Machine learning in Medical Image Analysis

Lecture 4

Semen Kiselev

Recap

1. Segmentation
2. Thresholding
3. Connected component decomposition
4. Morphological operations

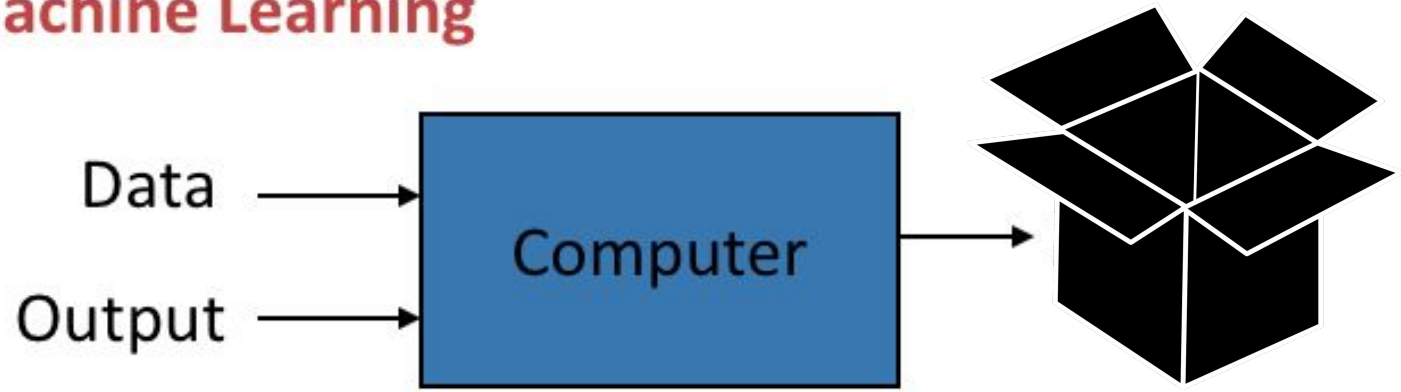
Agenda

1. Glossary
2. Types of learning
3. Types of methods
4. Objective function
5. Confusion matrix and metrics
6. The single number evaluation metric
7. Train-val-test split
8. K-fold cross validation
9. Fully connected neural networks

TLDR;

Traditional Programming

Machine Learning



Metrics

MSE
MAE
MAPE
PSNR
SSIM
ACCURACY
PRECISION
RECALL
F1-score
AUC-ROC
.
.

Glossary

a. **Sample** (*x*) - unit of data (observation, instance). E.g. Image, a vector of numbers

b. **Dataset** - a collection of samples.

c. **Ground-truth** - information provided by **direct observation** as opposed to information provided by inference.
E.g. **human-expert said** that the image contains a **cat**.
Then, the **ground-truth is 'cat'**.

d. **Label** (*y*) - some value associated with the sample, **defining the ground-truth** annotation for that sample.
   i. **categorical** <=> classification/categorization learning problem
   ii. **continuous** <=> regression learning problem
   iii. **Can be anything**: image, text, sound, vector …

e. **Prediction** (*y_hat*, ŷ) - the output of an algorithm for some particular sample.

Types of learning

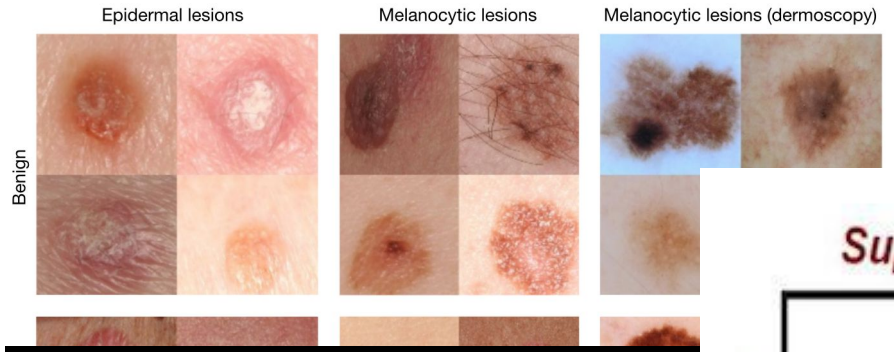
1. **Supervised**
   a. **Labeled** dataset, each sample has a ground-truth label
   b. Typically used to make **predictions based on historical data**.

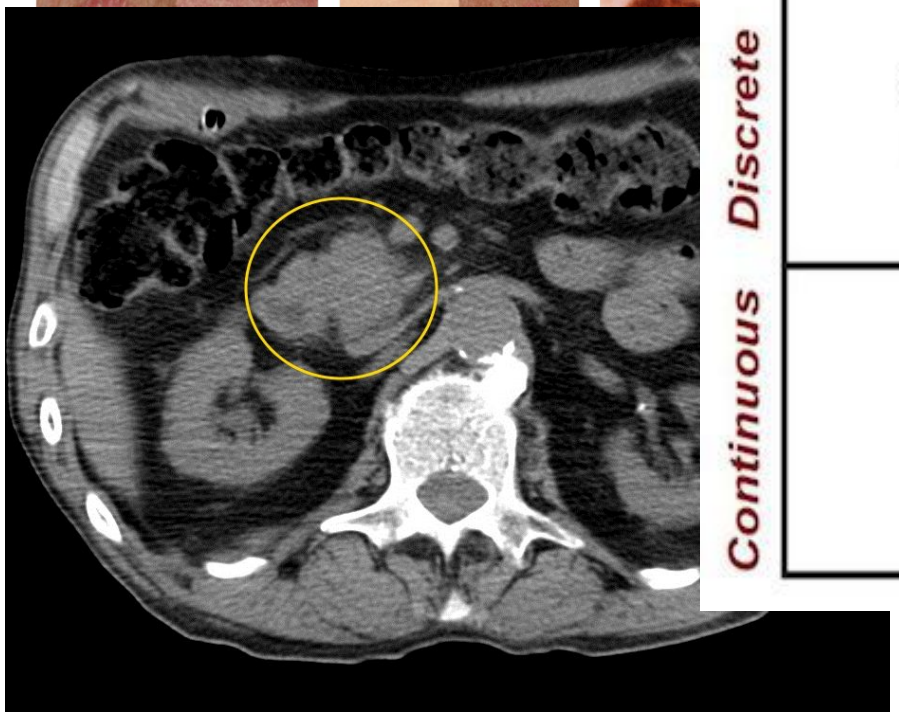
Input	Output	Application
Chest X-ray	Pathology? 0/1	Clinical screening
CT scan	Heart Volume in m^3	Heart diseases
Picture of tumor	Malignant/Benign? 0/1	Cancer detection
Radiologist report	Machine readable labels	Dataset creation

