Classification

Classification vs. regression

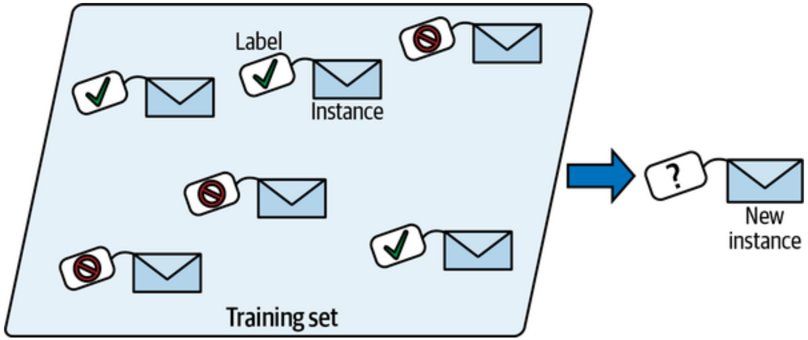
In supervised learning we try to find a function f, which systematically produces the output values y_m associated with the input values $\boldsymbol{X}_{m,:}$:

$$f(\boldsymbol{X}_{m,:}) \rightarrow y_m$$

Classification

Target variable y: categorical

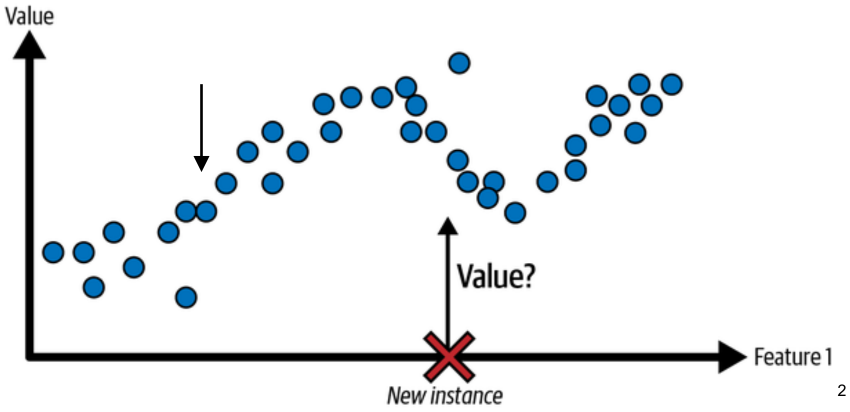
$$y_m \in \{C_1, C_2, \dots, C_K\}$$



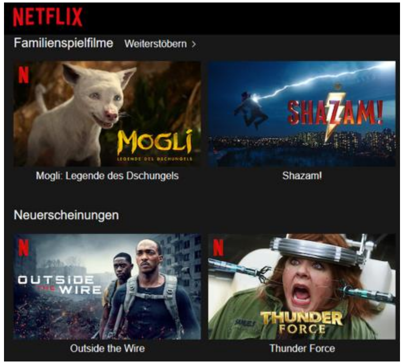
Regression

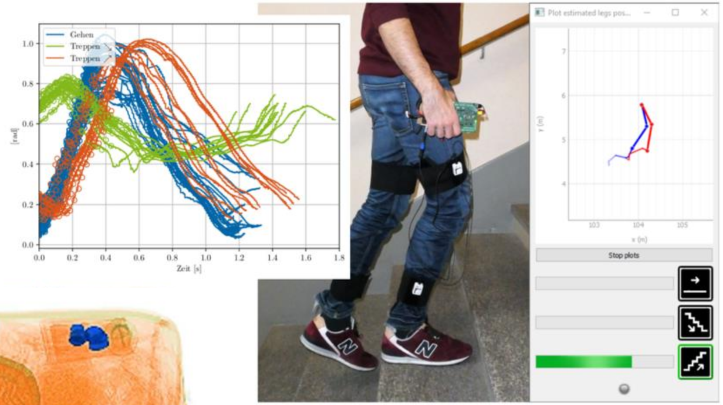
Target variable y: numerical - continuous

