ERHVERVSAKADEMI
AARHUS

Machine Learning – Week 6

Sila, Sept 30th 2025.

Welcome - Todays Program.

Contents:

-Theory Part (not that much new today…):

- General problems with training using NN
- Activation functions and dying Neurons

Hands-on Part:

- Scikit learn MLPClassifier (This is a neural network)
- Measuring prediction accuracy.
- Titanic mini dataset
- Exercises - training on real (limited) dataset from Titanic. Competition!

Machine Learning – Exercises from week 5

First - we will have a look at exercises from week 4 though.

In groups of 2.
Discuss your solutions for last weeks exercises.

(Afterwards we will discuss
the exercises in class).

Exercise 6 & 7 Week 5.

- ***Iris***
- ***Blobs***

Classification with neural nets.

Machine Learning – Exercise week 5

Exercise 1, Week 7.

Starting from the Moons dataset:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons;
from pandas import DataFrame
# generate 2d classification dataset
X, y = make_moons(n_samples=1000, noise=0.1)
# scatter plot, dots colored by class value
df = DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
colors = {0:'red', 1:'blue'}
fig, ax = plt.subplots()
grouped = df.groupby('label')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y',
label=key, color=colors[key])

plt.show()
```

//See week 4 - Exercises (Scripts, Canvas)//

In groups of 2-3. Discuss & Code:

You are here asked to construct a neural net, similarly to the one you used last week – and train it. So that it can be used to predict data points, according to the Moons dataset.

a) What is the minimum number of hiddens layers needed?

b) What is the minimum number of neurons per layer needed?

c) Try out other values for the (make_moons) samplesize & noise. Also, try out other ratios for testsize.

And then redo a) and b).

Machine Learning

Intro to Neural Networks – The Second Week.

General training problems and activation functions

General training problems in NN

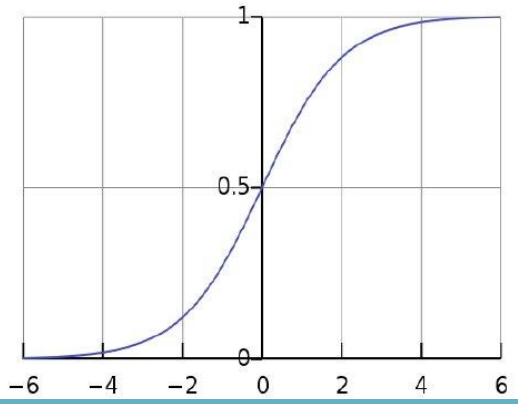
Problems for the weight-update algorithms

Vanishing gradients: This can occur when the gradient is very small - then the weights are almost never changed. This means the network will not converge on a good solution. **This is called saturation**.

Exploding gradients: Opposite problem - the gradient (rate of change) gets bigger and bigger and weights in layers are getting very big updates. This can lead to divergence of the network and also bad solutions.

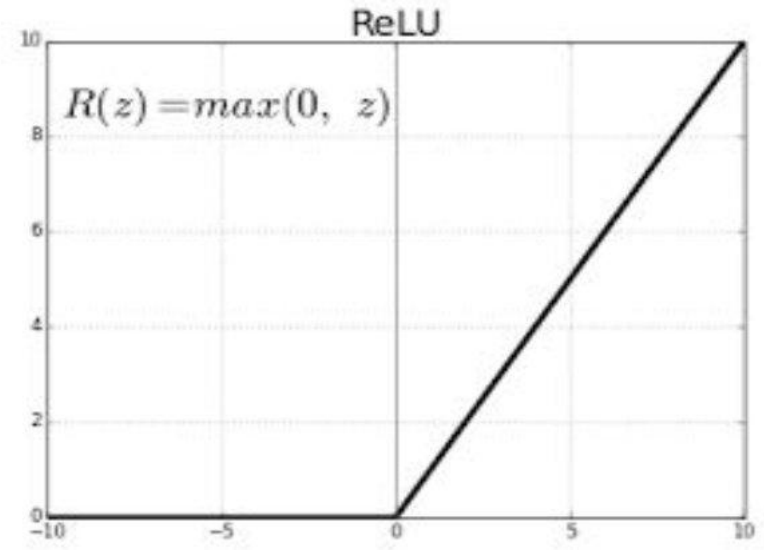
Activation functions revisited: Saturation

- Landmark paper in 2010 "Understanding the difficulty of Training Deep Feedforward Neural Networks", by Glorot and Bengio.
- They showed a problem with the Sigmoid Activation function



The problem is that if the input gets very big or very small for the neuron, then the gradient (slope) of the sigmoid function approaches 0. So this means that in the back propagation algorithm, the updates become very small. This is called saturation.

Relu - better - but still with problems.



In this case, there are no problems with large values, since the gradient (slope) is constant and does not approach zero as with the Sigmoid function.
However, there can be a problem with so-called “Dead neurons”.

A dead neuron outputs 0 almost all the time. This can happen if the weighted sum of all the inputs < 0, then ReLu outputs 0. In some cases this effect can “trickle” through the network and there can be a risk for a section of dead neurons.
It is a risk, not a certainty, though.

Newer alternatives to ReLu.

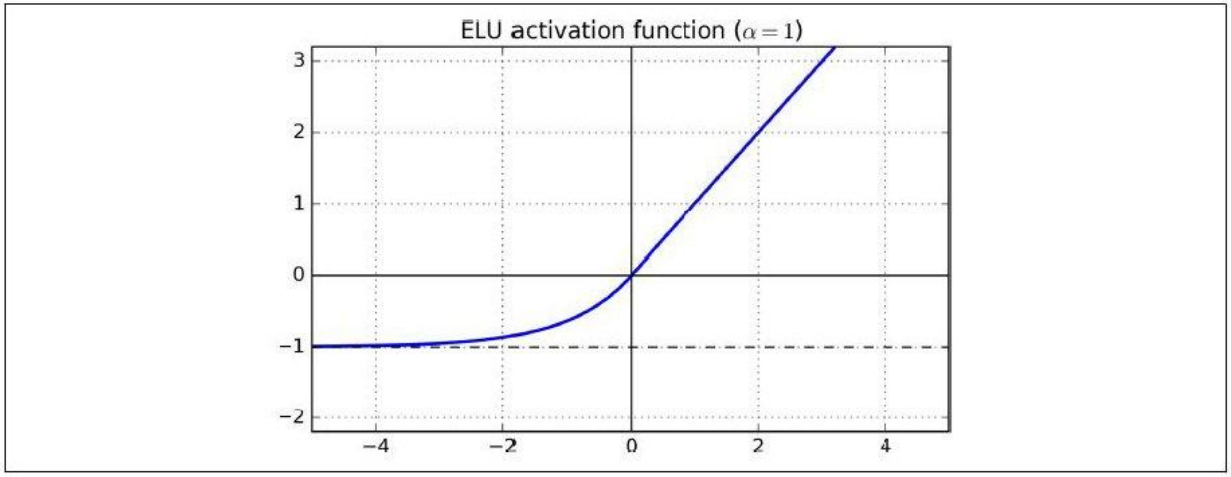
Paper in 2015
by Clevert et al:

Exponential Linear Unit (ELU) function.

Better than ReLu, but slower to compute.

Equation 11-2. ELU activation function

$$\mathrm{ELU}_{\alpha}(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$



Scikit-learn MLPClassifier

Scikit-learn workflow in general with MLPClassifier

1. Loading data
2. Data cleaning and scaling
3. Define a neural network topology
4. Train a neural network
5. Evaluate the trained model on (test) data.
6. If happy: Prediction of new values on unknown data, if not then change parameters and retrain

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

1. Loading data - from a .csv file (pandas)

