Lecture 3 – Classification, logistic regression

Jens Sjölund
https://jsjol.github.io/
Department of Information Technology
Uppsala University

Course webpage

Summary of lecture 2 (I/III)

Regression is about learning a model that describes the relationship between an input variable $\mathbf{x}$ and a numerical output variable y

$$y = f(\mathbf{x}; \boldsymbol{\theta}) + \epsilon.$$

Linear regression is regression with a linear model

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p + \epsilon.$$

How to learn/train/estimate θ?

Use the **maximum likelihood** principle: assume $\varepsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ iid
$\Rightarrow$ **least squares** & **normal equations**

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \, \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2, \qquad \widehat{\boldsymbol{\theta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y},$$

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} 1 & -\mathbf{x}_1^\mathsf{T}- \\ \vdots & \vdots \\ 1 & -\mathbf{x}_n^\mathsf{T}- \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Summary of lecture 2 (II/III)

We can make **arbitrary nonlinear transformations** of the inputs, for example polynomials

$$y = \theta_0 + \theta_1 \underset{\substack{\shortparallel \\ x_1}}{v} + \theta_2 \underset{\substack{\shortparallel \\ x_2}}{v^2} + \theta_3 \underset{\substack{\shortparallel \\ x_3}}{v^3} + \cdots + \theta_p \underset{\substack{\shortparallel \\ x_p}}{v^p} + \epsilon$$

(v = original input variable, x_k transformed input variables or features)

Categorical input variables are handled by creating dummy variables.

Summary of lecture 2 (III/III)

Overfitting may occur when the model is **too flexible!**

Can be handled using **regularization**, which amounts to adding a term $R(\boldsymbol{\theta})$ to the cost function which controls the model flexibility,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \underbrace{V(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}_{\text{data fit}} + \lambda \underbrace{R(\boldsymbol{\theta})}_{\text{penalty}}$$

Ridge regression

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \; \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2 + \gamma \|\boldsymbol{\theta}\|_2^2$$

LASSO

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \; \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2 + \gamma \|\boldsymbol{\theta}\|_1$$

Outline – Lecture 3

Aim: To introduce the classification problem and derive a first useful classifier: logistic regression.

Outline:

1. Summary of Lecture 2
1. The classification problem
2. Logistic regression
3. Diagnostic tools for classification (via example)

UPPSALA UNIVERSITET

Categorical outputs

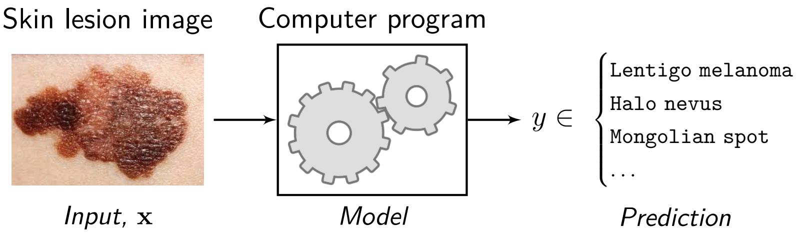
Many machine learning applications have **categorical outputs** y

- Face verification:
  $y \in \{\texttt{match}, \texttt{no match}\}$
- Identify the spoken vowel from an audio signal: $y \in \{\texttt{A}, \texttt{E}, \texttt{I}, \texttt{O}, \texttt{U}, \texttt{Y}\}$
- Diagnosis system for leukemia:
  $y \in \{\texttt{ALL}, \texttt{AML}, \texttt{CLL}, \texttt{CML}, \texttt{no leukemia}\}$
- Anomaly detection, e.g. in cybersecurity or predictive maintenance:
  $y \in \{\texttt{normal}, \texttt{anomalous}\}$

The classification problem

Classification: learn a **model** which, for each input data point $\mathbf{x}$ can predict its **class** $y \in \{1, \dots, M\}$.

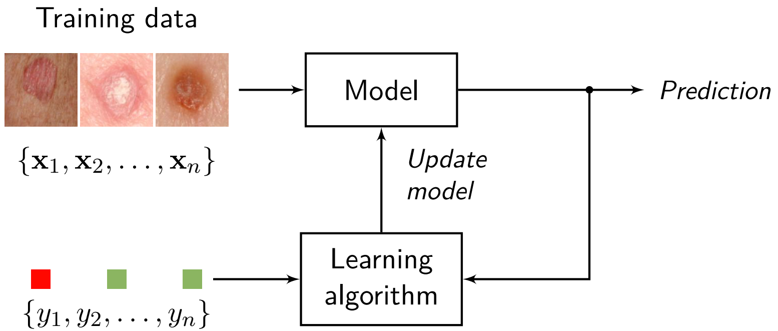
ex) Classifying skin lesions



Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau and Sebastian Thrun. **Dermatologist-level classification of skin cancer with deep neural networks**. *Nature*, 542:115–118, 2017.