2. **Unsupervised**
   a. **Unlabeled** dataset. Raw data.
   b. Typically used to **discover the underlying structure in data**.
      i. **Clasterisation**: we expect the algorithm to divide the data into a fixed number of groups. I.e. algorithm assigns a categorical label for each sample
      ii. **Dimensionality reduction**: we expect the algorithm to assign a vector of continuous values to each sample. In order to later compare samples using vector space similarity function (e.g. cosine similarity)

3. **Semi-supervised**
   a. The small-portion of the dataset has labels. The **majority has no labels**.
   b. Typically used to make predictions based on historical data in the domain, for which the **annotated dataset is not available**.

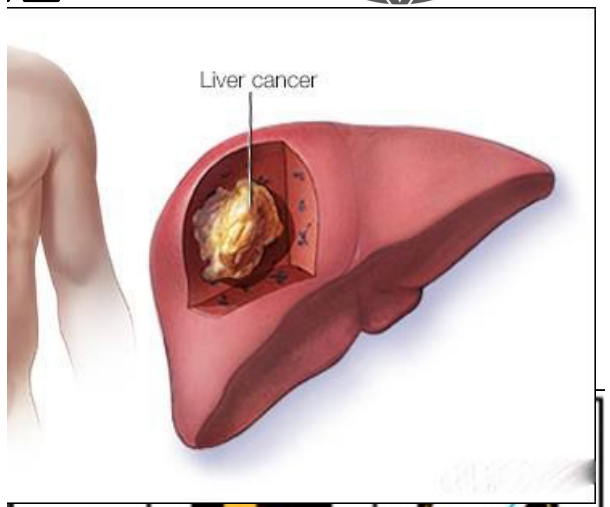
Types of learning

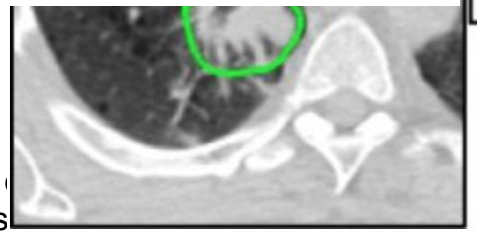


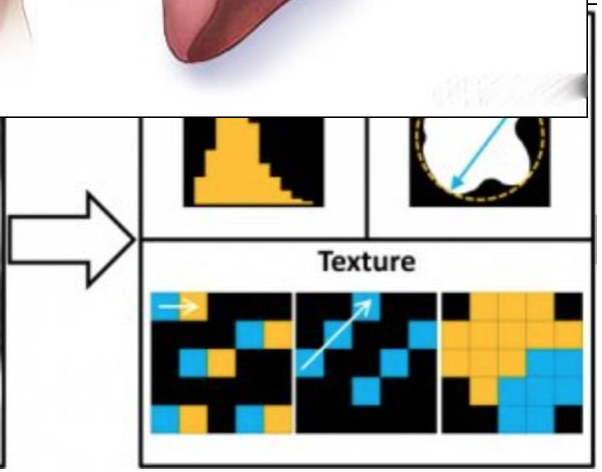


	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

VS.







Esteva, A., Kuprel, B., Novoa, R. et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056

Floyd et al. (2011). Prediction
Selection of Predictive Models
127. 29-43. 10.1007/978-3-642-18472-7_3.

Types of methods

Given: the dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where x_n is a sample, and y_n is its corresponding label.

Our goal: is to apply a statistical learning method to the training data in order to estimate the unknown function f

To find: a function f_hat such that $y \approx \hat{f}(x)$, for any (x, y) from the dataset.

1. **Parametric**: we make an explicit assumption about the complexity of an estimated function f.
2. **Non-parametric**: no explicit assumption about the complexity of f.

Types of methods

Given: the dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where x_n is a sample, and y_n is its corresponding label.

Our goal: is to apply a statistical learning method to the training data in order to estimate the unknown function f

To find: a function f_hat such that $y \approx \hat{f}(x)$, for any (x, y) from the dataset.

1. **Parametric**: we make an explicit assumption about the complexity of an estimated function f.
   Advantages:
   - Reduces the problem of estimating the function to the problem of estimating the fixed set of parameters, which is generally much easier.

   Disadvantages:
   - Model complexity we choose usually does not match the true unknown complexity of f .

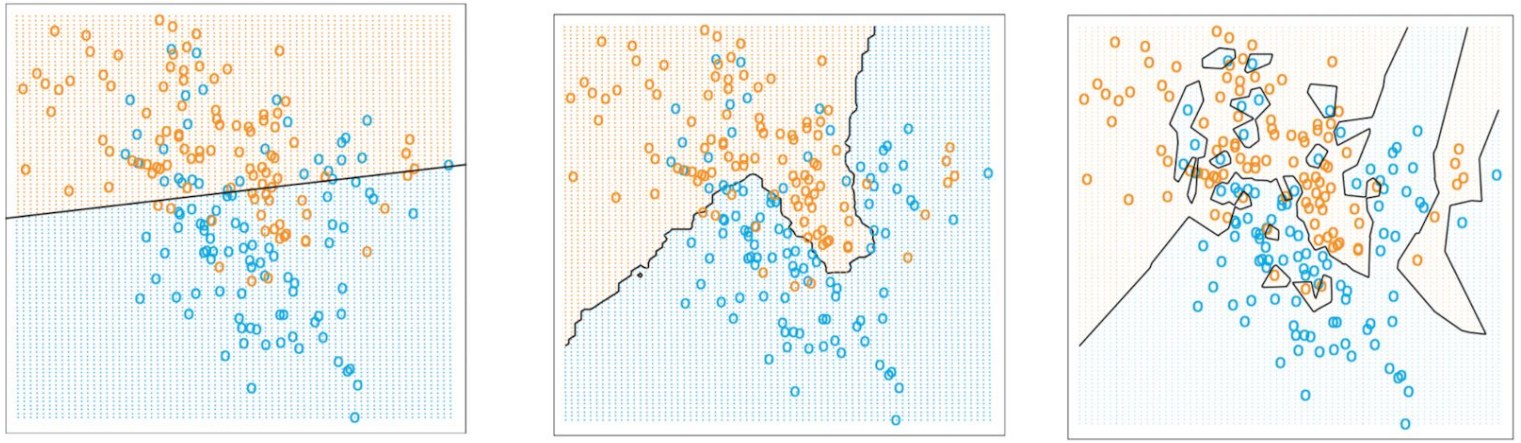
2. **Non-parametric**: no explicit assumption about the complexity of f .
   Advantages:
   - No time required to estimate the model parameters. i.e. no training phase.