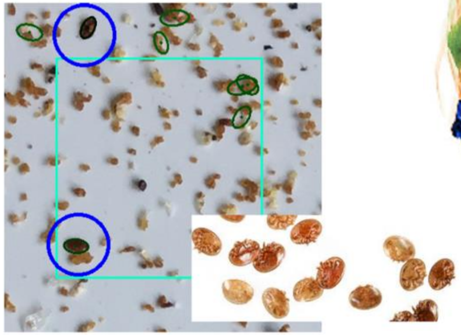
$$y_m \in \mathbb{R}$$



Classification examples



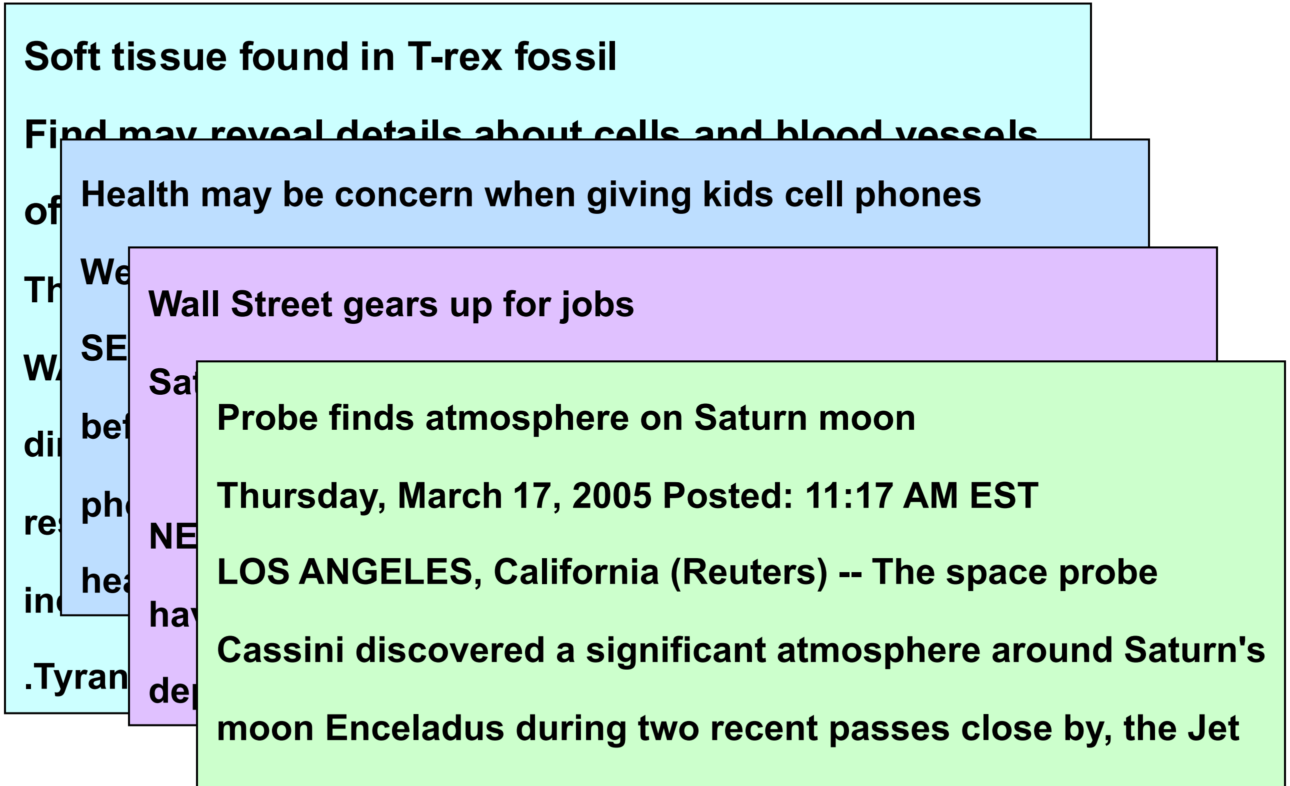




Text classification

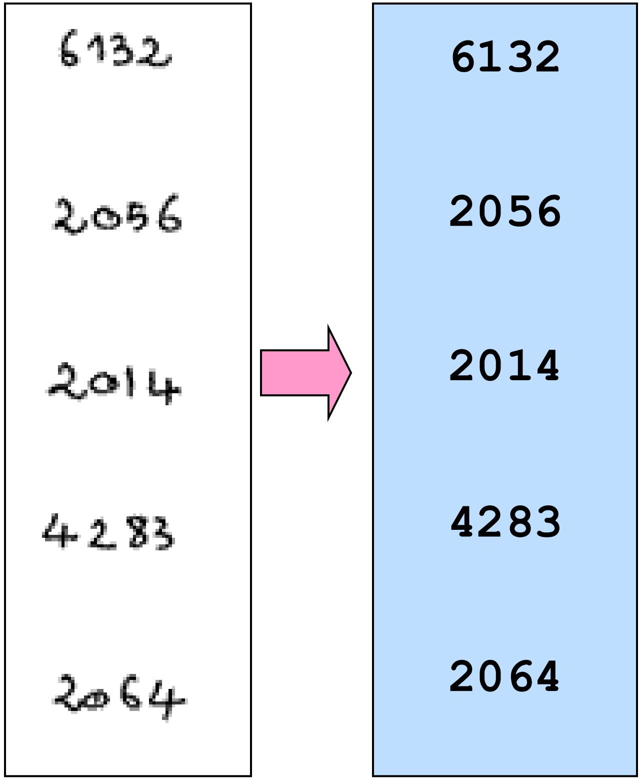
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

Categorize text documents into predefined categories . For example, categorize news into ‘sports’, ‘politics’, ‘science’, etc.



Character recognition

Identify handwritten characters: classify each image of a character into one of 10 categories ‘0’, ‘1’, ‘2’ …



Types of classicifation problems

Binary: 2 possible classes

- **Structured data**

Credit amount	cash	age	purpose
60’000	25’000	55	«Auto»
…			

Risk
0
1

- **Images**

Hot Dog

No Hot Dog

- **Text**
  - Sentiment analysis:

positive	negative
«Great pasta and perfect location!»	«Bad service – never again!»

  - Spam detection: spam/ham

Multinomial (multi-class): >2 possible classes

Iris Versicolor
Iris Setosa
Iris Virginica

French
English
German

Multilabel:

Love story
Action
Comedy
Science fiction
Thriller

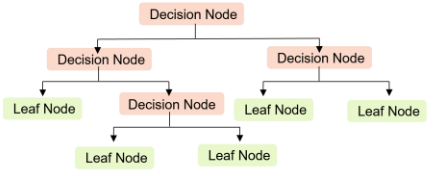
Classification algorithms

Generative classifiers

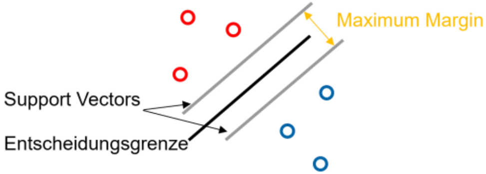
e.g. Naive Bayes

$$p(C_c|x) \propto p(C_c) \prod_{n=1}^{N} p(x_n|C_c)$$

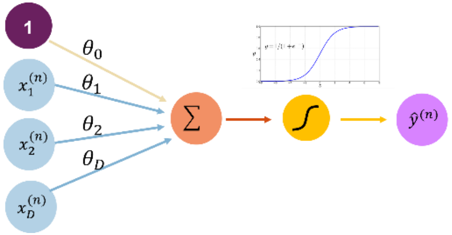
Decision Trees



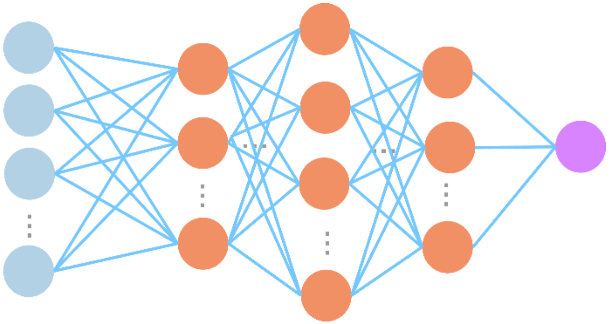
Support Vector Machines (SVM)



Logistic regression



Neural networks



Logistic Regression

Logistic regression solves a binary classification task

Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

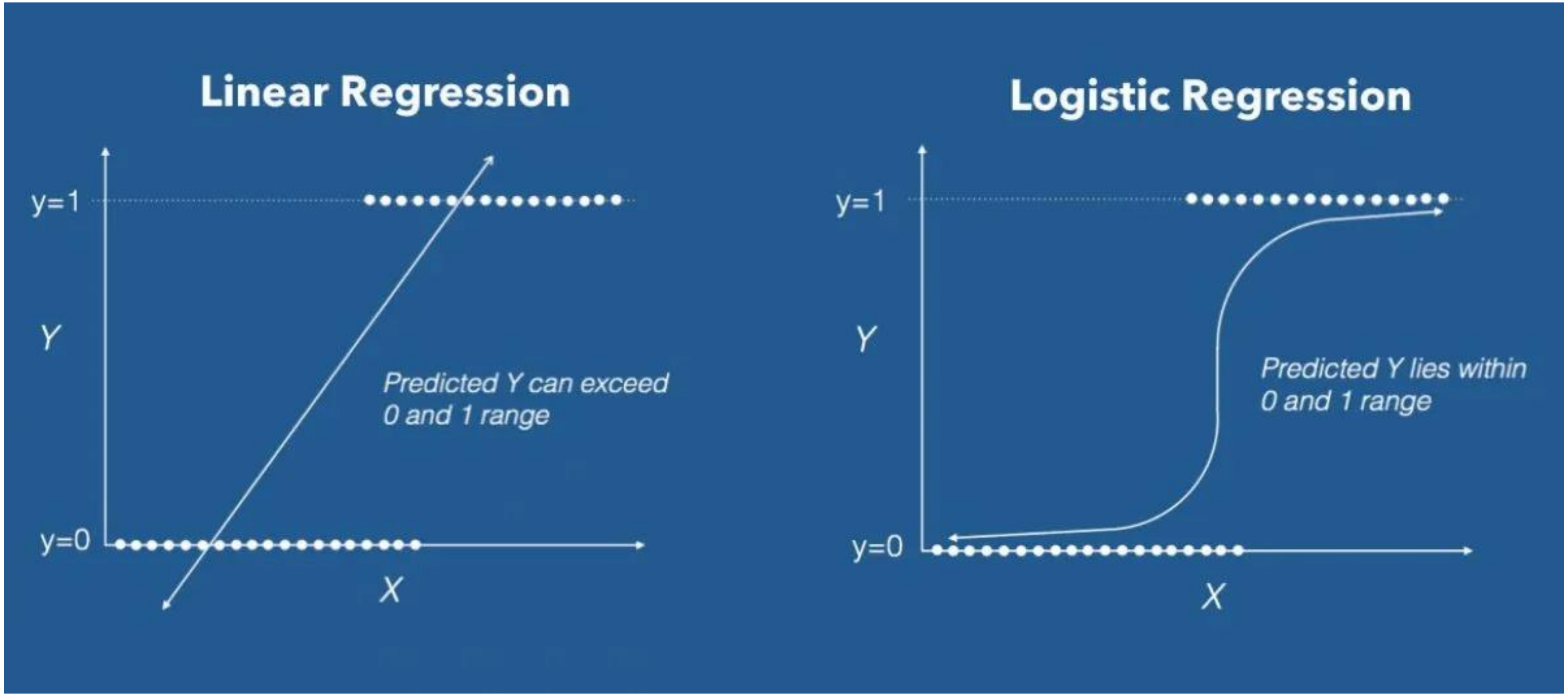
Logistic regression, although termed 'regression' it is used for classification!

- Given are M training samples $(\boldsymbol{X}_{m,:},\ y_m)$
- Each $\boldsymbol{X}_{m,:}$ is a *N*-dimensional **feature vector**

  the features (independent variables) can be continuous or discrete
- $y_m \in \{0, 1\}$ are the **discrete** *labels.*

  Binary classification: 0/1, i.e. true/false, positive/negative …

Why not regression?