```
import pandas as pd

data=pd.read_csv('titanic_train_500_age_passengerclass.csv',
sep=',',
Header=0)

# show a summmary of the the data
print(data.describe(include='all'))

#print all the data
print(data.values)
```

2. Data cleaning

```
# dropping features - columns
data.drop('PassengerId',axis=1,inplace=True)

#calculations
avg = data["Age"].mean()

#filling in missing data - you can put any value, maybe not
#best with average value?
data['Age'] = data['Age'].fillna(avg)

#splitting data
xtrain = data.head(400)
xtest = data.tail(100)
```

2. Scaling

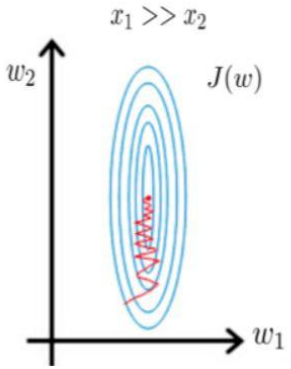
```
from sklearn.preprocessing import
StandardScaler
scaler = StandardScaler()
scaler.fit(xtrain)

xtrain = scaler.transform(xtrain)
xtest= scaler.transform(xtest)

#in this case the y data is not scaled - it is already
# between 0 and 1 (inclusive)
```

Machine Learning – Scaling, why?

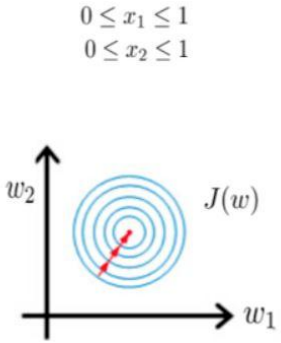
Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

Feature scaling is needed to bring every feature in the same footing without any upfront importance.
If we convert weight from grams to kg, then price becomes more important here….

Gradient descent without scaling

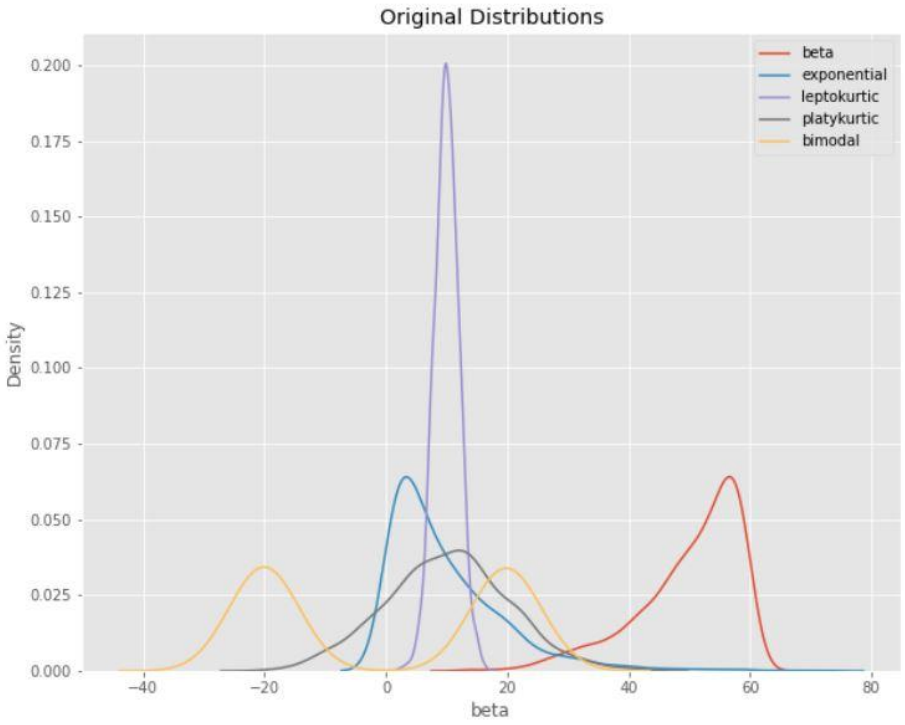


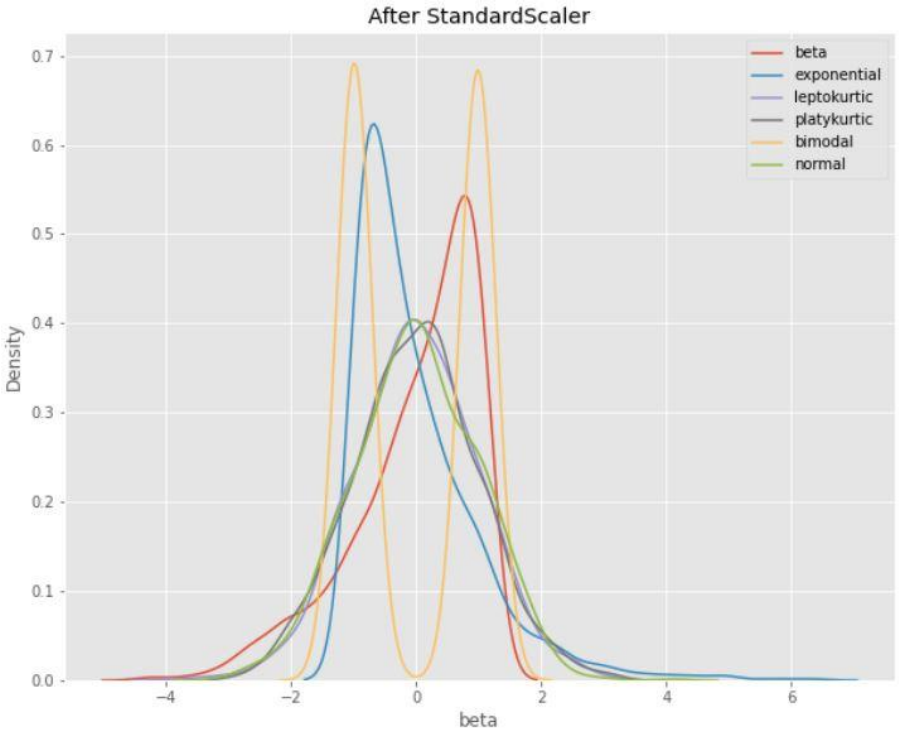
Gradient descent after scaling variables



Also, many algorithms, like gradient descent neural networks, converge faster with feature scaling than without it.

Machine Learning – Scaling. Example.





3. Defining the Neural Network

```
from sklearn.neural_network import MLPClassifier
```

Example 1:
1 hidden layer with 32 neurons

```
mlp = MLPClassifier(hidden_layer_sizes=(32),max_iter=1000,random_state=0)
```

max_iter is **equivalent to maximum number of epochs you want the model get trained on**.

Example 2:
2 hidden layers with 16 in layer 1 and 8 neurons in layer 2

```
mlp = MLPClassifier(hidden_layer_sizes=(16,8),max_iter=1000,random_state=0)
```

See: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

4. Training a Neural network

We need to have labelled data - so we are talking supervised learning

mlp.fit(xtrain,ytrain)

fit(X, y)

Fit the model to data matrix X and target(s) y.

4. Training a model - epochs and batch size

one epoch : meaning one forward pass and one backward pass of all the training Samples

batch size : The number of training examples in one forward/backward pass. So if you have 1000 samples and a batch size of 100, then 10 iterations is needed per each epoch. After 100 samples the weights are updated.

Advantage of small batch size: You use less memory and typically the network will converge faster as the weights are updated more frequently.

Disadvantage of small batch size: Typically the network will “fluctuate” more since the samples could contain outliers.

4. Training - epoch and batch size in scikit-learn

```
mlp =
MLPClassifier(hidden_layer_sizes=(32),max_iter=1000,random_s
Tate=0)

#training with different batch_size and learning rate
mlp =
MLPClassifier(hidden_layer_sizes=(32),max_iter=1000,batch_si
ze=32,learning_rate_init=0.01,random_state=0)

#default learning rate : 0.001
```

5. Evaluating the model - measuring accuracy

You can evaluate how good the model is on your **test data** (not the training data!).