Training a classifier

Supervised learning: The model is **learned** by adapting it to labeled **training data** $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$.



Training data
$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
Model
Prediction
Update model
Learning algorithm
$\{y_1, y_2, \dots, y_n\}$

Classification

A **classification model** can be specified in terms of the **conditional class probabilities**

$$p(y = m \,|\, \mathbf{x}) \quad \text{for} \quad m = 1, \dots, M,$$

encoding the probability for class m given (i.e. conditioned on) that we know the input $\mathbf{x}$.

More specifically (for a binary classification task, $M = 2$, $y = 1$ or $y = -1$) we learn a model $g(\mathbf{x})$ that describes the conditional class probabilities $p(y \,|\, \mathbf{x})$ ($=$ a **classifier**),

$$g(\mathbf{x}) = p(y = 1 \,|\, \mathbf{x}).$$

UPPSALA UNIVERSITET

Why not linear regression?

Can we use linear regression for classification problems?

ex) Classifying e-mails as spam. Code the output as

$$y = \begin{cases} -1 & \text{if } \texttt{ham} \text{ (= good email)}, \\ 1 & \text{if } \texttt{spam}, \end{cases}$$

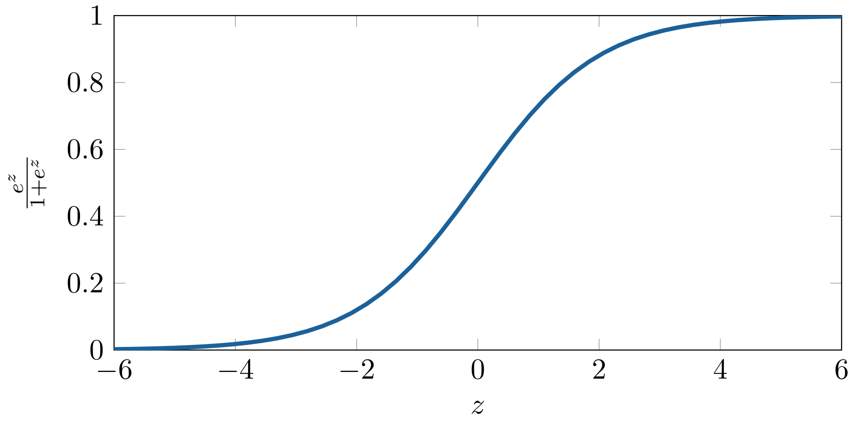
and learn a linear regression model. Classify as spam if $\hat{y} > 0$.

Why is this not a good idea?

- Sensitive to unequally sized classes in the training data.
- Difficult to generalize to $K > 2$ classes.

Logistic function (aka sigmoid function)

The function $f : \mathbb{R} \mapsto [0, 1]$ defined as $f(z) = \frac{e^z}{1+e^z}$ is known as the **logistic function**.



UPPSALA
UNIVERSITET

Classification – the multi-class case

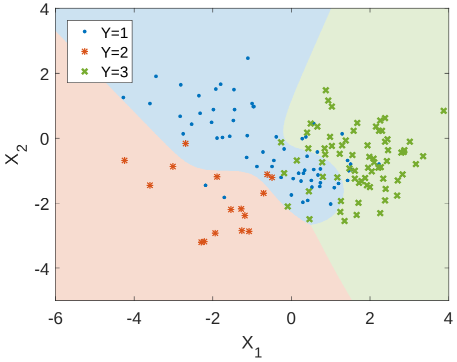
We return a vector valued classifier $\boldsymbol{g}(\mathbf{x})$, where

$$\begin{bmatrix} p(y=1\,|\,\mathbf{x}) \\ p(y=2\,|\,\mathbf{x}) \\ \vdots \\ p(y=M\,|\,\mathbf{x}) \end{bmatrix} \text{ is modeled by } \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_M(\mathbf{x}) \end{bmatrix} = \boldsymbol{g}(\mathbf{x}).$$

Each element $g_m(\mathbf{x})$ corresponds to the conditional class probability $p(y=m\,|\,\mathbf{x})$.

Decision boundaries

The input space can be segmented into M regions, separated by so-called **decision boundaries**.



Finding the decision boundary

The **decision boundary** is found by solving the equation $g(x) = 1 - g(x)$.

