Neural Networks in Action - Tarea 1

Alexandre Bergel
http://bergel.eu
23/09/2020

Outline

1. Performance of a neural network
2. Tarea 1

Outline

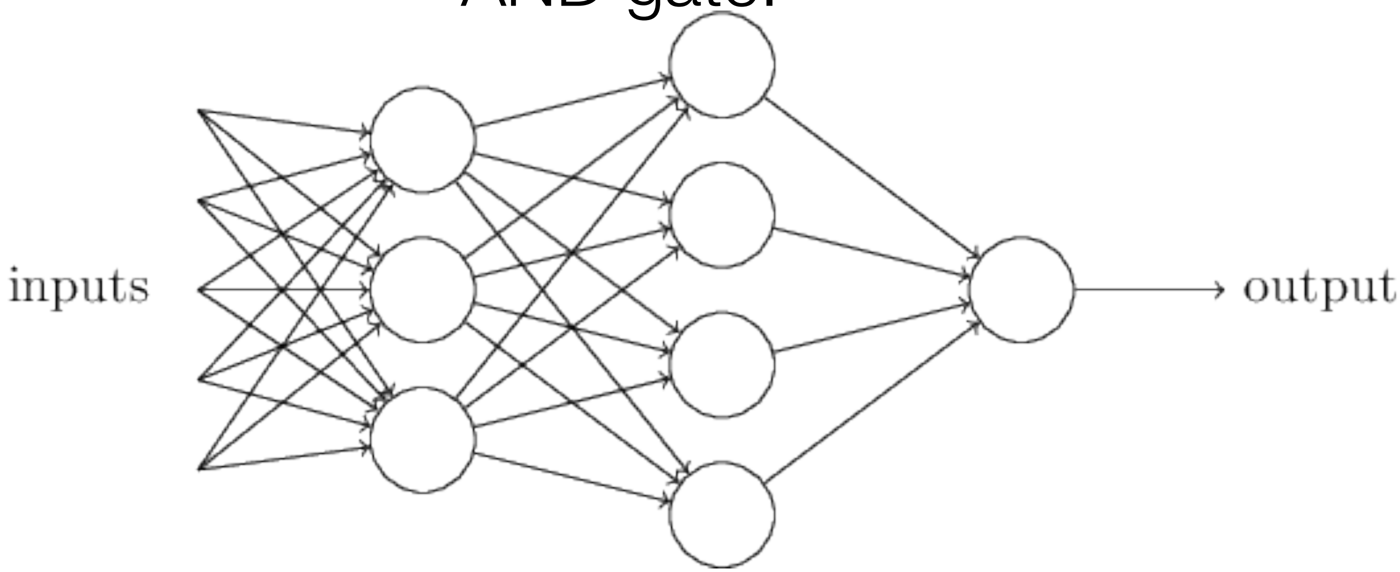
1. **Performance of a neural network**
2. Tarea 1

Epoch

1 *epoch* = one forward pass and one backward pass of *all* the training examples

AND gate:

0 0
0 1
1 0
1 1



0
0
0
1

Training of 4 the combinations to the network = 1 epoch

Performance of a Neural Network

It is important to have indicators on *how your network is learning and performing*

Typically, such metrics are computed per *epoch.*

Many metrics are available, depending on what your neural network is supposed to do.

Typically, you need to *compute* some metric values *after each epoch*

Labeled dataset

Assume you wish to train a neural network-based model over a labeled dataset

A labeled dataset is a dataset in which samples are tagged with one or more labels

Example: the Iris dataset

https://archive.ics.uci.edu/ml/datasets/iris

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	2767455

Source:

Creator:

R.A. Fisher

Donor:

Michael Marshall (MARSHALL%PLU '@' io.arc.nasa.gov)

Data Set Information:

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the fi
latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant.

This is an exceedingly simple domain.

This data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick '@' espeedaz.net). T

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	2767455

Source:

Creator:

R.A. Fisher

Donor:

Michael Marshall (MARSHALL%PLU '@' io.arc.nasa.gov)

Data Set Information:

This is perhaps the best known database to be fo ... ic in the fi
latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant.