```
#mlp is a trained neural network
predictions = mlp.predict(xtest)
matrix = confusion_matrix(ytest,predictions)

print(matrix)

#output is the confusion matrix
Note: We use xtest and ytest, NOT xtrain,ytrain
```

We will come back to what exactly this confusion matrix in a later slide.

6. Making predictions with a trained model

We can simply give a single row (or multiple rows) of new data to the network like this:

```
predictions = model.predict(row)
print("predicted as : "+str(predictions))
```

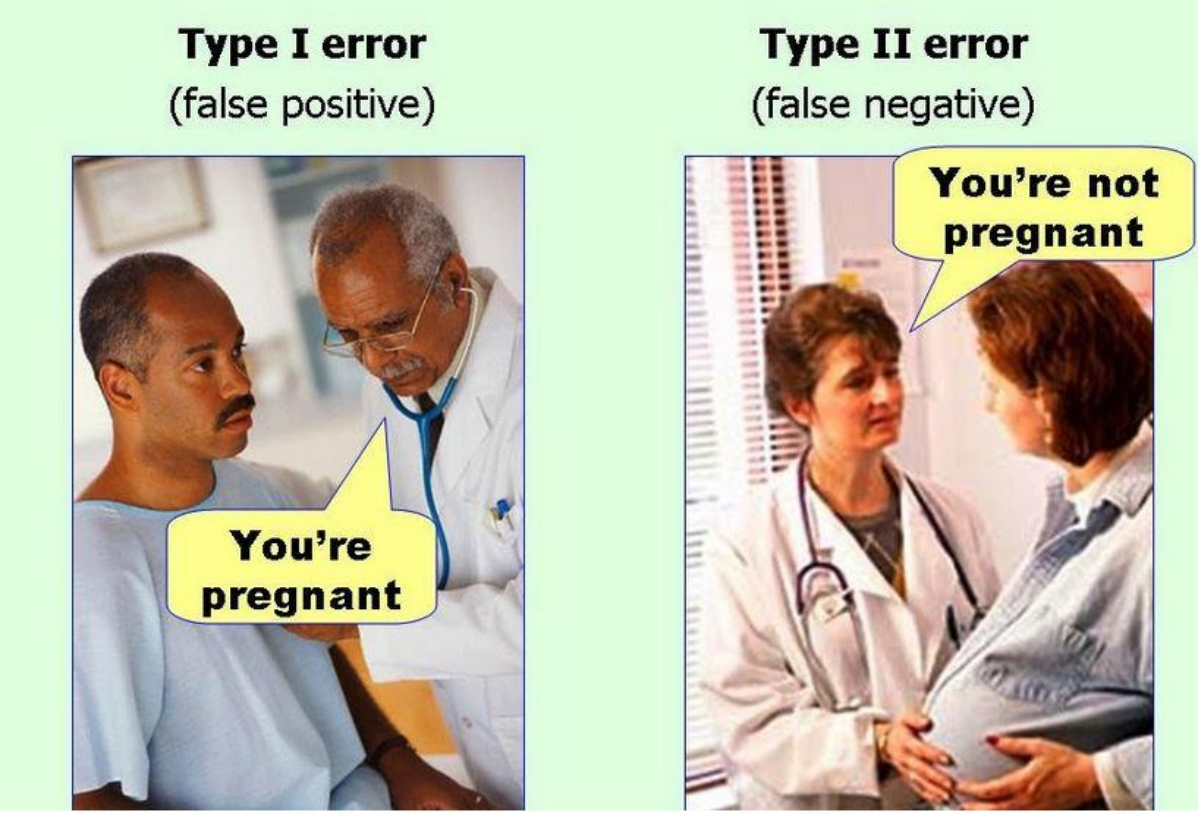
The result is then an array of the predictions (in the titanic case it will be 1 or 0).

Measuring prediction accuracy

Machine Learning – Confusion Matrix.

	Predicted class POSITIVE (spam ✉)	Predicted class NEGATIVE (normal ✉)	
Actual class POSITIVE (spam ✉)	TRUE POSITIVE (TP) ✉ ✉ **320**	FALSE NEGATIVE (FN) ✉ ✉ **43**	*Recall* $= \frac{TP}{TP + FN}$ $= \frac{320}{320 + 43} = 0.882$
Actual class NEGATIVE (normal ✉)	FALSE POSITIVE (FP) ✉ ✉ **20**	TRUE NEGATIVE (TN) ✉ ✉ **538**	
	Precision $= \frac{TP}{TP + FP}$ $= \frac{320}{320 + 20} = 0.941$		

Machine Learning – Confusion Matrix.



Machine Learning – Confusion Matrix.

$$P = \frac{TP}{TP + FP}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives} = \frac{terrorists\ correctly\ identified}{terrorists\ correctly\ identified + individuals\ incorrectly\ labeled\ as\ terroists}$$

Precision: The ratio of correct positive predictions to the total predicted positives.