   Disadvantages:
   - Enormously large number of observation is required to obtain accurate estimate of f .
   - Making predictions can be time consuming.

Types of methods: Parametric

Parametric: we make an explicit assumption
about the complexity of an estimated function f

Classification



(a) (b) (C)

Examples of complexity assumption:

a) **Linear model**: $\hat{y} = b_0 + b_1 x$

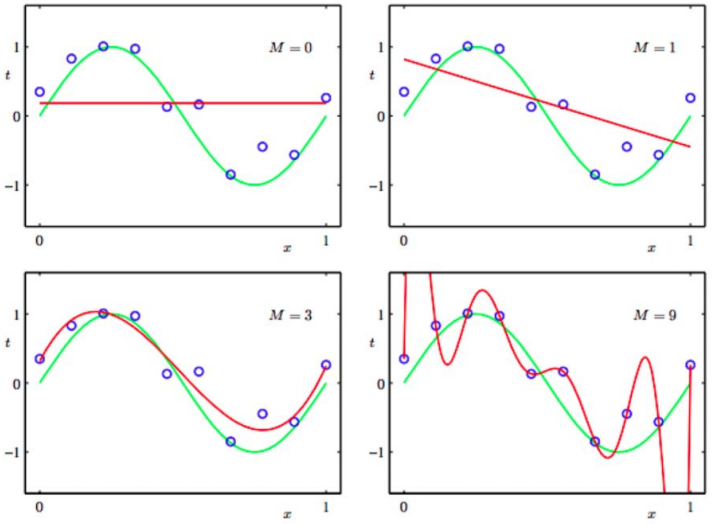
b) **Quadratic model**: $\hat{y} = b_0 + b_1 x + b_2 x^2$

c) **N-degree polynomial**: $\hat{y} = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$

Types of methods: Parametric

Parametric: we make an explicit assumption
about the complexity of an estimated function f

Regularization



Polynomial Curve Fitting: Polynomials having various order M (in red), fitted to the data (in blue) coming from the true underlying curve shown in green.

Examples of complexity assumption:

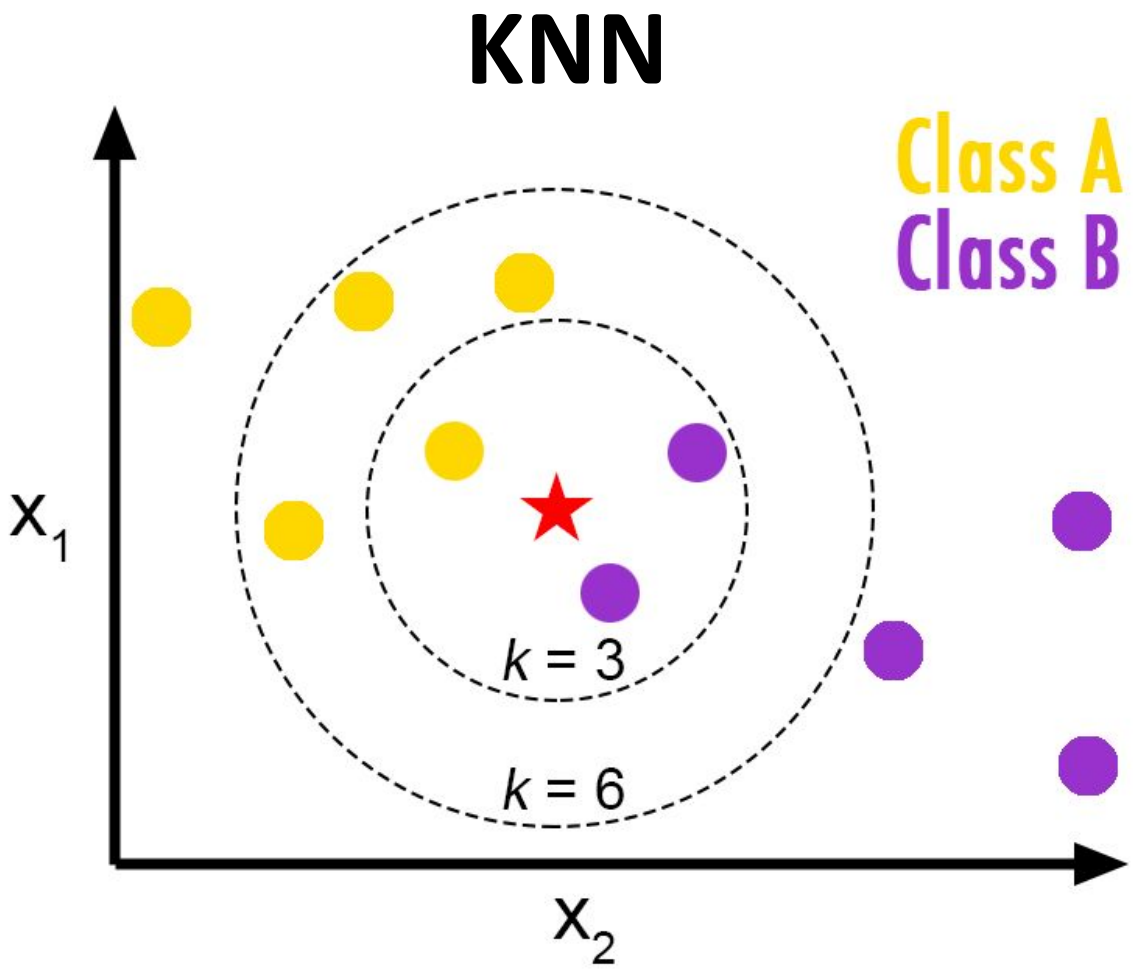
a) **Linear model**: $\hat{y} = b_0 + b_1 x$

b) **Quadratic model**: $\hat{y} = b_0 + b_1 x + b_2 x^2$

c) **N-degree polynomial**: $\hat{y} = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$

Types of methods: Non-parametric

Make an estimate using the data.