For logistic regression this corresponds to

$$\frac{e^{\boldsymbol{\theta}^\mathsf{T}\mathbf{x}}}{1 + e^{\boldsymbol{\theta}^\mathsf{T}\mathbf{x}}} = \frac{1}{1 + e^{\boldsymbol{\theta}^\mathsf{T}\mathbf{x}}}$$

which we can write as $e^{\boldsymbol{\theta}^\mathsf{T}\mathbf{x}} = 1$. Hence, we have the following expression for the decision boundary

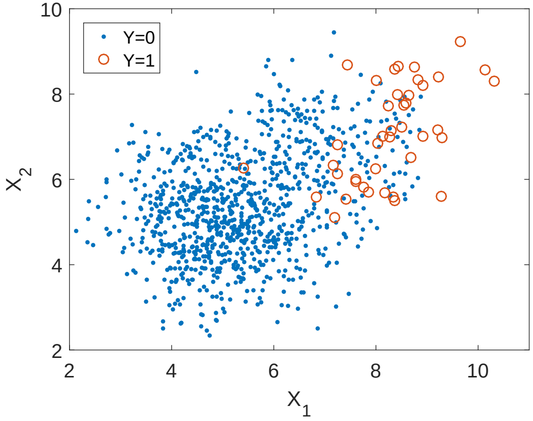
$$\boldsymbol{\theta}^\mathsf{T}\mathbf{x} = 0.$$

Linear expression for the decision boundary!

ex) Logistic regression

Consider a *toyish* problem where we want to build a classifier for whether a person has a certain disease $(y = 1)$ or not $(y = -1)$ based on two biological indicators x_1 and x_2.

The training data consists of $n = 1,000$ labeled samples (right).



ex) Logistic regression

We then fit a logistic regression model

$$p(y = 1 \,|\, \mathbf{x}) = \frac{e^{\theta_0 + \theta_1 x_1 + \theta_2 x_2}}{1 + e^{\theta_0 + \theta_1 x_1 + \theta_2 x_2}}$$

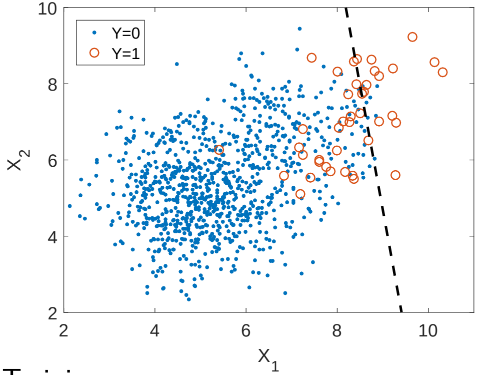
using maximum likelihood.

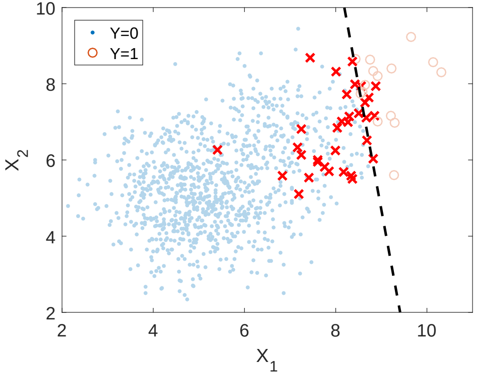
To evaluate the performance of a classifier, our default choice is the **misclassification rate**:

$$\frac{1}{n} \sum_{i=1}^{n} \underbrace{\mathbb{I}(\hat{y}_i \neq y_i)}_{\begin{cases} 1 & \text{if } \hat{y}_i \neq y_i \\ 0 & \text{if } \hat{y}_i = y_i \end{cases}}$$

UPPSALA UNIVERSITET

ex) Logistic regression: training error



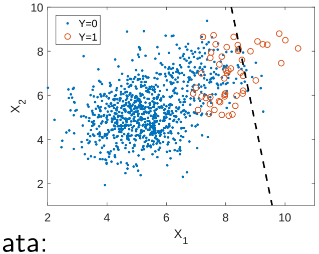


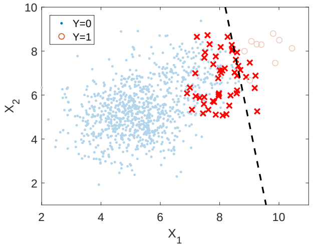
Training error:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(\hat{y}_i \neq y_i) = 3.3\%.$$

ex) Logistic regression: test error

To further test the classifier we evaluate it on ***previously unseen*** **test data**: $\{(\mathbf{x}'_i, y'_i)\}_{i=1}^{n_t}$.