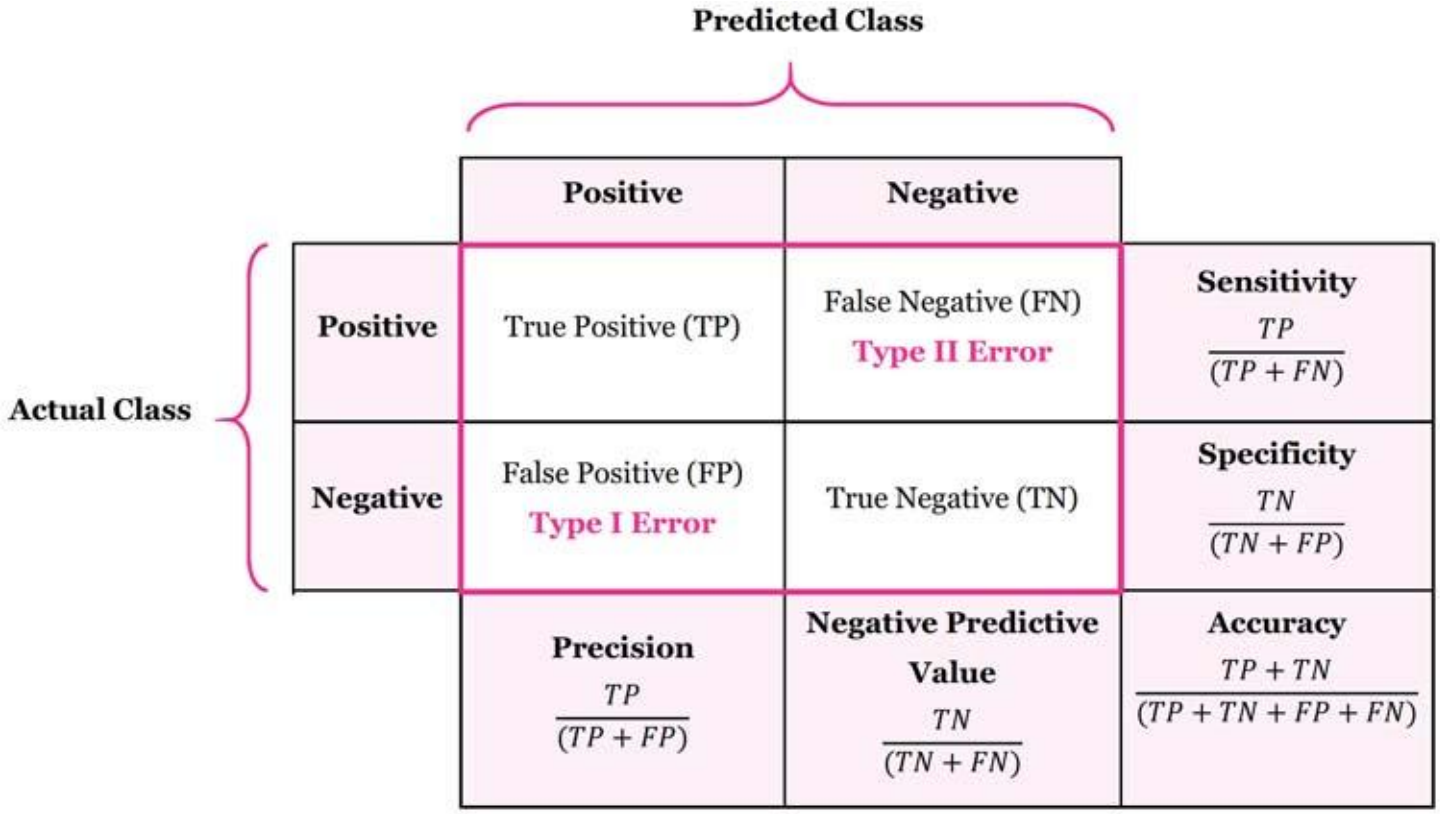
Machine Learning – Confusion Matrix.

$$R = \frac{TP}{TP + FN}$$

$$recall = \frac{true\ positives}{true\ positives\ +\ false\ negatives} = \frac{terrorists\ correctly\ identified}{terrorists\ correctly\ identified\ +\ terrorists\ incorrectly\ labeled\ as\ not\ terrorists}$$

The ratio of correct positive predictions to the total positives examples.

Machine Learning – Confusion Matrix.



Machine Learning – Intro to Neural Nets 2

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Machine Learning – Intro to Neural Nets 2

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

True Positive + False Positive = Total Predicted Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Machine Learning – Confusion matrix.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

True Positive + False Negative = Actual Positive

Machine Learning – Accuracy.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

In terms of confusion matrix it is given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Machine Learning – Exercise 2.
Discuss In groups.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

How often (in percent) is this model correct?

True Positive Rate: When it's actually yes, how often does it predict yes?

False Positive Rate: When it's actually no, how often does it predict yes?

Machine Learning – ROC curves.