Supervised parametric

Objective function

1. In order **to optimize parameters** of our model, we **define Objective function** , which **measures how big the prediction error** of our model is.

    a. In order to estimate parameters of a model, we use an algorithm called **gradient descent**, which tells us how to **change each parameter** of a model, in order **to decrease** the value of an **objective** function.

    b. Essentially it **computes partial derivatives** of objective function w.r.t. each model parameter. Then each parameter is updated in the following manner: $param = param - lr * \frac{\partial Objective}{\partial param}$,

    where lr states for “learning rate” and defines how much the values of parameters are changed in the direction opposite to the direction of steepest ascent (if it is not clear, please refer to the gradient concept from Calculus).

,

Common objective functions

Binary classification. Labels are either 0 or 1 ($y \in \{0, 1\}$). Model predictions are from 0 to 1 ($\hat{y} \in [0, 1]$).

- Binary cross entropy $BCE = -y \log \hat{y} - (1 - y) \log (1 - \hat{y})$

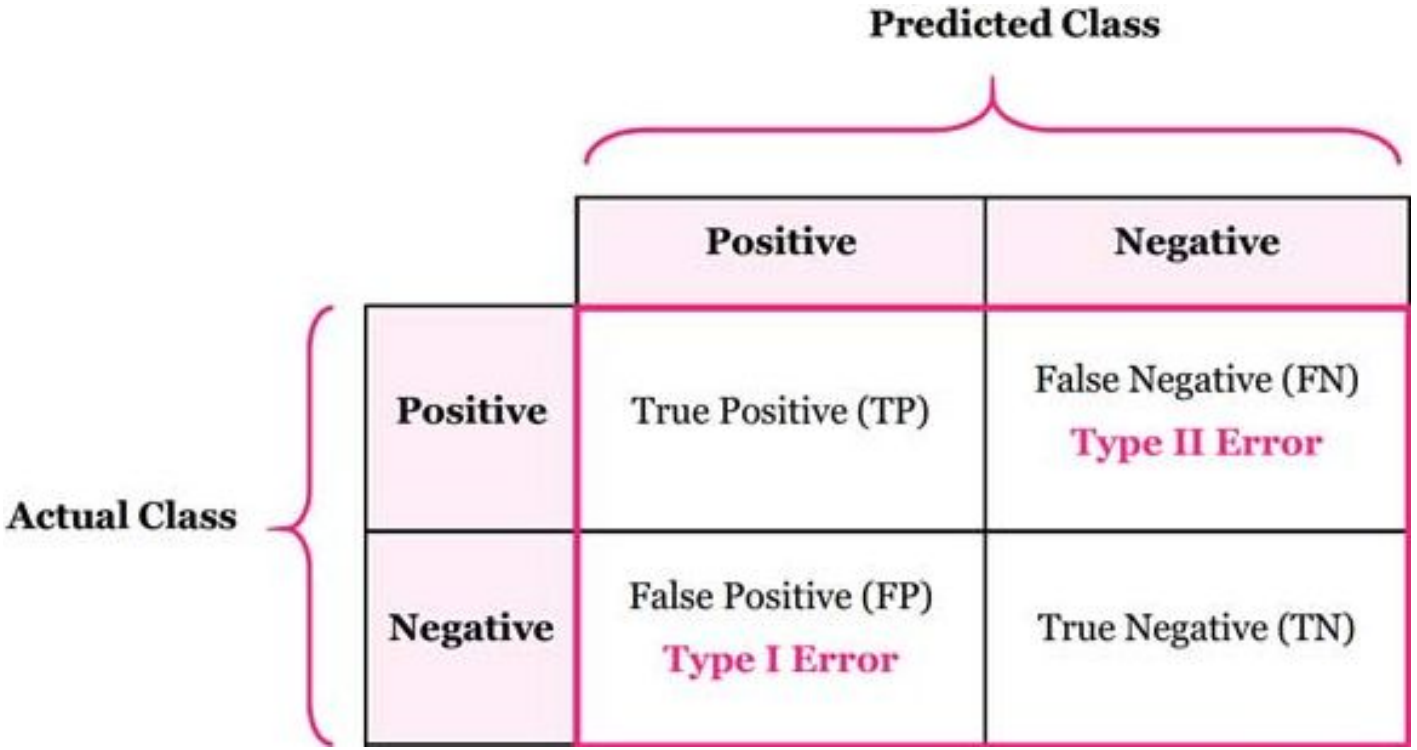
Regression

- Mean absolute error $\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$
- Mean squared error $\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