(Estimated) error on the test data:

$$\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbb{I}(\hat{y}'_i \neq y'_i) = 5.0\%$$

Error on the test data for the naive classifier $\hat{y} \equiv 0$: 5.1%.

ex) Logistic regression: confusion matrix

Instead of just looking at the misclassification rate it is better to compute the so-called **confusion matrix**.

True negative **False positive**

		Predicted condition	
		$\hat{y} = -1$	$\hat{y} = 1$
True condition	$y = -1$	941	8
	$y = 1$	42	9

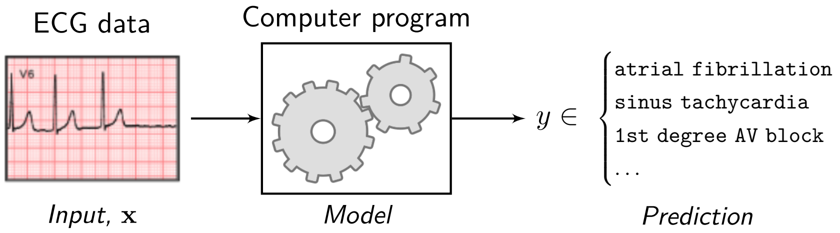
False negative **True positive**

Out of 51 patients affected by the disease, only 9 are correctly classified!

The **True Positive Rate (TPR)** is just $9/51 \approx 17.7\%$.

A classification problem from our research

Aim: Automatic classification of Electrocardiography (ECG) data.

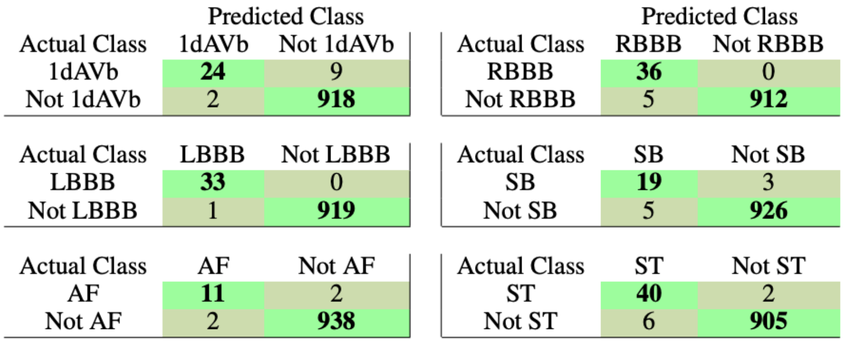


We are now at human level (medical doctors) performance.

Antonio H. Ribeiro, Manoel Horta, Gabriela Paixao, Derick Oliveira, Paulo R. Gomes, Jessica A. Canazart, Milton Pifano, Wagner Meira Jr., Thomas B. Schön and Antonio Luiz Ribeiro. **Automatic diagnosis of short-duration 12-lead ECG using a deep convolutional network**. In *Nature Communications*, 2020. To appear.

UPPSALA UNIVERSITET

Confusion matrices for ECG classification



	Predicted Class	
Actual Class	1dAVb	Not 1dAVb
1dAVb	**24**	9
Not 1dAVb	2	**918**

	Predicted Class	
Actual Class	RBBB	Not RBBB
RBBB	**36**	0
Not RBBB	5	**912**

Actual Class	LBBB	Not LBBB
LBBB	**33**	0
Not LBBB	1	**919**

Actual Class	SB	Not SB
SB	**19**	3
Not SB	5	**926**

Actual Class	AF	Not AF
AF	**11**	2
Not AF	2	**938**

Actual Class	ST	Not ST
ST	**40**	2
Not ST	6	**905**

Conservative predictions

Common to start thinking about a classifier minimizing the total misclassification error.

What if false negatives are "worse" than false positives?

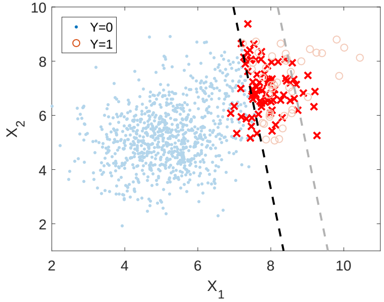
Idea: Modify the prediction model

$$\hat{y}(\mathbf{x}_\star) = \begin{cases} 1 & \text{if } g(\mathbf{x}_\star) > r, \\ -1 & \text{otherwise,} \end{cases}$$

where $0 \leq r \leq 1$ is a user chosen threshold.