This is an exceedingly simple domain.

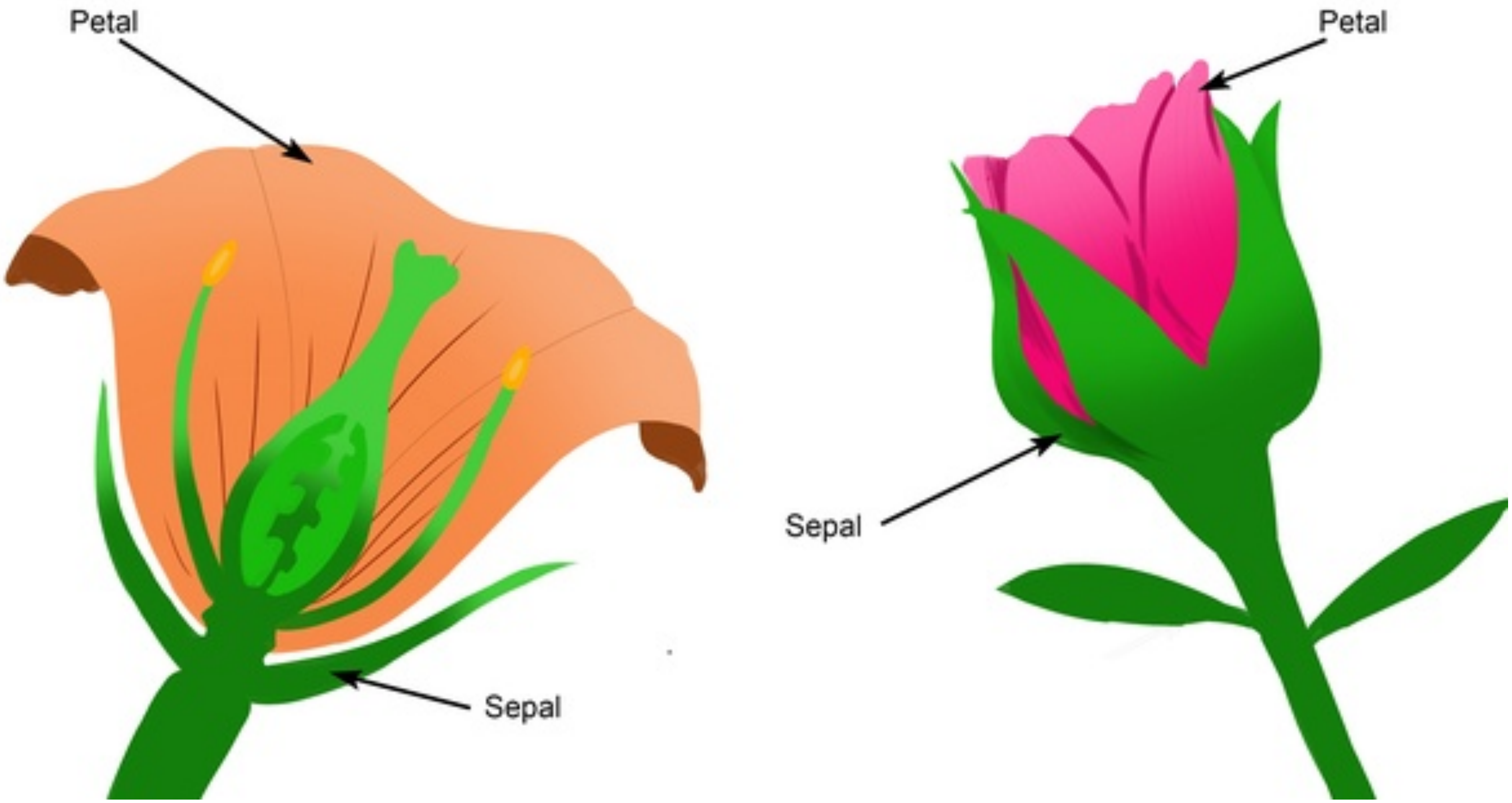
This data differs from the data presented in Fisher ... az.net). T