The hypothesis in logistic regression

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Logistic Function (or "Sigmoid Function"):

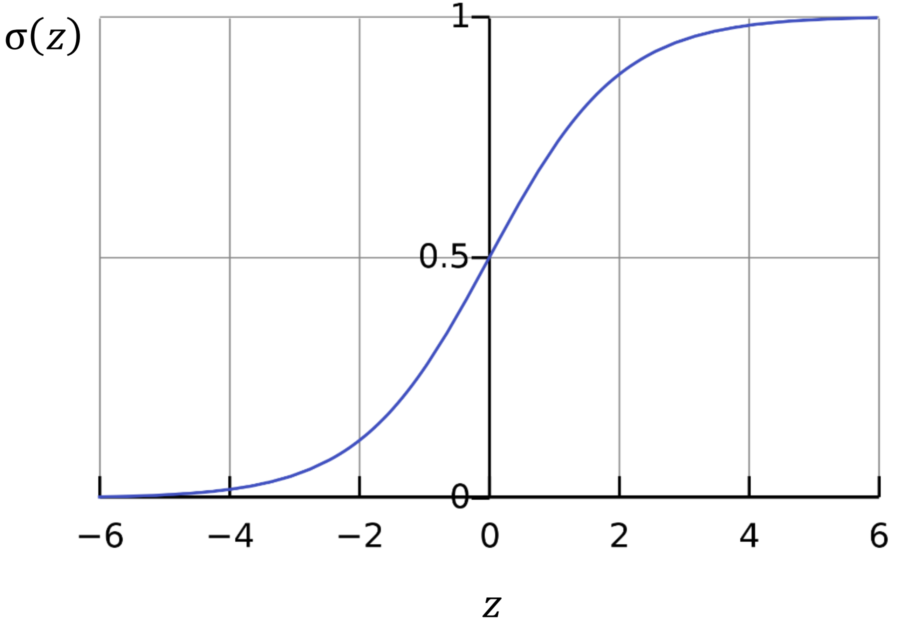
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

with $z = \theta_0 + \theta_1 x = \boldsymbol{\theta}^T \boldsymbol{X}_{m,:}$

Properties of the sigmoid σ :

- smooth distribution between 0 and 1
- $\sigma(z) = 0.5$
- convenient derivatives:

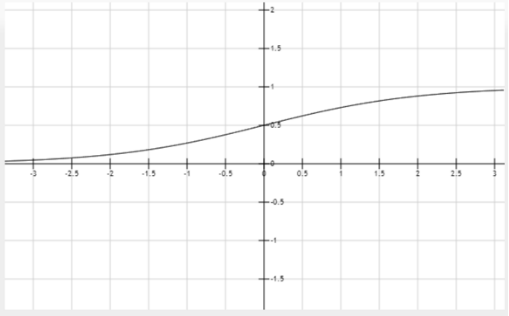
$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$$



Hypothesis: $h(\boldsymbol{X}_{m,:}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{X}_{m,:})$

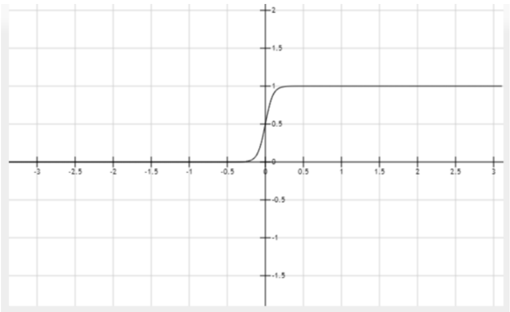
Effect of the parameters in logistic Function

$\sigma(z)$



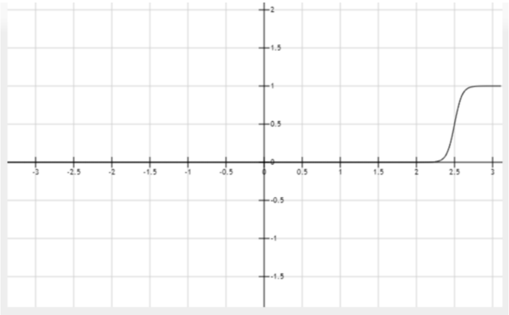
x

$$\frac{1}{1+e^{-(0+1\cdot x)}}$$



x

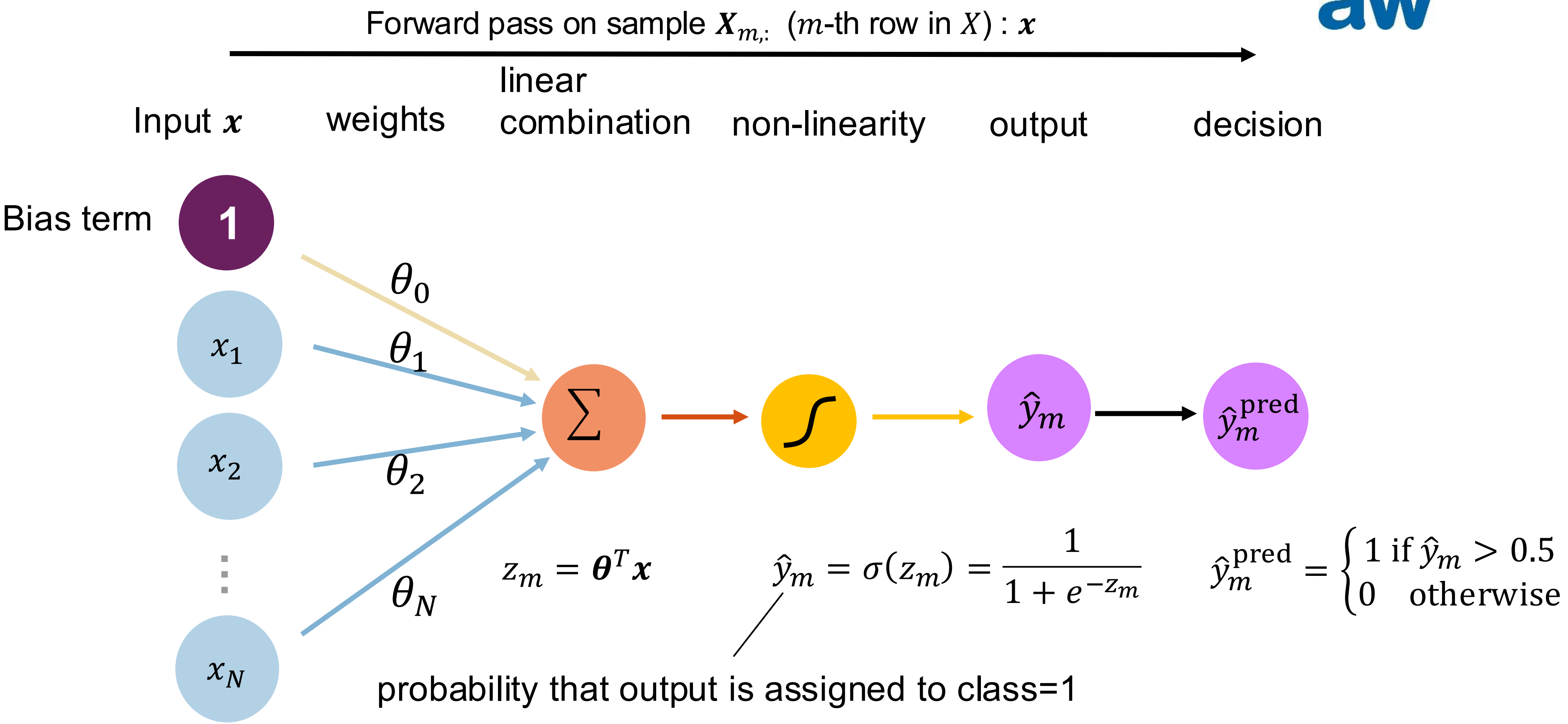
$$\frac{1}{1+e^{-(0+20\cdot x)}}$$



x

$$\frac{1}{1+e^{-(50+20\cdot x)}}$$

Logistic regression for one sample, N features



The loss function in logistic regression

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw School of Engineering

For the N-dimensional training example $\boldsymbol{X}_{m,:}$ the model yields a value $0 < \hat{y}_m < 1$
If the corresponding label y_m is 1, the likelihood of y_m being the positive class according to the model, is given by $\hat{y}_m$. If y_m is 0, the likelihood of y_m being the class 0 is given by $1 - \hat{y}_m$
This leads to the likelihood of the M (independent) training samples according to the model:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{N} (\hat{y}_m)^{y_m} \ (1 - \hat{y}_m)^{1-y_m}$$