Output of logistic regression	Actual class	Prediction
0,05	0	0
0,1	0	0
0,15	0	0
0,2	0	0
0,25	0	0
0,3	0	0
0,35	1	0
Threshold		
0,4	0	1
0,45	1	1
0,5	1	1
0,55	0	1
0,6	1	1
0,65	1	1
0,7	1	1
0,75	1	1
0,8	1	1
0,85	1	1
0,9	1	1
0,95	1	1

Output of logistic regression	Actual Class	Prediction
0,05	0	0
0,1	0	0
0,15	0	0
0,2	0	0
0,25	0	0
0,3	0	0
0,35	1	0
0,4	0	0
0,45	1	0
0,5	1	0
Threshold		
0,55	0	1
0,6	1	1
0,65	1	1
0,7	1	1
0,75	1	1
0,8	1	1
0,85	1	1
0,9	1	1
0,95	1	1

Machine Learning – ROC curves.

ROC plots 2 parameters:

Output of logistic regression	Actual class	Prediction
0,05	0	0
0,1	0	0
0,15	0	0
0,2	0	0
0,25	0	0
0,3	0	0
0,35	1	0
Threshold		
0,4	0	1
0,45	1	1
0,5	1	1
0,55	0	1
0,6	1	1
0,65	1	1
0,7	1	1
0,75	1	1
0,8	1	1
0,85	1	1
0,9	1	1
0,95	1	1

Output of logistic regression	Actual Class	Prediction
0,05	0	0
0,1	0	0
0,15	0	0
0,2	0	0
0,25	0	0
0,3	0	0
0,35	1	0
0,4	0	0
0,45	1	0
0,5	1	0
Threshold		
0,55	0	1
0,6	1	1
0,65	1	1
0,7	1	1
0,75	1	1
0,8	1	1
0,85	1	1
0,9	1	1
0,95	1	1

- **True positive rate** (Recall)

$$TPR = \frac{TP}{TP + FN}$$

- **False Positive rate**

$$FPR = \frac{FP}{FP + TN}$$

Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

Machine Learning – ROC curves.

ROC plots 2 parameters:

- **True positive rate** (Recall)

$$TPR = \frac{TP}{TP + FN}$$

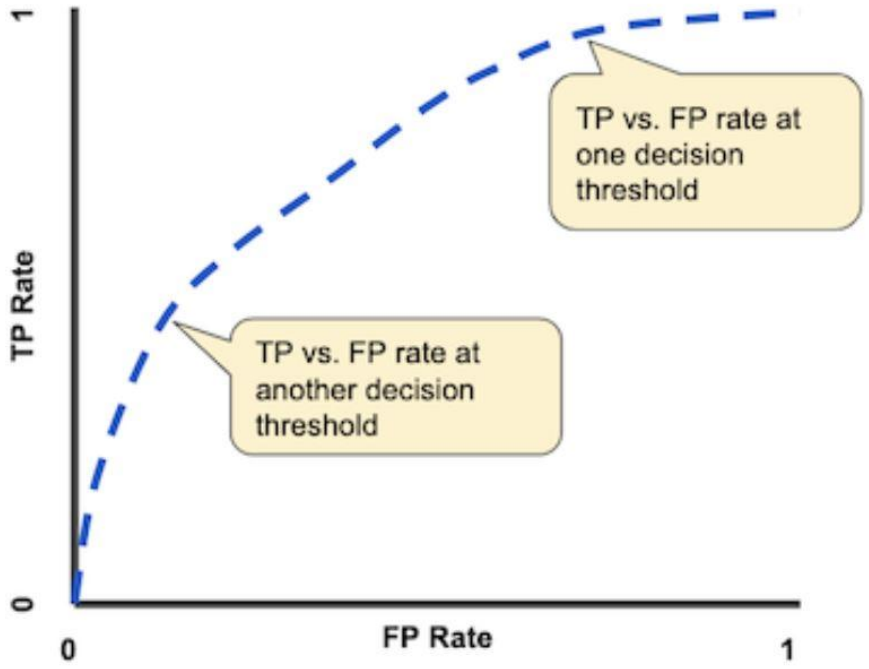
- **False Positive rate**

$$FPR = \frac{FP}{FP + TN}$$

A ROC curve (receiver operating characteristic curve) graph shows the performance of a classification model at all classification thresholds.