,

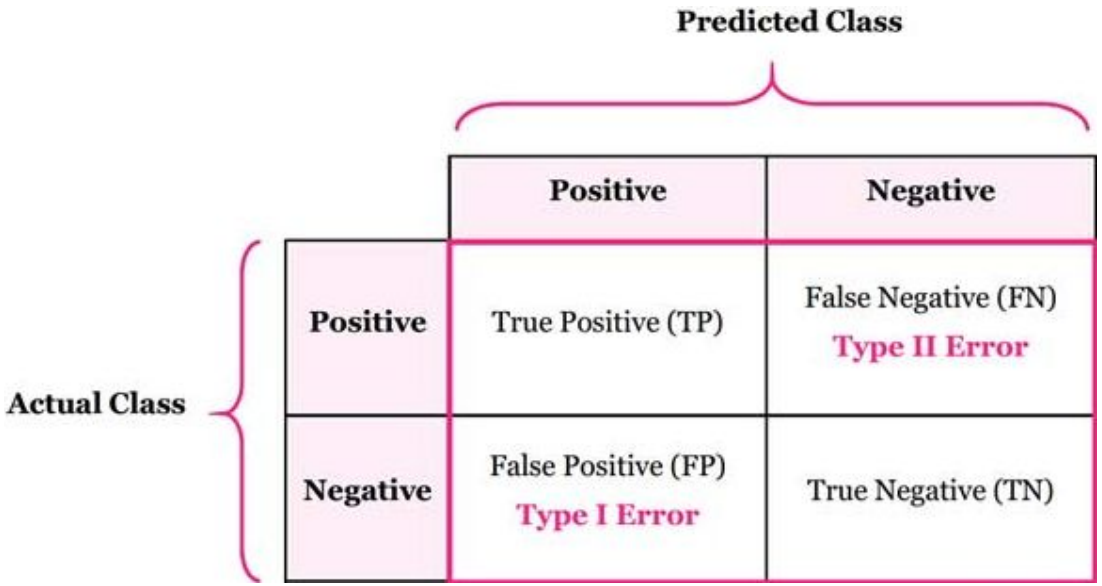
Confusion matrix

- Use case.
- **Binary classification**.
- Labels are either 0 or 1 ($y \in \{0, 1\}$).
- Model predictions (after thresholding) are either 0 or 1 ($\hat{y} \in \{0, 1\}$).
- Thus, using model predictions for all samples in the dataset, **construct the following matrix**:



Confusion matrix

- Use case.
- **Binary classification**.
- Labels are either 0 or 1 ($y \in \{0, 1\}$).
- Model predictions (after thresholding) are either 0 or 1 ($\hat{y} \in \{0, 1\}$).
- Thus, using model predictions for all samples in the dataset, **construct the following matrix**:



1. **TP** (True positive) - number of times your model **predicted positive** class and the ground-truth **label** is **indeed positive**.
2. **TN** (True negative) - number of times your model **predicted negative** class and the ground-truth **label** is **indeed negative**.
3. **FP** (False positive) - number of times your model **predicted positive** class, but the ground-truth **label** is **negative**.
4. **FN** (False negative) - number of times your model **predicted negative** class, but the ground-truth **label** is **positive**.

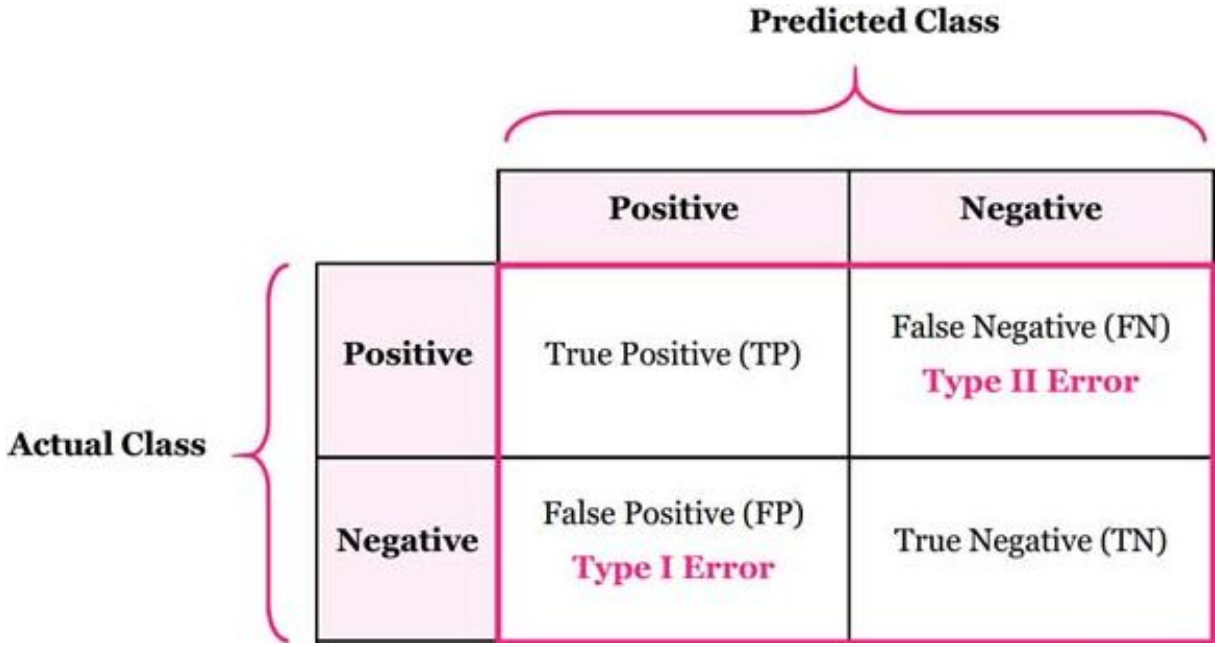
Confusion matrix: importance

1. **TP** (True positive) - number of times your model **predicted positive** class and the ground-truth **label** is **indeed positive**.
2. **TN** (True negative) - number of times your model **predicted negative** class and the ground-truth **label** is **indeed negative**.
3. **FP** (False positive) - number of times your model **predicted positive** class, but the ground-truth **label** is **negative**.
4. **FN** (False negative) - number of times your model **predicted negative** class, but the ground-truth **label** is **positive**.

Assume: **POSITIVE** == patient **with cancer**

Do **FP** and **FN** are still equally important here?