The log likelihood is more practical:

$$\log L(\boldsymbol{\theta}) = \sum_{m=1}^{M} y_m \log \hat{y}_m + (1 - y_m) \log(1 - \hat{y}_m)$$

Log is strictly increasing → any θ that maximize the log likelihood also maximise the likelihood, and vice versa.
The base of the logarithm is not important, but needs to be applied consistently

The cross entropy loss

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

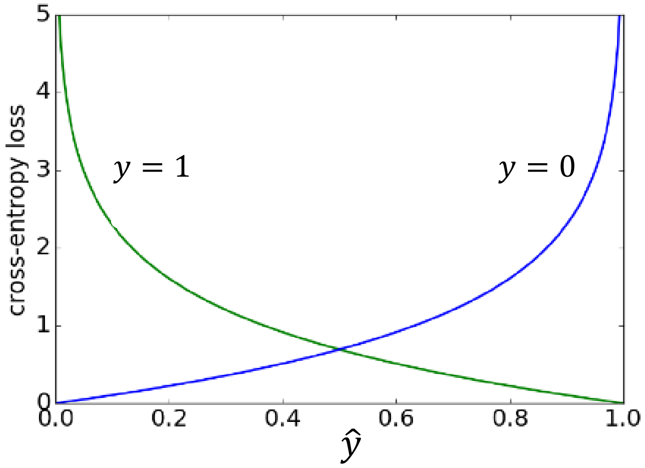
The cross-entropy loss (CE) is the negative log-likelihood

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}) = -\sum_{m=1}^{M} y_m \log \hat{y}_m + (1 - y_m)\log(1 - \hat{y}_m)$$

with y_m: binary $\{0,1\}$, $\hat{y}_m$ scalar in the interval $[0,1]$

With natural logarithm:

$\boldsymbol{y_m}$	$\boldsymbol{\hat{y}_m}$	$\boldsymbol{\mathcal{L}_{CE}}$
0	0.0001	0.0001
0	0.2	0.2231
0	0.5	0.6931
0	0.8	1.6094
0	0.9999	9.2103
1	0.9999	0.0001
1	0.8	0.2231
1	0.5	0.6931
1	0.2	1.6094
1	0.0001	9.2103



The cost function in logistic regression

Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

Average cross entropy over all training samples:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{m=1}^{M} y_m \log \hat{y}_m + (1-y_m)\log(1-\hat{y}_m)$$

$$= -\frac{1}{M}\sum_{m=1}^{M} y_m \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{X}_{m,:}}}\right) + (1-y_m)\log\left(1-\frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{X}_{m,:}}}\right)$$

Linear regression vs. logistic Regression

Linear Regression

Hypothesis: $h\left(\boldsymbol{X}_{m,:}, \boldsymbol{\theta}\right) = \boldsymbol{\theta}^T \boldsymbol{X}_{m,:}$

Cost Function: Average sum of squared residuals

$$J(\boldsymbol{\theta}) = \frac{1}{2M} \sum_{i=1}^{m} (y_m - \hat{y}_m)^2$$

$$= \frac{1}{2M} \sum_{m=1}^{M} (y_m - \boldsymbol{\theta}^T \boldsymbol{X}_{m,:})^2$$

Gradient of the cost function with respect to the parameters:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_n} = \frac{1}{M} \sum_{m=1}^{M} (y_m - \hat{y}_m)(-x_{mn})$$

Logistic Regression

$h\left(\boldsymbol{X}_{m,:}, \boldsymbol{\theta}\right) = \sigma\left(\boldsymbol{\theta}^T \boldsymbol{X}_{m,:}\right)$

Average cross entropy

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{m=1}^{M} y_m \log \hat{y}_m + (1 - y_m) \log(1 - \hat{y}_m)$$

$$= -\frac{1}{M} \sum_{m=1}^{M} y_m \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}_{m,:}}}\right) + (1 - y_m) \log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}_{m,:}}}\right)$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_n} = \frac{1}{M} \sum_{m=1}^{M} (y_m - \hat{y}_m)(-x_{mn})$$

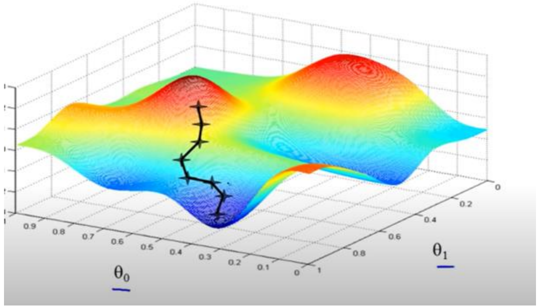
Gradient Descent for Logistic Regression

Cost function: $\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{m=1}^{M} y_m \log \hat{y}_m + (1 - y_m)\log(1 - \hat{y}_m)$

This leads[1] to the following update rule in Gradient Descent
Repeat until convergence:

$$\theta_n = \theta_n - \alpha \frac{\partial}{\partial \theta_n}\mathrm{J}(\boldsymbol{\theta}) \text{ --> } \quad \theta_n = \theta_n - \alpha \frac{1}{M}\sum_{m=1}^{M}(y_m - \hat{y}_m)(-x_{mn})$$

This is similar to the Least Mean Squares Update Rule in Linear Regression, except that here we use a **non-linear function** in the hypothesis to compute the outputs.



[1]For mathematical derivation, see Andrew Ng: CS229 Lecture notes http://cs229.stanford.edu/notes/cs229-notes1.pdf

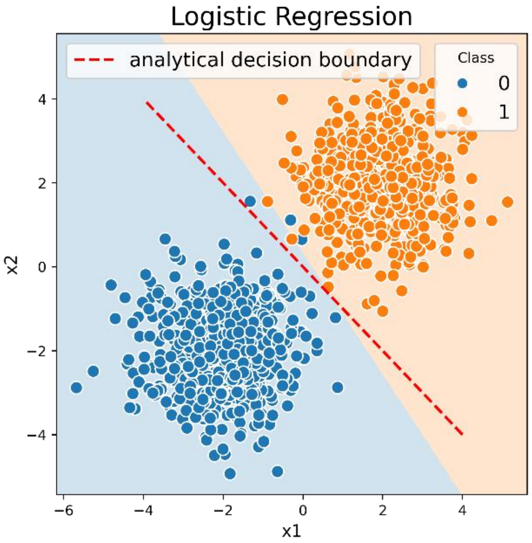
Logistic regression is a linear model for binary classification problems

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

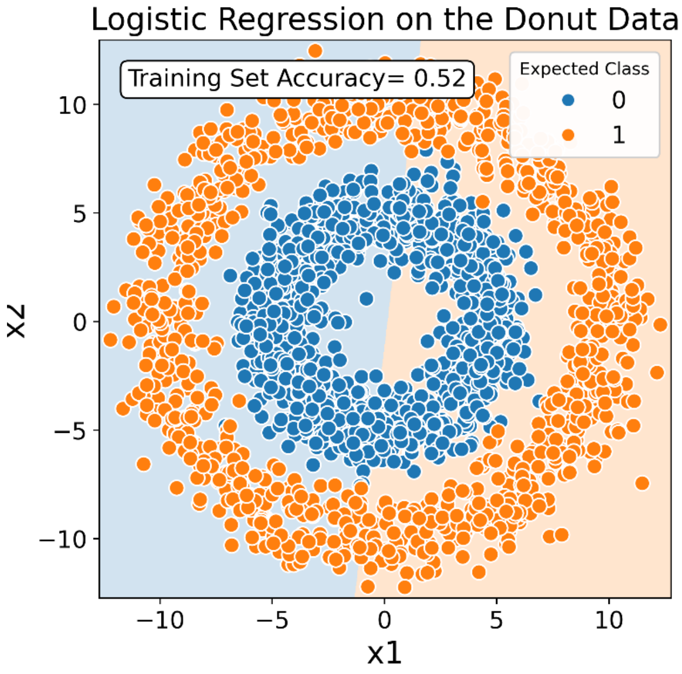
It learns linear decision boundaries between two classes by minimising the cross entropy loss