Machine Learning – ROC curves.

ROC plots 2 parameters:



- **True positive rate** (Recall)

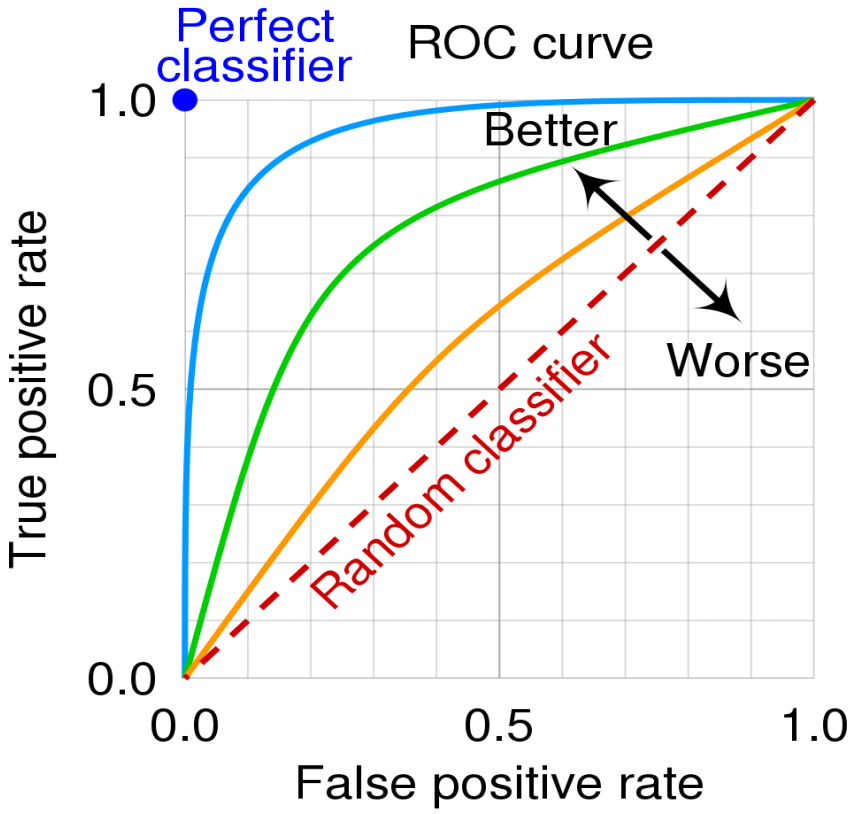
$$TPR = \frac{TP}{TP + FN}$$

- **False Positive rate**

$$FPR = \frac{FP}{FP + TN}$$

Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

Machine Learning – Intro to Neural Nets 2. ROC.



$$TruePositiveRate(Recall) = \frac{TP}{TP+FN} = \frac{0}{0+5} = 0$$

$$FalsePositiveRate = \frac{FP}{FP+TN} = \frac{0}{0+5} = 0$$

Machine Learning – Intro to Neural Nets 2. ROC.

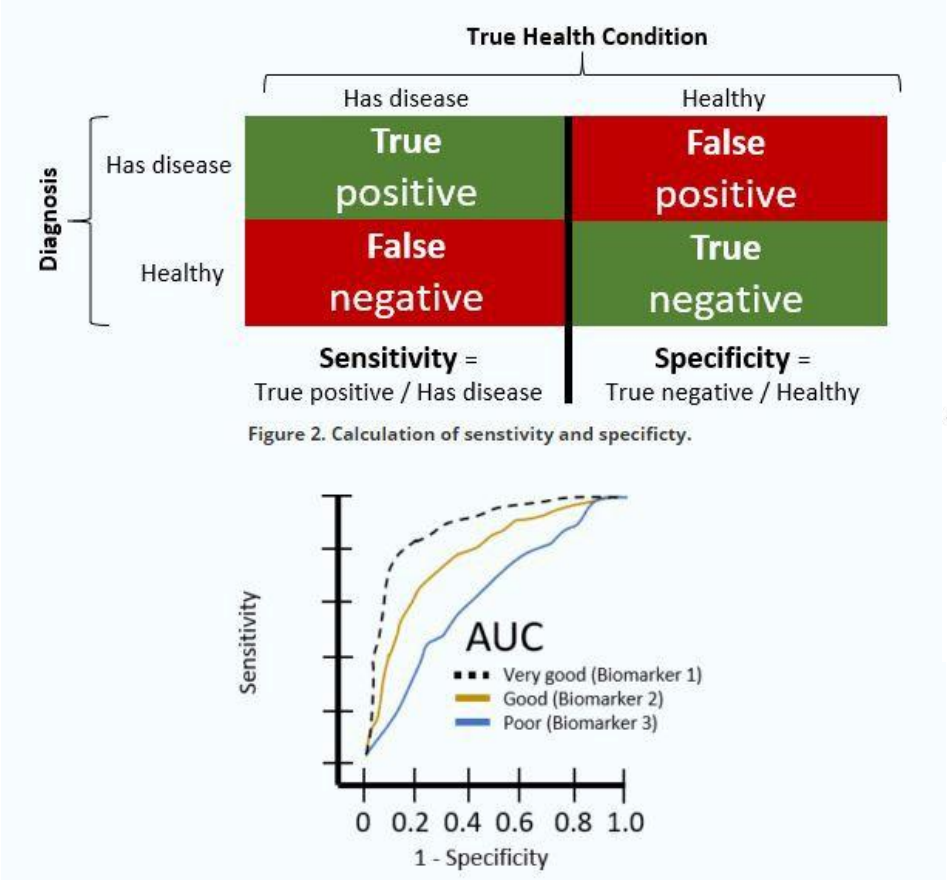
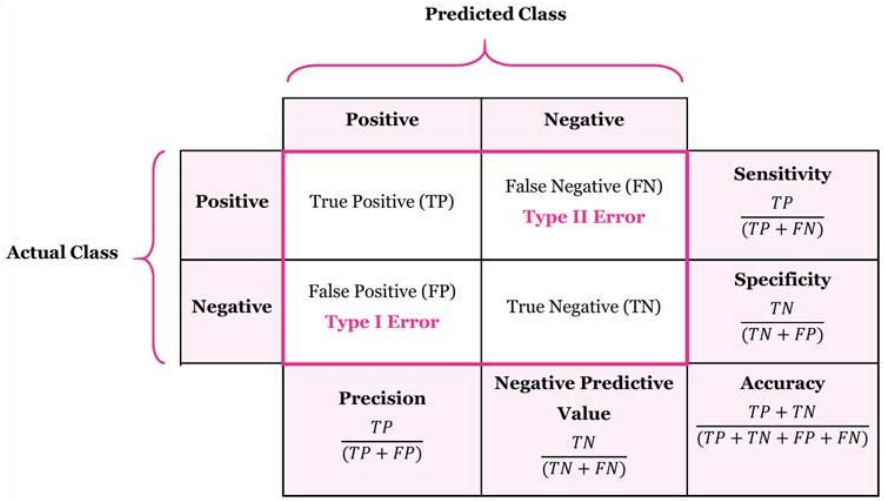


Figure 2. Calculation of senstivity and specificty.



Machine Learning – Exercise 3.
Discuss In groups.

Let us say you have a binary predictor where the TN = 12, TP = 23, FP = 5, FN = 10. What percentage of the total predictions did this predictor classify correctly?

You will need to understand this in order to calculate the prediction accuracy for the mini-titanic exercise (later today) in order to understand what is going on…

Don't rush it, but take your time to fully understand what is going on here (HINT: The percentage should be a nice round number…..!)

Machine Learning – Intro to Neural Nets 2

F1 Score

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