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Attribute Information:

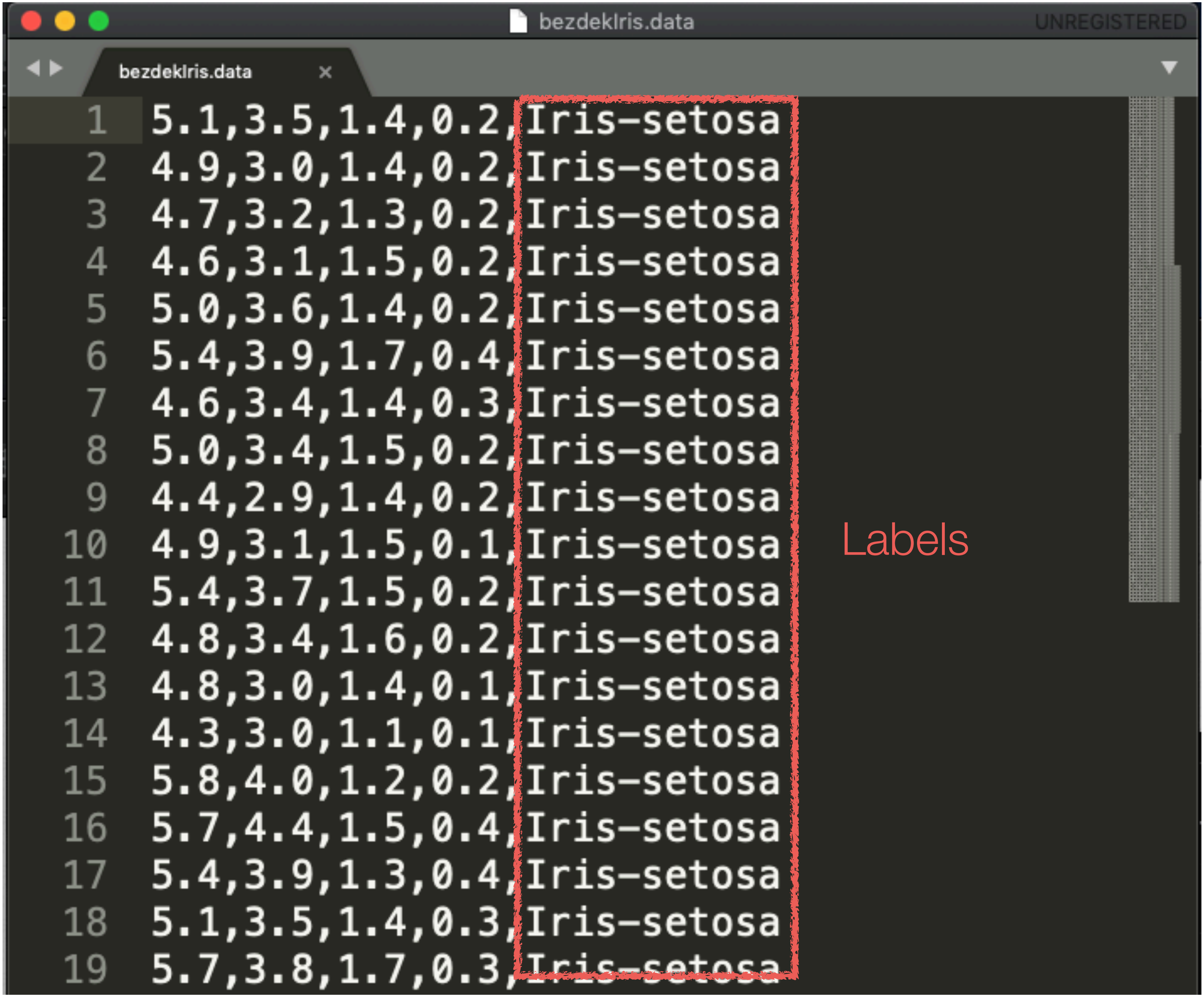
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica


Petal
Sepal
Petal
Sepal

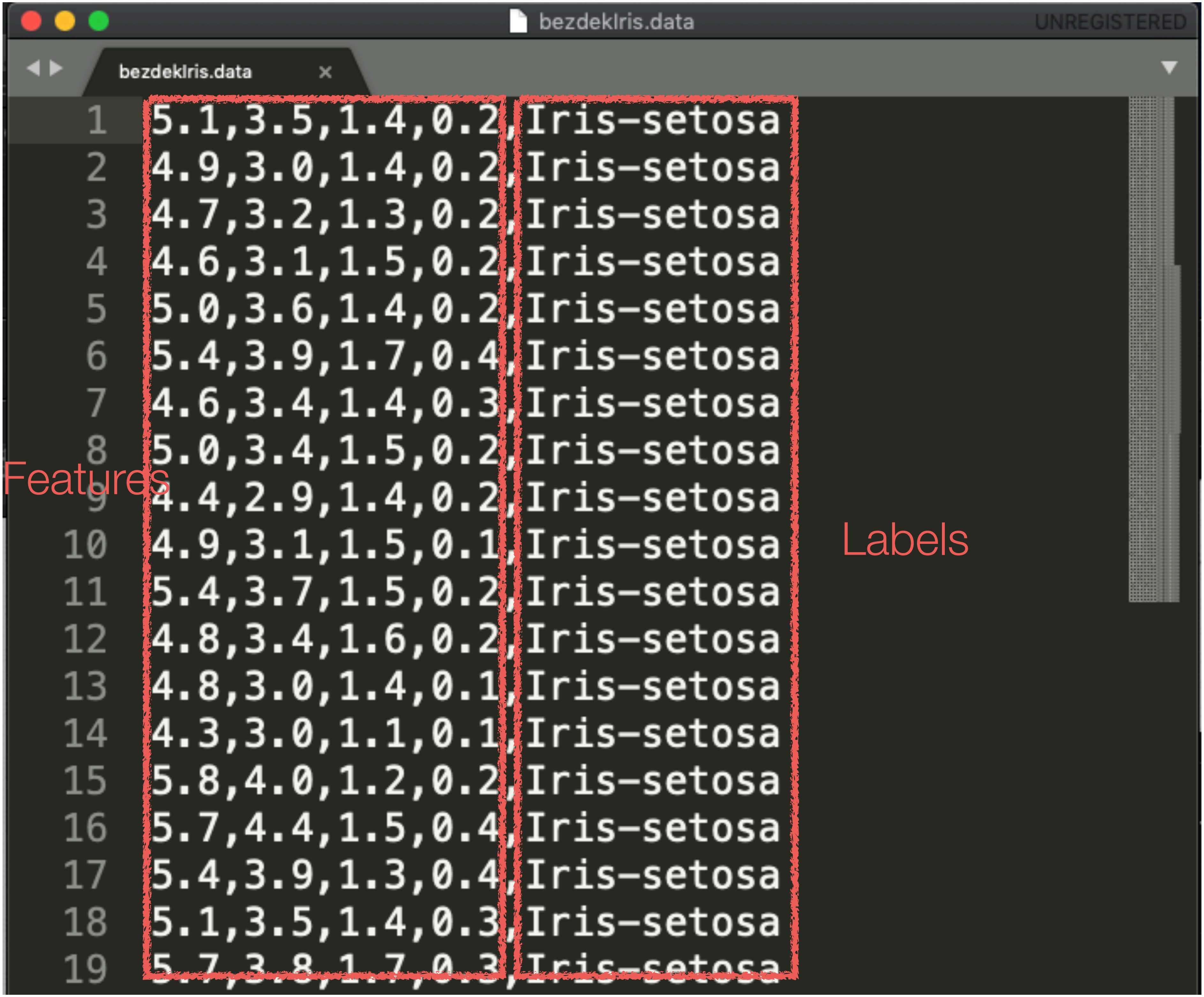
bezdekIris.data UNREGISTERED

bezdekIris.data ×

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
```




```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
```


bezdekIris.data
UNREGISTERED
bezdekIris.data
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
Features
Labels

Training a network

The dataset is said to be labeled because it contains labels

The Iris dataset contains three labels: `Iris-setosa`, `Iris-versicolor`, `Iris-virginica`

We are here considering using a neural network to *solve a classification task*

If you give a flower characteristic, e.g., 5.7,3.0,4.2,1.2

Then you wish the network to find the right label, which is `Iris-versicolor`. We call this a *test*.

Training a network

Before testing a neural network, you *need to train* it

An important aspect of the training and testing, is to *not test a network with the very same data it has learn*

Else, a simple dictionary is enough :-)

Dividing your dataset

From the original dataset, you need to extract a portion to train your network, and another portion to test it

A simple procedure (which is enough for Tarea 1), is to take 80% of the dataset for training and 20% for testing

We call this procedure *train/test split*

Dividing your dataset

Cross-validation is a statistical method used to estimate performance of a machine learning models (and not only neural network)

Useful to test a model in *presence of unseen data*

k-Fold Cross-Validation is a resampling procedure with a unique parameter, k

It is a *very popular method* because it is simple to understand and result in less bias than in the train/test split