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{m=1}^{M} y_m \log \hat{y}_m + (1 - y_m)\log(1 - \hat{y}_m)$$

with respect to the model parameters $\boldsymbol{\theta}$ using gradient descent



It fails with data, that is not linearly separable



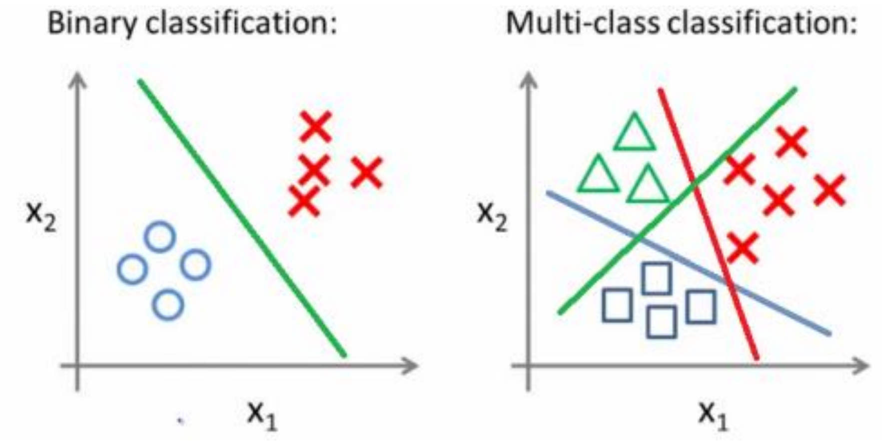
Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Multinomial Classification

Multinomial (Multi-class) Classification

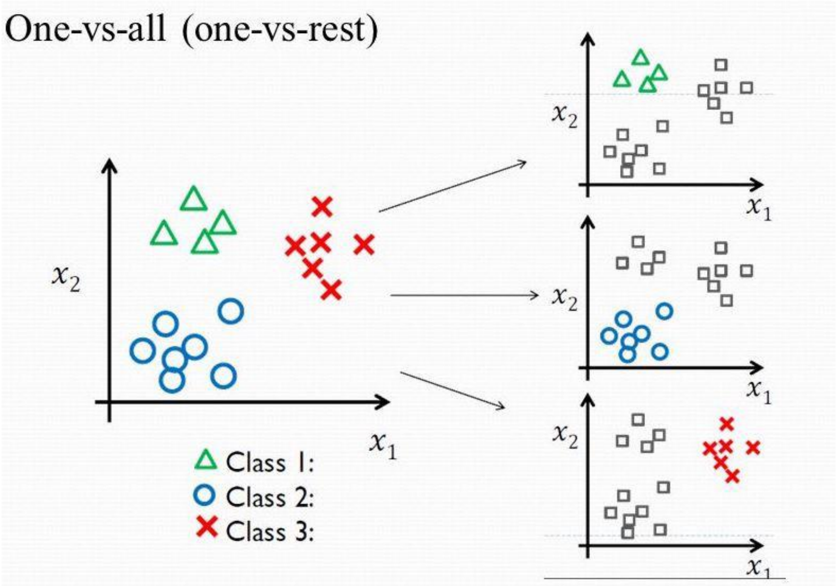
- **Binary vs multinomial classification**



- **Examples of multinomial classification**
  - **Visual digit recognition in OCR**
  - **Obstacle recognition in autonomous driving**

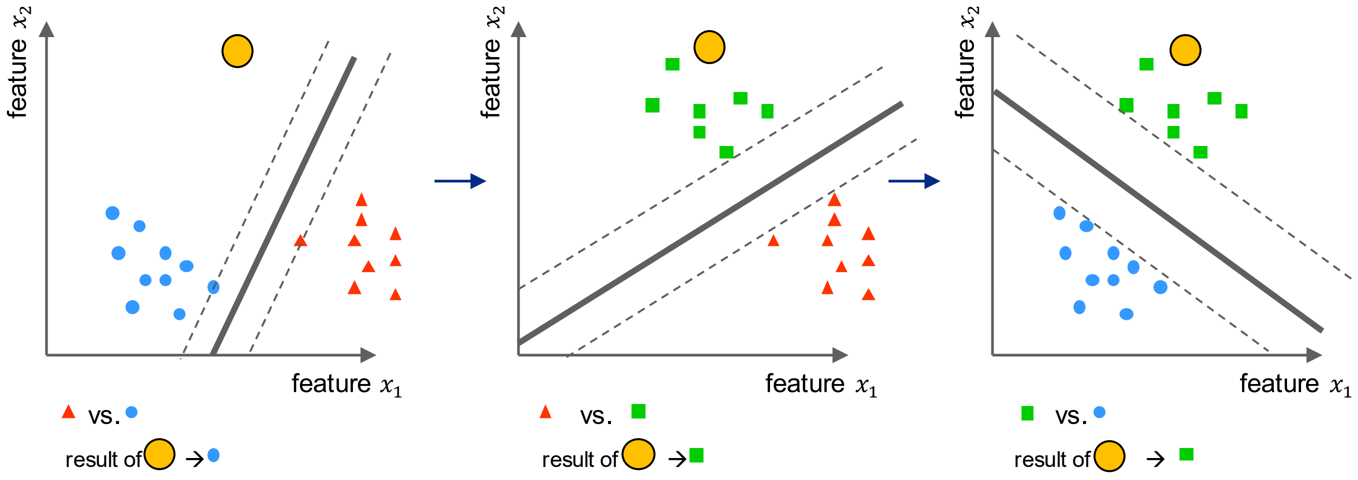
One-vs-Rest approach for multinomial classification

- C classes
- C binary classification problem: each class vs all the others
- C probabilities p_c
- Choose the highest probability to predict the class.



One vs. One

- assume C classes and learn all $C(C-1)/2$ pairwise comparisons
- classify unknown instance by majority vote among all pairwise comparisons



Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

Softmax regression

The labels are now also encoded in matrix form

Example with $M = 4$ samples and $C = 3$ classes {1,2,3}:

$$\boldsymbol{y}_{M\times 1} := \begin{pmatrix} 1 \\ 1 \\ 3 \\ 1 \end{pmatrix} \xrightarrow{\text{one−hot encoding}} \boldsymbol{Y}_{M\times C} := \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