F1 is an overall measure of a model's accuracy that combines precision and recall, in that weird way that addition and multiplication just mix two ingredients to make a separate dish altogether. That is, a good F1 score means that you have low false positives and low false negatives, so you're correctly identifying real threats and you are not disturbed by false alarms. An F1 score is considered perfect when it's `1`, while the model is a total failure when it's `0`.

For example precision = 1 and recall = 0. It looks that in this case precision is ignored, and the F1 score remain equal to 0.
It behaves like that in all cases. If one of the parameters is small, the second one no longer matters.
F1 score emphasizes the lowest value.

Machine Learning – Exercise 4.
In groups. 2 -3.

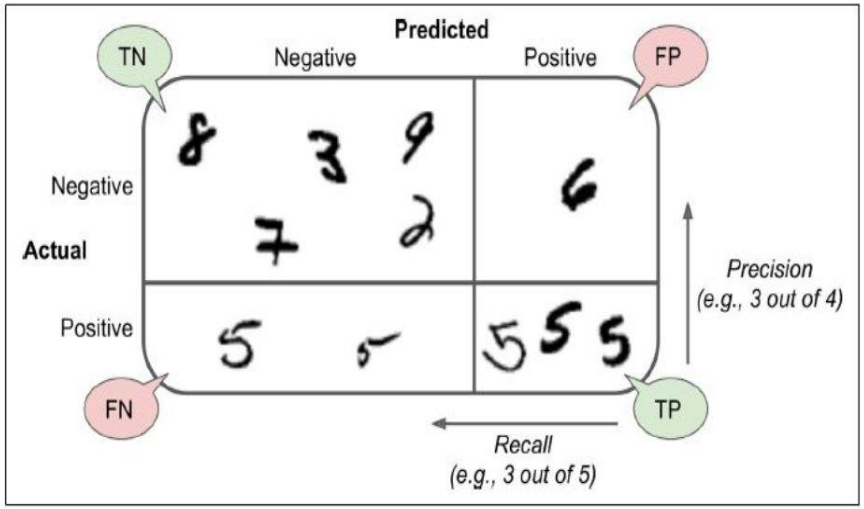
Lets imagine that we have a zip code reader – and we want to identify a number as either a 5 or not a 5.

We make ***a very simple algorithm that just always predicts 5*** – no matter what the number is (0,1,2..9).

Be very careful…. Insert in confusion matrix….

What is recall …is that a good value…?!
What is precision… is that a good value?

Confusion Matrix - not confusing :-)



This classifier tries to identify a number as either a 5 or not a 5.

Recall = 0.60
(ratio of positives correctly predicted)

Precision = 0.75
(accuracy of positives predicted)

(page 91 in the book has defs.)

If we just always predict 5, we get 100% recall, but very low precision.
With a perfect prediction, FN = FP = 0.

Confusing Matrix - in Python