k-Fold Cross-Validation

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Train a model on the training set
    4. evaluate the model on the test set
    5. Remember the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores (e.g., average or mean of the model score). Could also be accompanied with standard deviation

k-Fold Cross-Validation example

Example with 6 (of the 150) shuffled samples of the Iris dataset
We use k = 3

```
5.7,2.9,4.2,1.3,Iris-versicolor
7.3,2.9,6.3,1.8,Iris-virginica
4.6,3.4,1.4,0.3,Iris-setosa
6.7,2.5,5.8,1.8,Iris-virginica
6.2,2.9,4.3,1.3,Iris-versicolor
5.0,3.4,1.5,0.2,Iris-setosa
```

k-Fold Cross-Validation example

Example with 6 (of the 150) shuffled samples of the Iris dataset
We use k = 3

5.7,2.9,4.2,1.3,Iris-versicolor 7.3,2.9,6.3,1.8,Iris-virginica	Fold 1
4.6,3.4,1.4,0.3,Iris-setosa 6.7,2.5,5.8,1.8,Iris-virginica	Fold 2
6.2,2.9,4.3,1.3,Iris-versicolor 5.0,3.4,1.5,0.2,Iris-setosa	Fold 3

Step 2 - Split the dataset into k groups (called folds)

k-Fold Cross-Validation example

Example with 6 (of the 150) shuffled samples of the Iris dataset

We use k = 3

5.7,2.9,4.2,1.3,Iris-versicolor 7.3,2.9,6.3,1.8,Iris-virginica	Fold 1
4.6,3.4,1.4,0.3,Iris-setosa 6.7,2.5,5.8,1.8,Iris-virginica	Fold 2
6.2,2.9,4.3,1.3,Iris-versicolor 5.0,3.4,1.5,0.2,Iris-setosa	Fold 3

Step 3 - For each Fold X, you need:

1 - create a new neural network

2 - train it with the folds Y, $Y \neq X$

3 - evaluate the model on Fold X

4 - keep the score (e.g., precision and recall)

k-Fold Cross-Validation example

Example with 6 (of the 150) shuffled samples of the Iris dataset
We use k = 3