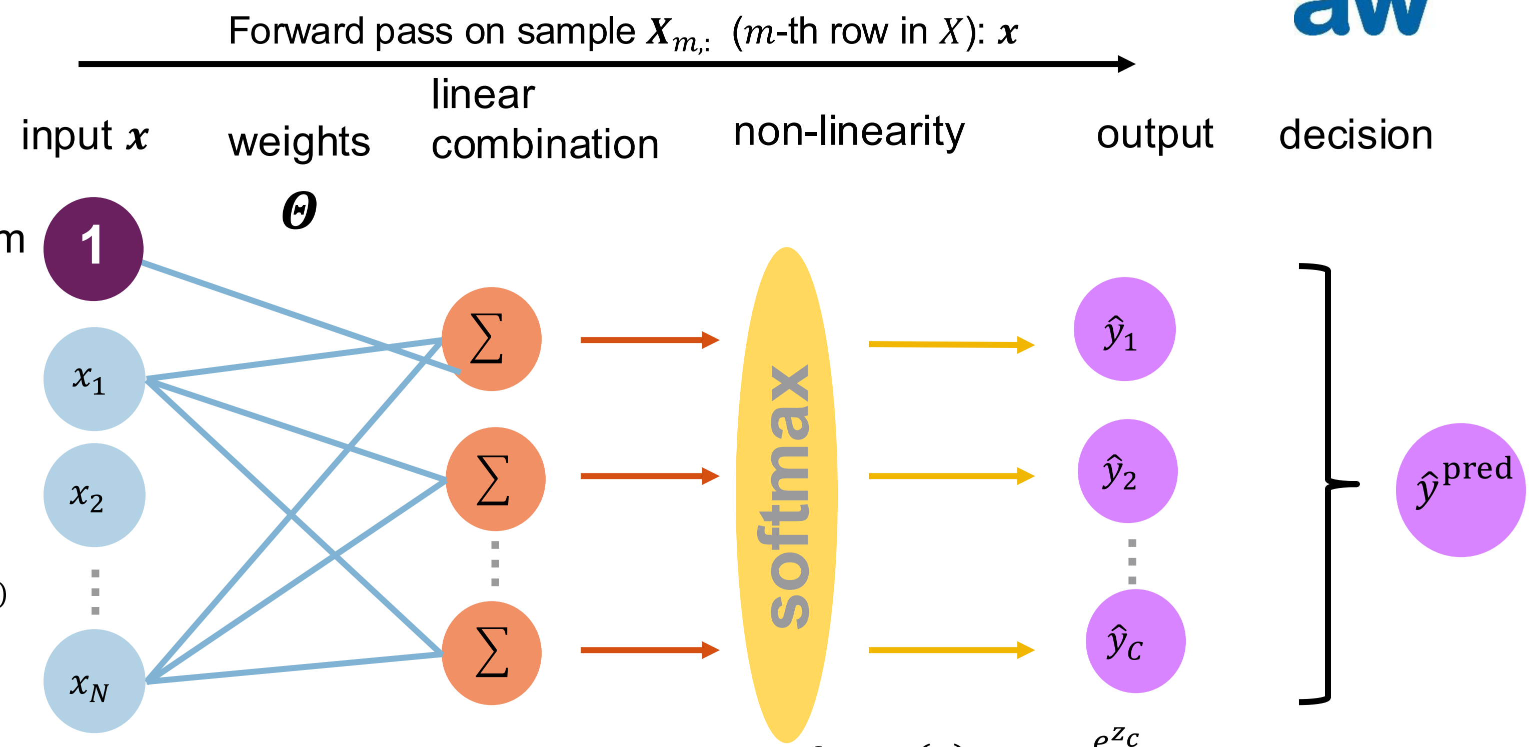
M training samples, C classes

$$\boldsymbol{Y}_{M\times C} := \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1C} \\ y_{21} & y_{22} & \cdots & y_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ y_{M1} & y_{M2} & \cdots & y_{MC} \end{pmatrix}$$

m-th row (encoded as a column vector)

$\boldsymbol{Y}_{m,:} = \begin{pmatrix} y_{m1} \\ y_{m2} \\ \vdots \\ y_{mC} \end{pmatrix}$ is the one-hot encoding of the label for sample m, so $y_{mc} = \mathbb{I}(y_m = c)$

Softmax regression



Dimensions
N features
M samples
C classes
$\boldsymbol{x}: (N+1) \times 1$
$\boldsymbol{\Theta}: C \times (N+1)$
$\boldsymbol{z}: C \times 1$
$\hat{\boldsymbol{y}}: C \times 1$

$$\boldsymbol{z} = \boldsymbol{\Theta}\boldsymbol{x} \qquad \hat{y}_c = \text{softmax}(\boldsymbol{z})_c = \frac{e^{z_c}}{\sum_{c'=1}^{C} e^{z_{c'}}} \qquad \text{argmax}$$

Probability that sample $\boldsymbol{x}$ is assigned to class label c

The learning in softmax regression

The loss function for multinomial logistic regression generalises the loss function for binary logistic regression from 2 to C classes:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{c=1}^{C} y_{mc}\log \hat{y}_{mc}$$

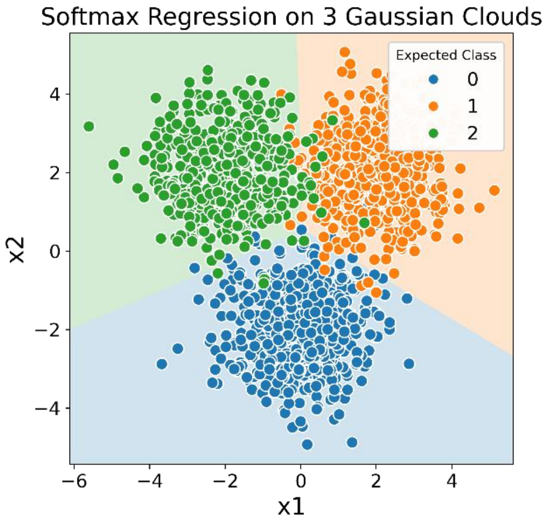
$$= -\frac{1}{M}\sum_{m=1}^{M} y_{mk}\log \hat{y}_{mk}$$ (where k is the correct class index for sample m)

$$= -\frac{1}{M}\sum_{m=1}^{M} 1\cdot\log\frac{\exp(z_k)}{\sum_{c=1}^{C}\exp(z_c)} = -\frac{1}{M}\sum_{m=1}^{M}\log\frac{\exp(\boldsymbol{\Theta}_{k,:}^{T}\boldsymbol{X}_{m,:})}{\sum_{c=1}^{C}\exp(\boldsymbol{\Theta}_{c,:}^{T}\boldsymbol{X}_{m,:})}$$

⟶ Minimisation by gradient descent

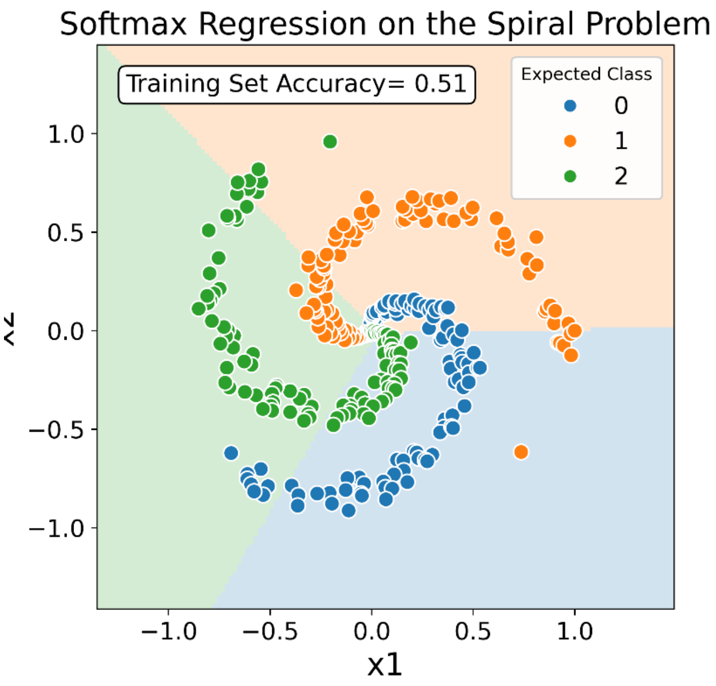
Softmax regression for multinomial classification

Example: 3 Classes sampled from a multivariate normal distribution



Softmax Logistic Regression still learns linear decision boundaries

Softmax regression fails non non-linear problems



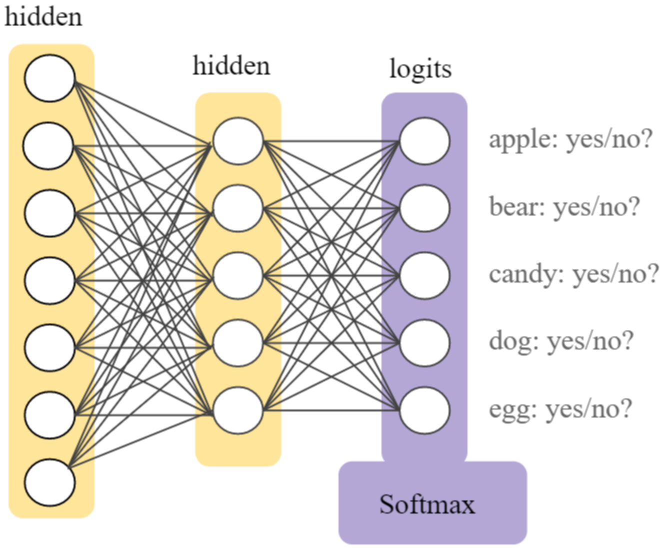
Softmax vs One-vs-rest

One-vs-rest

Each class has its own model

Softmax

One model for all classes



Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

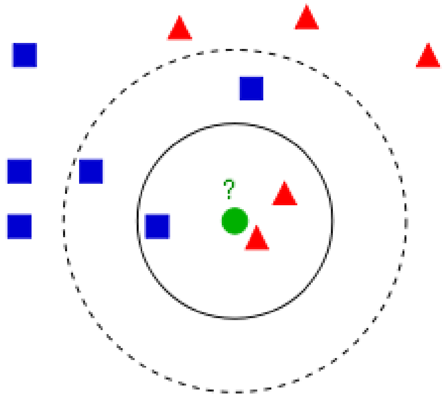
Nearest Neighbor Classification

Instance Based Classification: K-Nearest Neighbor (KNN)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