UPPSALA UNIVERSITET

ex) Logistic regression, cont'd



	$\hat{y} = -1$	$\hat{y} = 1$
$y = -1$	881	68
$y = 1$	10	41

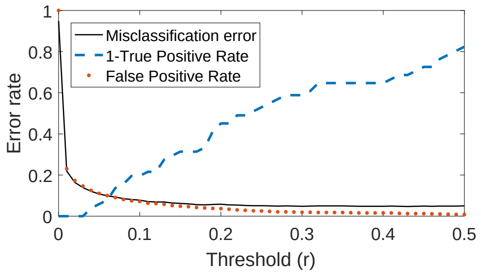
Table: Confusion matrix ($r = 0.2$)

If we set the threshold at $r = 0.2$,

- ▲ The ***true positive rate*** is increased to $41/51 = 80.4\%$.
- ▼ However, the ***misclassification error*** is increased to 7.8%.

UPPSALA UNIVERSITET

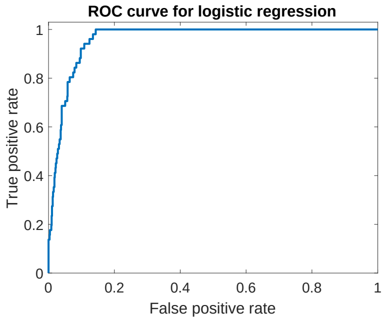
ex) Logistic regression: error rates



As we increase the threshold r from 0 to 0.5:

- ▲ The ***misclassification error*** decreases
- ▲ The number of ***non-diseased persons incorrectly classified as diseased*** **(False Positive Rate)** decreases.
- ▼ The number of ***diseased persons incorrectly classified as non-diseased*** **(1 − True Positive Rate)** increases!

ex) Logistic regression: ROC and AUC



- **ROC[1] curve**: plot of TPR vs. FPR.
- **Area Under Curve (AUC):** condensed performance measure for the classifier, taking all possible thresholds into account.
- AUC $\in [0,1]$ where AUC $= 0.5$ corresponds to a random guess. [***ex)*** AUC $= 0.96$]

[1]For *Receiver Operating Characteristics*, which is a historical name.

A few concepts to summarize lecture 3

Classification: The problem of modeling a relationship between an (arbitrary) input $\mathbf{x}$ and a categorical output y.

Decision boundaries: Points in the input space where the classifier $\hat{y}(\mathbf{x})$ changes value.

Logistic regression: A model that uses a logistic function to model a binary output variable.

Confusion matrix: Table with predicted vs true class labels (for binary classification $\Rightarrow$ number of true negatives, true positives, false negatives, and false positives).

UPPSALA UNIVERSITET

More on evaluating performance

Training/learning: Minimizing the average loss on training data. Loss functions differ between methods and the values can not directly be compared.

We would like to evaluate already trained models with respect to hyperparameters (e.g. regularization parameter γ) but also different methods (logistic regression, neural net, etc.).

Error function

$$E(\hat{y}(\mathbf{x})), y) = \begin{cases} \mathbb{I}\left(\widehat{y}(\mathbf{x}) = y\right) & \text{(classification)} \\ \left(\widehat{y}(\mathbf{x}) - y\right)^2 & \text{(regression)} \end{cases}$$

Note: Error function not necessarily the same as loss function.

More on evaluating performance

Training error

$$E_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n} E(\widehat{y}(\mathbf{x}_i), y_i)$$

Measures how well a predictor $\widehat{y}$ performs on training data, but we are interested in new data. Let $p(\mathbf{x}, y)$ be the joint distribution over data.

Expected new data error

$$E_{\text{new}} = \mathbb{E}_\star\left[E(\widehat{y}(x_\star), y_\star)\right] = \int E(\widehat{y}(x_\star), y_\star)p(x_\star, y_\star)dx_\star dy_\star$$

Impossible to compute since $p(\mathbf{x}, y)$ is unknown, but minimizing E_{new} is our ultimate goal.

Maybe we can learn it?

Approximating integrals

By the law of large numbers, we can approximate integrals using samples:

$$\mathbb{E}\left[h(\mathbf{x})\right] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i), \quad \mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}_i),\ i = 1, \ldots, n$$

With samples from $p(\mathbf{x}, y)$, we can estimate E_{new}!

Note: Important that samples come from real world, "in-production" distribution.

Training data are (or should be) n samples from $p(\mathbf{x}, y)$.

Approximate $E_{\text{new}} \overset{?}{\approx} E_{\text{train}}$? **NO!**

Training data is part of the predictor $\widehat{y}(\mathbf{x}; \mathcal{T})$!