```
5.7,2.9,4.2,1.3,Iris-versicolor
7.3,2.9,6.3,1.8,Iris-virginica
```
Fold 1

```
4.6,3.4,1.4,0.3,Iris-setosa
6.7,2.5,5.8,1.8,Iris-virginica
```
Fold 2

```
6.2,2.9,4.3,1.3,Iris-versicolor
5.0,3.4,1.5,0.2,Iris-setosa
```
Fold 3

Step 4 - Provide a summary of the result
e.g., average, median, std precision, std error

k-Fold Cross-Validation

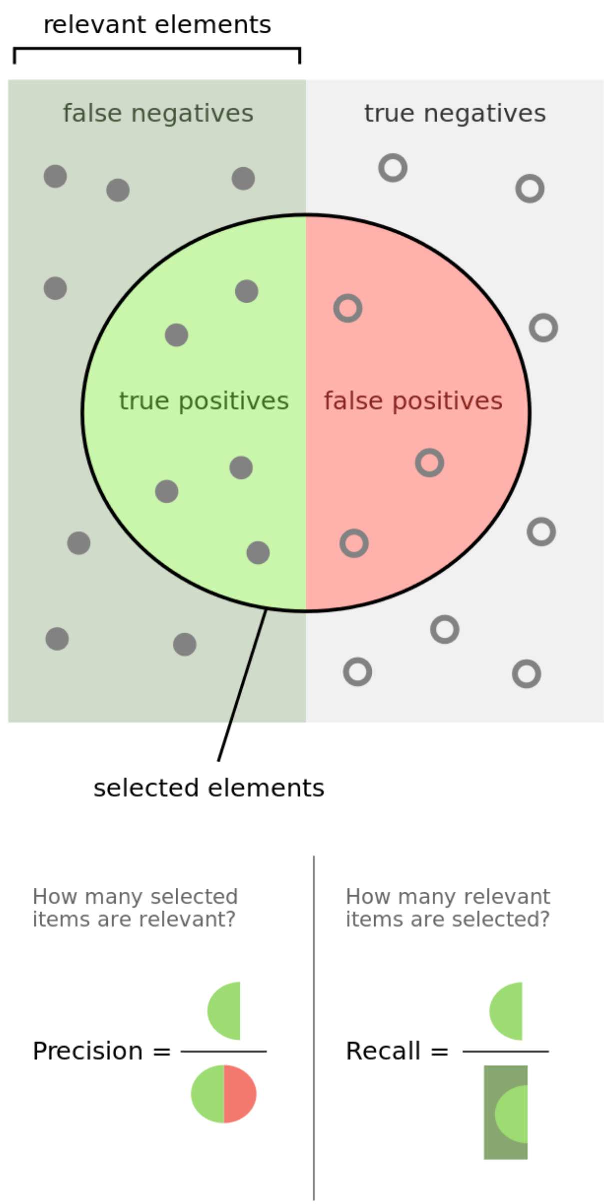
Scikit-learn offer the class `KFold()`

You can use it, if you wish, for your tarea / project

Gentle introduction to k-fold cross-validation:

https://machinelearningmastery.com/k-fold-cross-validation/

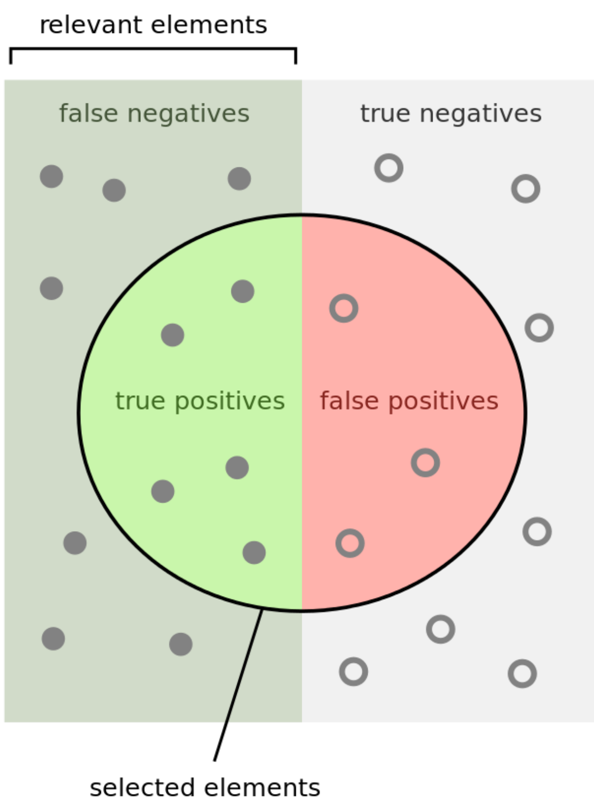
Precision and Recall

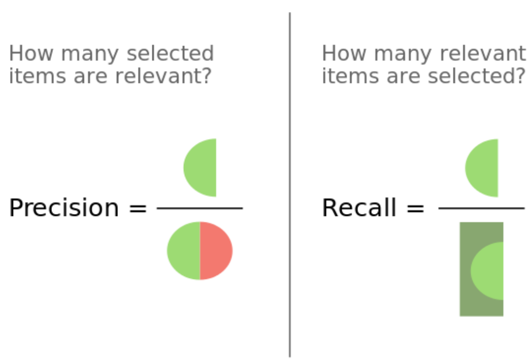