- The (input,output)-pairs of the training data
  $$\{(\boldsymbol{X}_{m,:}, y_m) | 1 \leq m \leq M\}$$
  is saved in a search structure
- k: user defined hyperparameter
- To label an **unknown data point**, the **most frequent label among the k closest training samples** is determined → red

- Requires a distance metric!
  - E.g. cosine distance



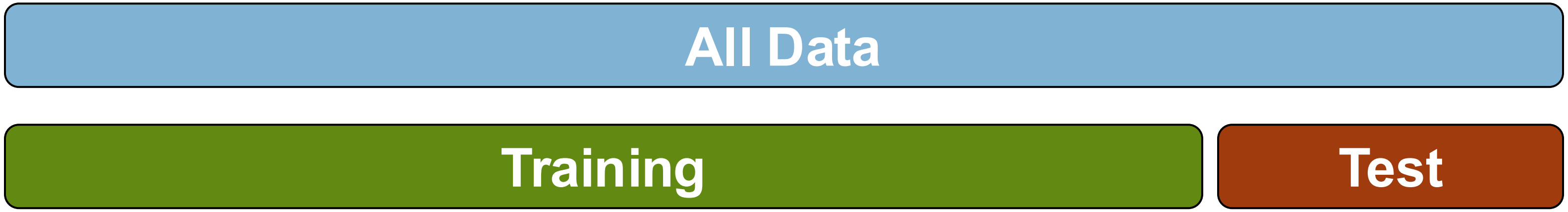
2 Classes: $y_m \in \{\mathrm{red}, \mathrm{blue}\}$

11 training samples: $M = 11$

Evaluation metrics for classification

Can we estimate quantitatively how well the model generalises beyond the training data?

General for supervised machine learning: Split the available data into a training and independent test set



Estimate the generalisation error based on the independent test set

Classification Error and Accuracy - example

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

inputs x_m	expected outputs y_m	predicted outputs $\hat{y}_m$
	1	1
	0	0
	1	1
	1	0
	1	0
	1	0
	0	1

$$\mathbb{I}(\hat{y}_m \neq y_m)$$

$$M = ?$$

$$\text{Error} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(\hat{y}_m \neq y_m)$$

$$=$$

$$\text{Accuracy} = 1 - \text{Error} =$$

Classification error and accuracy

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

The misclassification error for one sample x_m:$\mathbb{I}(\hat{y}_m \neq y_m) = \begin{cases} 1 \text{ if } \hat{y}_m \neq y_m \\ 0 \text{ otherwise} \end{cases}$

delivers a loss of 1 if the predicted value $\hat{y}_m$ does not agree with the expected value y_m

The misclassification rate (standard error) is the average over all samples:

$$\text{Error} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(\hat{y}_m \neq y_m)$$

$$\text{Accuracy} = 1 - \text{Error}$$

Classification error and accuracy do not distinguish between the different types of errors – some might be more severe than others.

Types of classification errors

Binary classification problem: 0/1

predicted

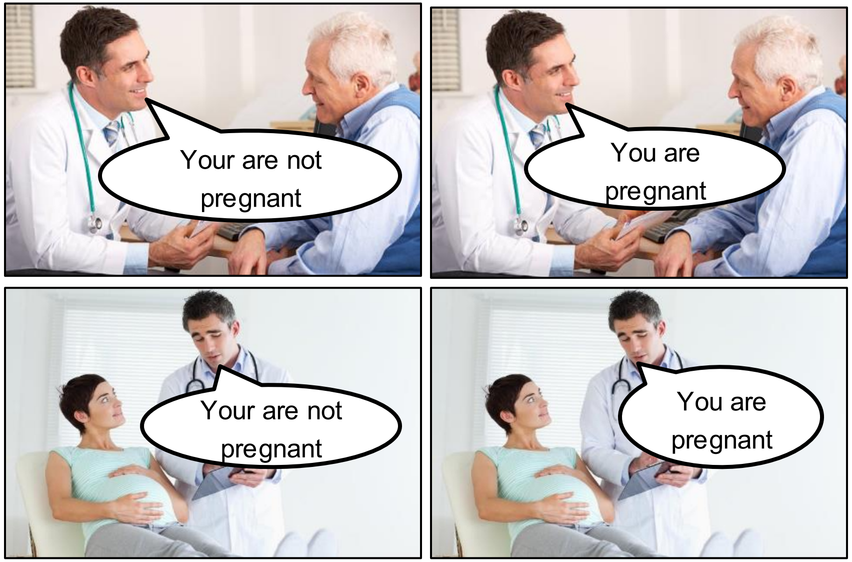
expected

$y \downarrow / \hat{y} \rightarrow$	Not Hot Dog (0)	Hot Dog (1)
Not Hot Dog (0)	TN 	 FP
Hot Dog (1)	FN 	 TP

- True Positives (TP - hit): The actual class was 1 (true) and the predicted is also 1 (true)
- True Negatives (TN): The actual class was 0 (false) and the predicted is also 0 (false)
- False Positives (FP – false alarm, type-I error): The actual class was 0 (false) and the predicted is 1 (true)
- False Negatives (FN – miss, type-II error): The actual class was 1 (true) and the predicted is 0 (false)

Types of classification errors

TN FP

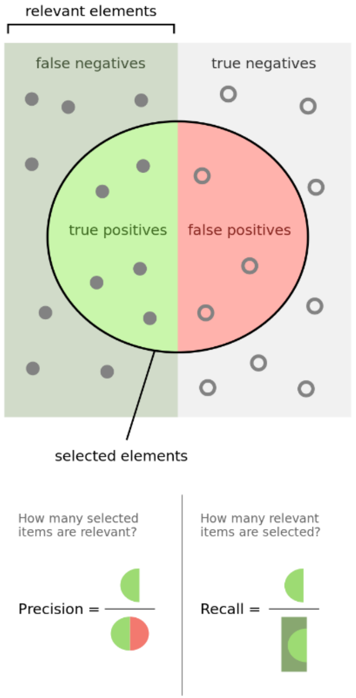


FN TP

Depending on the use case the types of errors in classification have different impact. This has to be reflected in the selection of the evaluation metrics.

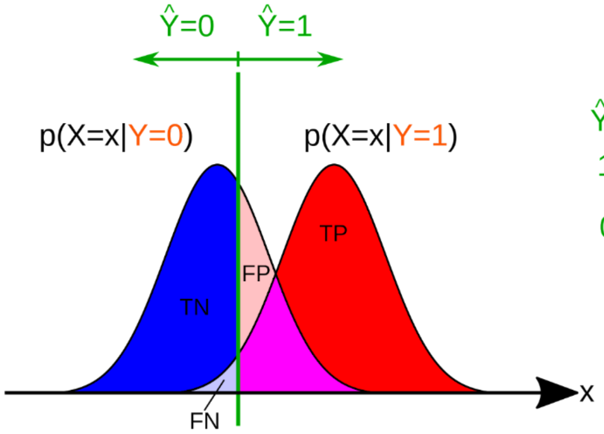
Evaluation metrics for classification

- Accuracy and Error
  - $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$
  - $\text{Error} = 1 - \text{Accuracy}$
  - Do not take into account different «costs» of errors
- Recall
  - $\text{Recall} = \frac{TP}{TP+FN}$ penalises FN, but ignores FP
  - Fraction of correctly predicted positive elements in relation to all positive elements (relevant)
- Precision
  - $\text{Precision} = \frac{TP}{TP+FP}$ penalises FP, but ignores FN
  - Fraction of correctly predicted positive elements in relation to all positive predicted elements (selected)



