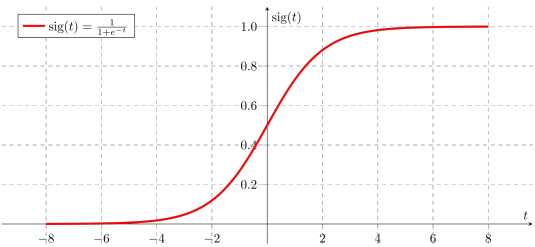
We model the posterior distribution as

$$y \mid \mathbf{x} \sim \text{Bernoulli}(\sigma(\mathbf{w}^T \mathbf{x} + w_0)) \tag{26}$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{27}$$

and \mathbf{w}, w_0 are the free model parameters.



This model is called **logistic regression**.

Absorbing the bias term

Like in the previous lecture, we again absorb the bias term by overloading the notation and defining

$$\mathbf{w}^T \mathbf{x} := w_0 + w_1 x_1 + \dots + w_D x_D \tag{28}$$

Which is equivalent to defining $x_0 = 1$.

Likelihood of logistic regression

Learning logistic regression comes down to finding a “good” setting of parameters \mathbf{w} that “explain” the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

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

$$p(\mathbf{y} \mid \mathbf{w}, \mathbf{X}) = \prod_{i=1}^N p(y_i \mid \mathbf{x}_i, \mathbf{w}) \tag{29}$$

$$= \prod_{i=1}^N \underbrace{p(y = 1 \mid \mathbf{x}_i, \mathbf{w})^{y_i}}_{=1 \text{ if } y_i=0} \underbrace{(1 - p(y = 1 \mid \mathbf{x}_i, \mathbf{w}))^{1-y_i}}_{=1 \text{ if } y_i=1} \tag{30}$$

$$= \prod_{i=1}^N \sigma(\mathbf{w}^T \mathbf{x}_i)^{y_i} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))^{1-y_i} \tag{31}$$

Similarly to the linear regression case, we can define an error function as [negative log-likelihood](#)

$$E(\mathbf{w}) = -\ln p(\mathbf{y} \mid \mathbf{w}, \mathbf{X}) \quad (32)$$

$$= -\sum_{i=1}^N (y_i \ln \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \ln(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) \quad (33)$$

This loss function is called [binary cross entropy](#).

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

$$\mathbf{w}^* = \arg \min E(\mathbf{w}) \quad (34)$$

There doesn't exist a [closed form](#) solution for logistic regression. This means, we cannot compute the optimal \mathbf{w}^* using standard mathematical operations, such as multiplication, matrix inversion, etc.

However, there is still hope! We can use [optimization](#) to numerically solve our problem. We will cover this in the next lecture.

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

As we already well know, maximum likelihood estimation may often lead to overfitting. Just like in case of linear regression, we can control this by penalizing large weights.

$$E(\mathbf{w}) = -\ln p(\mathbf{y} \mid \mathbf{w}, \mathbf{X}) + \lambda \|\mathbf{w}\|_q^2 \quad (35)$$

Like before, for $q = 2$ this corresponds to MAP estimation with a Gaussian prior on \mathbf{w} .

Again, there is no closed form solution available.

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

The same can be done for multiple classes using the [softmax](#) function.

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

How does this relate to multiclass LDA?

The negative log-likelihood for multiclass LR can be written as

$$E(\mathbf{w}) = -\ln p(\mathbf{Y} \mid \mathbf{w}, \mathbf{X}) \quad (36)$$

$$= -\sum_{i=1}^N \sum_{c=1}^C y_{ic} \ln p(y_i = c \mid \mathbf{x}_i, \mathbf{w}) \quad (37)$$

$$= -\sum_{i=1}^N \sum_{c=1}^C y_{ic} \ln \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'} \exp(\mathbf{w}_{c'}^T \mathbf{x})} \quad (38)$$

and is called **cross entropy**.

Here we use **one-hot encoding**: vector of categorical variables $\mathbf{y} \in \mathcal{C}^N$ is encoded as a binary matrix $\mathbf{Y} \in \{0, 1\}^{N \times C}$, where

$$y_{ic} = \begin{cases} 1 & \text{if sample } i \text{ belongs to class } c \\ 0 & \text{else} \end{cases} \quad (39)$$

- In general, discriminative models achieve better performance when it comes to pure classification tasks.
- While generative models work reasonably well when their assumptions hold, they are quite fragile when these assumptions are violated.
- Generative modeling for high-dimensional / strongly correlated data like images or graphs is still an open research challenge.
- 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.