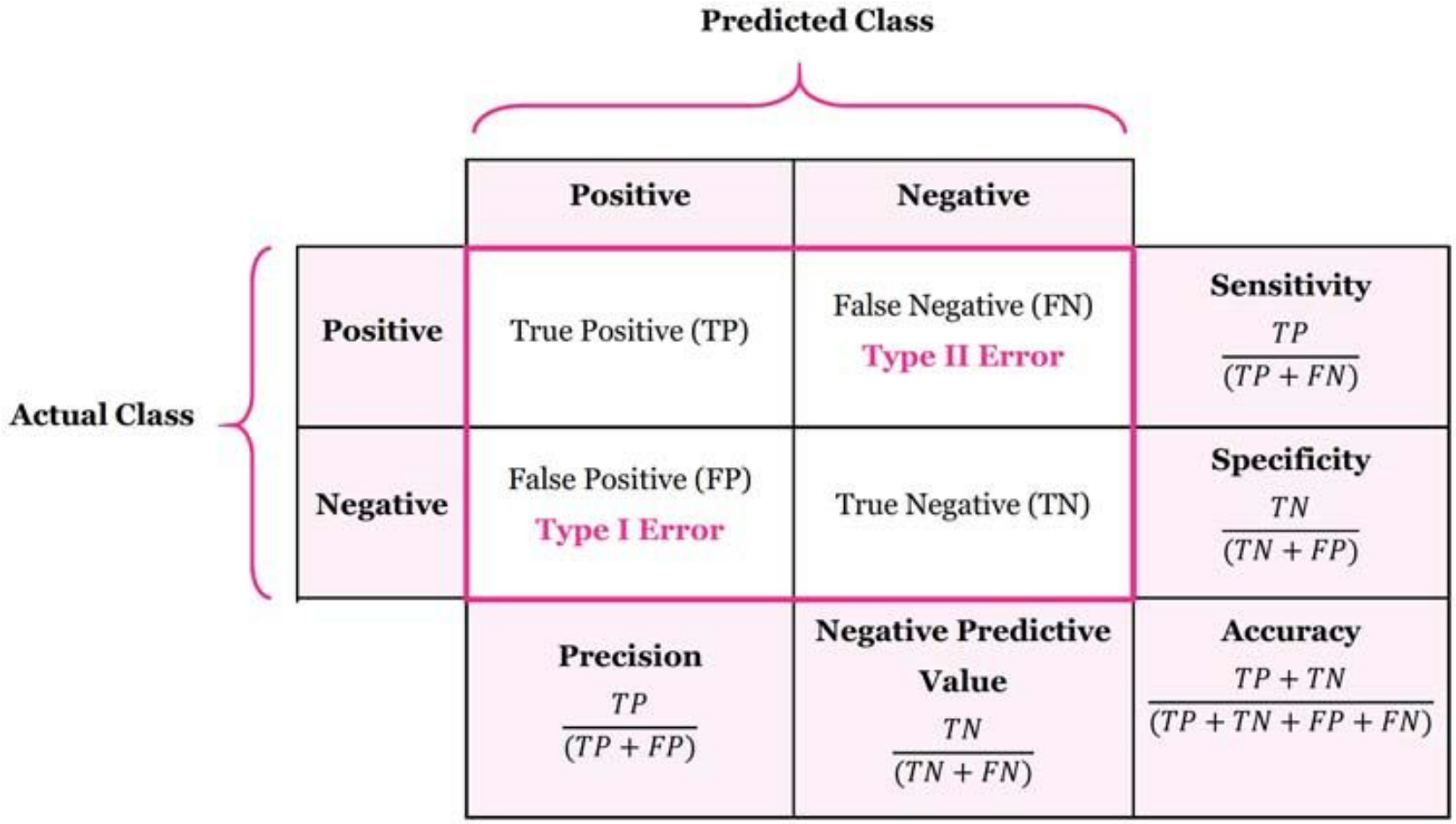
To investigate healthy patient one more time OR
to miss one deadly dangerous pathology?



Common metrics: classification

1. **Accuracy** - fraction of observations, where your model **prediction was correct**, to the **total** number of observations.
2. **Precision** (aka. positive predictive value) - a fraction of observations, where your model **prediction was positive and correct**, to the total number of **positive predictions**.
3. **Recall** (aka. sensitivity) - a fraction of observations, where your model **prediction was positive and correct**, to the total number of **positive observations**.
4. **F1-score** - the harmonic **mean between precision and recall.** $$\mathrm{F1} = 2\frac{\mathrm{precision} * \mathrm{recall}}{\mathrm{precision}+\mathrm{recall}}$$
5. **AUC-ROC**

Classification: TLDR;



Common metrics: regression

Regression. Labels are either 0 or 1: $y \in (-\infty, \infty)$. Model predictions are from 0 to 1: $\hat{y} \in (-\infty, \infty)$.

- Mean absolute error $\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
- Mean squared error $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- *Mean absolute percentage error $\mathrm{MAPE} = 100\%\frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$

,

*MAPE gives relative measurement, from least performance (MAPE = 0) to best performance (MAPE = 1) compared to MAE and MSE, which range from -inf to inf.

Importance of reliable single number evaluation metric

Classifier	Precision (p)	Recall (r)	F1-Score
A	95%	90%	92.4 %
B	98%	85%	91.0%

“You'll find that your progress will be much faster if you have a single real number evaluation metric that lets you quickly tell if the new thing you just tried is working better or worse than your last idea.” @Andrew Ng

Train-val-test split

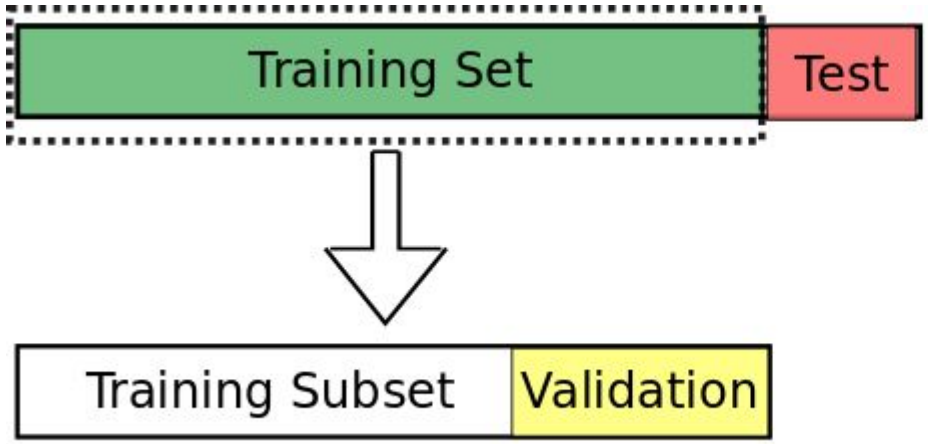
1. In real-world tasks, we want our model to generalize from training data to real-world data. Hence, the metrics computed on the training data are not representative. So, one needs a set of data, which is not used to train the model. Such a set is usually called a **“test set”**.