ROC Curve and Precision-Recall Curve Plot

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering



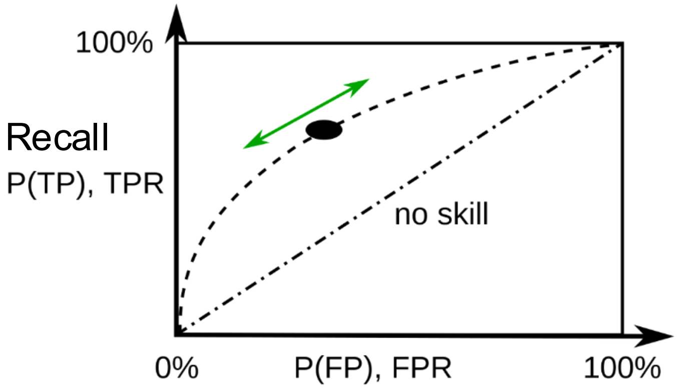
Ŷ \ Y	1	0
1	TP	FP
0	FN	TN

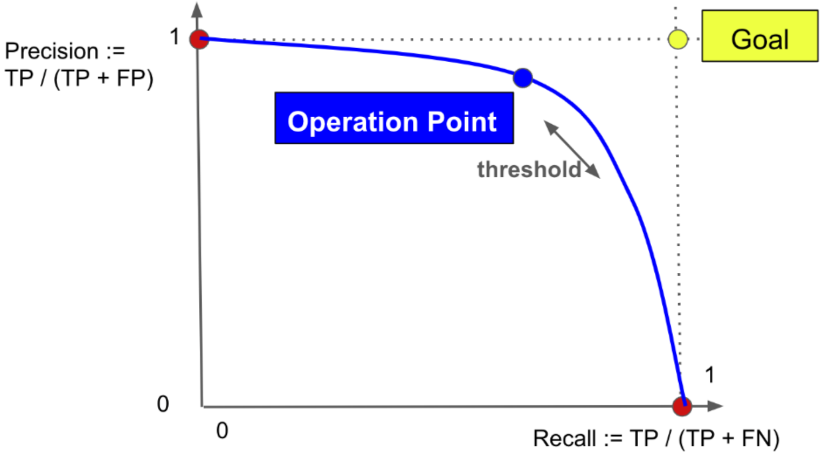
False Positive Rate

FP / #neg = FP / (TN + FP)

True Positive Rate (Recall)

TP / #pos = TP / (TP + FN)

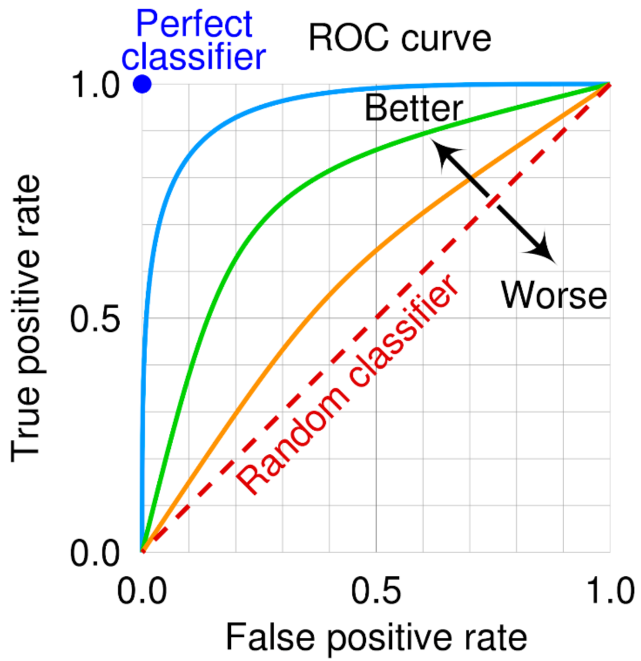




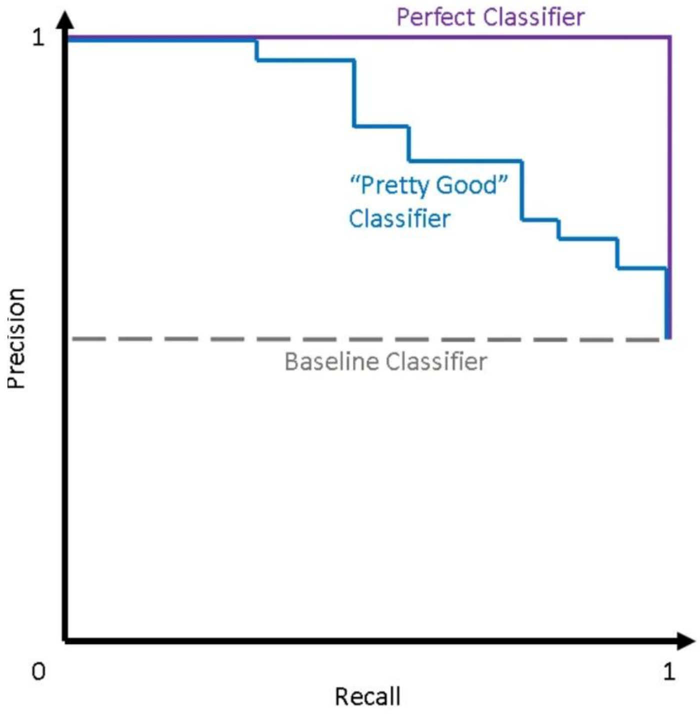
ROC Curve & PR curve

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

ROC Curve



PR curve



Evaluation metrics for classification: F-Score

Combination of recall and precision

Definition of the general $\boldsymbol{F_\beta}$-Score:

$$F_\beta = (1 + \beta^2) * \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

F_1-Score:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

A smaller $\boldsymbol{\beta}$ value, such as 0.5, gives more weight to precision and less to recall, whereas a larger $\boldsymbol{\beta}$ value, such as 2.0, gives less weight to precision and more weight to recall in the calculation of the F-score

Multi-Class Confusion Matrix

True Class \ Predicted Class	truck	ship	horse	automobile	frog	deer	dog	airplane	bird	cat		
truck	923	9	1	39	2			20	3	3	92.3%	7.7%
ship	10	931	1	10	2		1	37	4	4	93.1%	6.9%
horse	4	2	915	1	3	22	17	9	14	13	91.5%	8.5%
automobile	15	5	1	972				5	2		97.2%	2.8%
frog		1	1		943	3	4	5	16	27	94.3%	5.7%
deer	1	2	14	1	14	898	13	5	28	24	89.8%	10.2%
dog	3		17	2	13	18	801	7	28	111	80.1%	19.9%
airplane	6	23	5	4	5	4	1	923	21	8	92.3%	7.7%
bird	3	4	5	2	17	13	8	26	892	30	89.2%	10.8%
cat	7	5	12	4	30	24	48	12	32	826	82.6%	17.4%
	95.0%	94.8%	94.1%	93.9%	91.6%	91.4%	89.7%	88.0%	85.8%	79.0%		
	5.0%	5.2%	5.9%	6.1%	8.4%	8.6%	10.3%	12.0%	14.2%	21.0%		

https://www.mathworks.com/help/examples/nnet/win64/SortClassesAfterPrecisionOrRecallExample_03.png

Evaluation metrics for multi-class classification

Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

Medical diagnosisin $C = 3$ classes: {Healthy, Cold, Flue}

100 test results:

Predicted label

True label

c	Healthy	Cold	Flue	Support of ***c***
Healthy	60	8	2	70
Cold	4	12	4	20
Flue	0	2	8	10
				100

Metrics per class:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

$$F1_c = 2\frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Average metrics:

$$\text{Metric}_{\text{macro}} = \frac{\sum_{c=1}^{C} \text{Metric}_c}{C}$$

$$\text{Metric}_{\text{weighted}} = \sum_{c=1}^{C} \frac{\text{Metric}_c}{\text{Support}_c}$$

$$\text{Accuracy} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1} \text{Support}_c}$$

Evaluation metrics for multi-class classification

Medical diagnosis in $C = 3$ classes: {Healthy, Cold, Flue}

100 test results:

Predicted label

True label

c	Healthy	Cold	Flue	Support of c
Healthy	60	8	2	70
Cold	4	12	4	20
Flue	0	2	8	10
				100

	Precision	Recall	F_1-Score
Healthy	0.94	0.86	0.9
Cold	0.54	0.6	0.57
Flue	0.57	0.8	0.67
Unweighted: Macro average	0.68	0.75	0.71
Weighted average			
Accuracy			