```
#mlp is a trained neural network
predictions = mlp.predict(xtest)
matrix = confusion_matrix(ytest,predictions)
print(matrix)

#output is the confusion matrix
>>[[61  3]
>>[26 10]]
tn, fp, fn, tp = matrix.ravel()
print("(TN,FP,FN,TP)",(tn, fp, fn, tp))
#the individual values
>>(TN,FP,FN,TP) (61, 3, 26, 10)
```

Precision/recall in python (both on pos. and neg.)

```
#mlp is a trained neural network
predictions = mlp.predict(xtest)
print(classification_report(ytest,predictions))
```

```
>> precision recall f1-score support
      0 0.70 0.95 0.81 64
      1 0.77 0.28 0.41 36

avg / total 0.73 0.71 0.66 100
```

Titanic mini data set

Machine Learning – Exercises.

Titanic set

500 data points (rows), use 400 for training and the last 100 for test.

	A	B	C	D
1	PassengerId	Survived	Pclass	Age
2	1	0	3	22
3	2	1	1	38
4	3	1	3	26
5	4	1	1	35
6	5	0	3	35
7	6	0	3	
8	7	0	1	54
9	8	0	3	2
10	9	1	3	27
11	10	1	2	14

Have a look at the exercise 5 for more info.

Finish up exercise 1..4 first though!

Titanic set - lazy benchmark

A lazy unoptimized solution predicts 71 % correct using a NN. Surely you can do better!

Total: 100 in test set	**predicted dead**	**predicted survived**
Actual dead	TN: 61	FP: 3
Actual survived	FN: 26 (bad!)	TP: 10

So, here we get 61+10 = 71 out of 100 correct. So 71% correct.

Exercise 5…

Titanic…

Homework

Week 6.
Topics : Neural Nets.

Litterature:

[Hands-On Machine Learning]

[Hands-On Machine Learning]
Chapter 3, p. 108 – 128 (Sections: Confusion Matrix,
ROC curves - performance of our predictors)
Chapter 11, p. 357 (the beginning of the chapter) until p. 366
(problems with trainingusing Neural Nets)

Exercises.
For week 6.