2. Then, while you are developing your machine learning model, you choose some hyper-parameters of your model (e.g. complexity of your model, learning rate, regularisation term, number of closest neighbors in KNN), which gives you best performance on your test set. In this case, the performance measured on the test set does not reflect the true generalization performance of your model. So, one needs the set of data, which is used to optimize hyper-parameters. Such a set is usually called a “**dev set**” (“**hold-out set**” | “**val set**”).

3. The common strategy to split your initial dataset into **train/dev/test** sets is **60/20/20%** accordingly. (i.e 60% of data is used to train model)

Summary

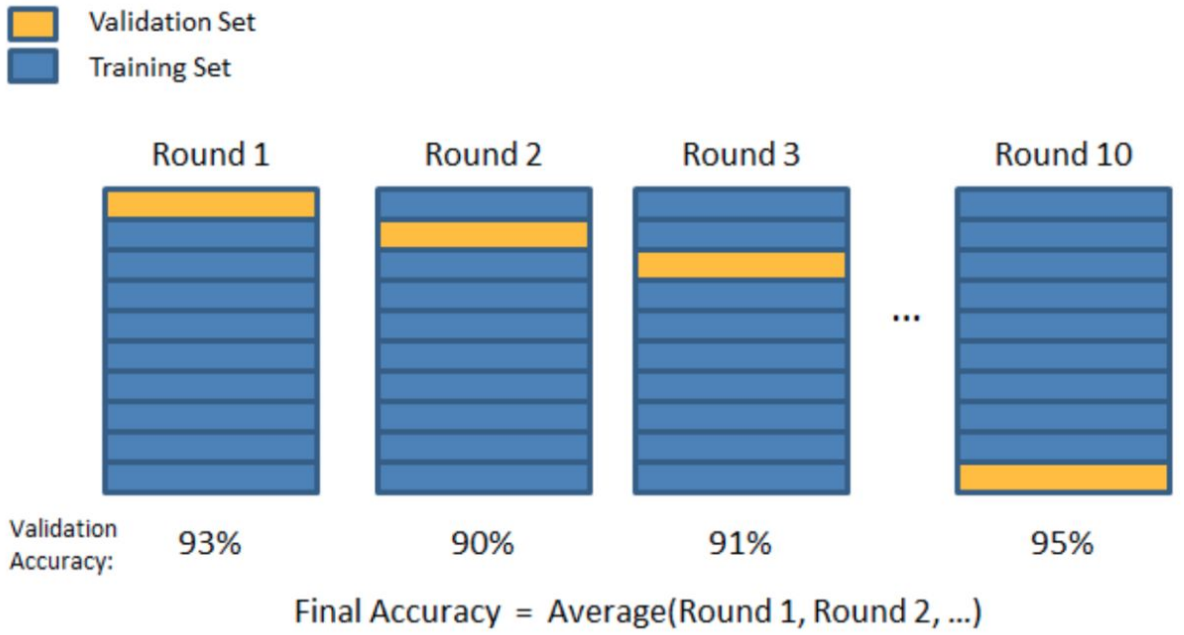
a. **Train-set**: optimize parameters
b. **Dev-set**: optimize hyper-parameters
c. **Test-set**: check model performance



K-fold Cross Validation

1. When a dataset has **no public split** for train-dev-test sets OR you are afraid that a randomly chosen **20% set of data is not representative** enough, your choice is **K-FOLD Cross-Validation**

2. This approach involves randomly dividing the set of observations into k groups (aka. folds), of approximately equal size. The first fold is treated as a validation set, and the combination of the remaining k − 1 folds is treated as a train set.

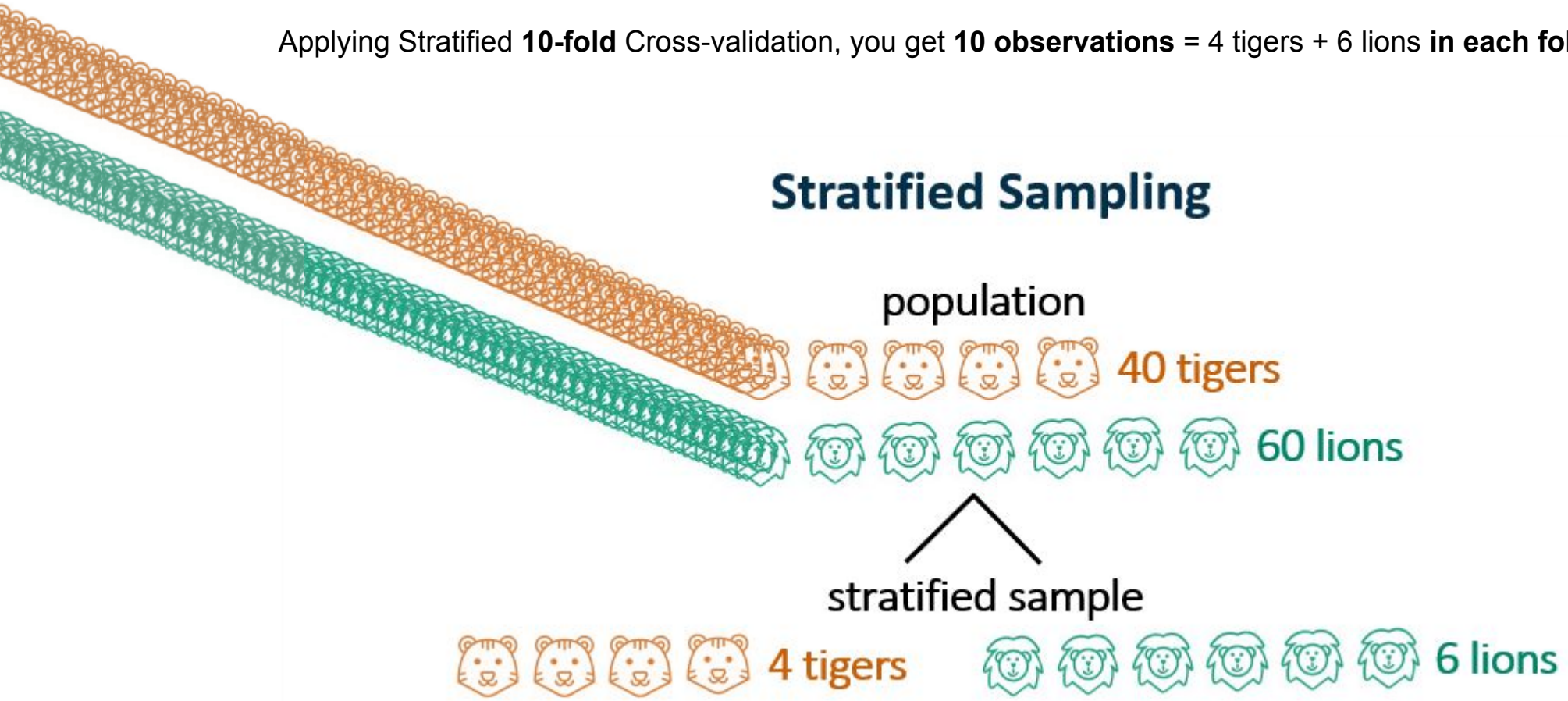


Stratified split

The splitting of data is governed by criteria such as ensuring that e**ach fold has the same proportion of observations with a given categorical value** (class).
e.g.

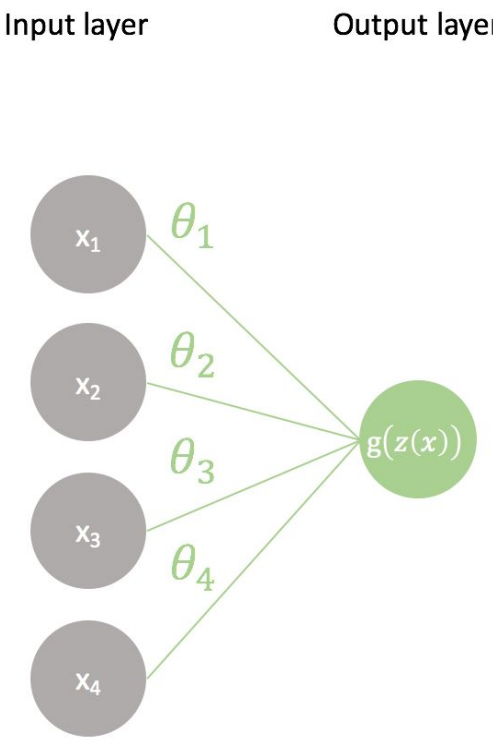
The dataset contains 100 pictures of tigers and lions, from which **40** are pictures of **tigers**, and **60** are pictures of **lions**.

Applying Stratified **10-fold** Cross-validation, you get **10 observations** = 4 tigers + 6 lions **in each fold**.



Fully connected neural network

Logistic regression. Let x be an n-dimensional feature vector: $x = (x_0, x_1, , x_i, \ldots, x_n)$



Logistic Regression

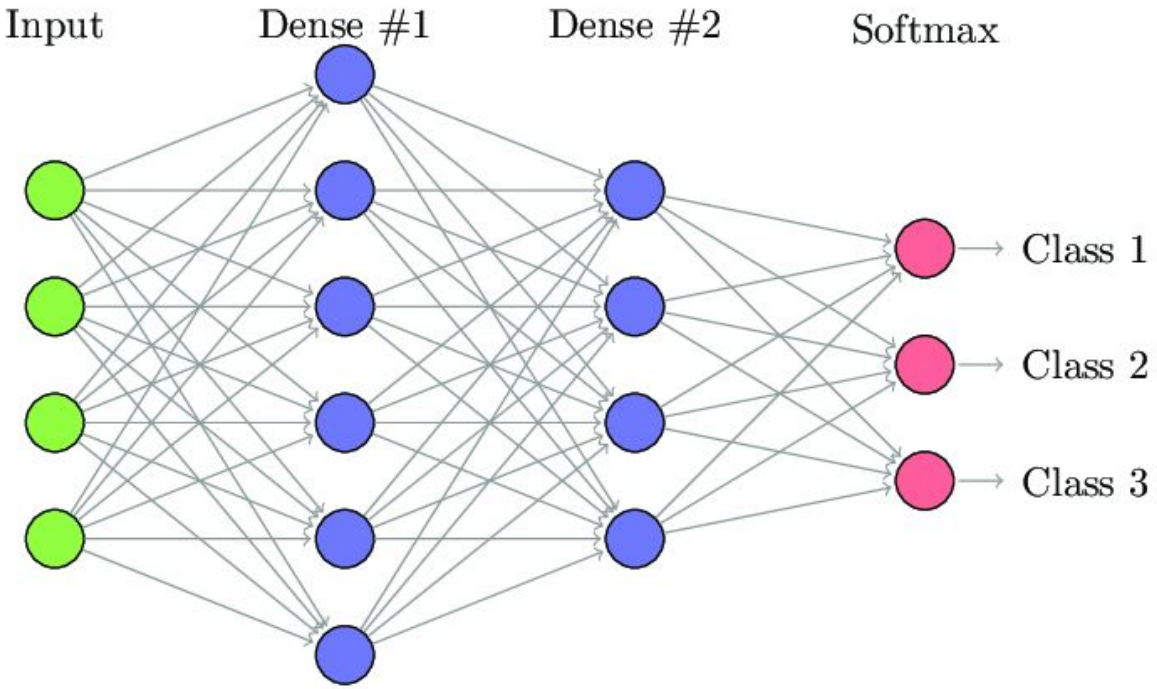
$$z(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

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

where $\theta_i(w_i)$ is a parameter to be optimized, $g(z)$ is activation function (which is the sigmoid function in case of Logistic Regression) , b is called a “bias term” and it is also a parameter to optimize.

Fully connected neural network

- One logistic regression unit is called a **neuron** in terms of Neural networks.
- The output of this unit is called **activation** (usually denoted as , for i-th neuron in a k-th hidden layer) of this neuron. $a_i^{(k)}$
- **Activation function** can be different (Tanh, ReLU, Leaky ReLU)
- **N such units** (neurons) form **one hidden layer** of a (Fully connected | Feed Forward) Neural Network.



$$b_i^{(2)} = (+1) * W_{0i}^{(2)}$$

Note: usually bias term is formulated in terms of extending the activation vector (input layer) with term equal to +1 and introducing a corresponding weight term.

Спасибо за внимание

INNOPOLIS UNIVERSITY