Precision is easy to compute. It is simply the ratio between the correct guesses and the number of guesses

Recall is the relation between the correct guesses and all the possible good solutions

Precision and Recall

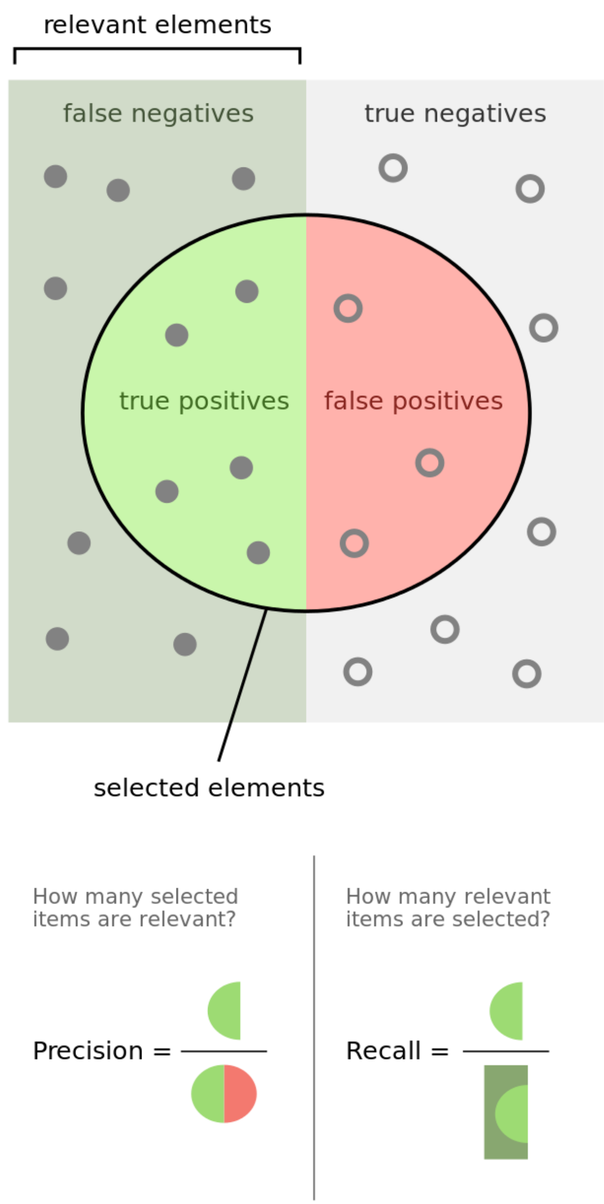




Other metrics are available, such as *F1-score*, which is a combination of precision and recall

$$F_1 = 2\frac{precision\,.\,recall}{precision + recall}$$

Precision and Recall



Precision and recall are easy to compute in presence of a *binary-class classification problem*

Work well for the blue vs red dots, but not for the iris dataset

Precision and Recall for multi-class classification problem

The confusion matrix (also called error matrix) is useful to visualize a performance of an algorithm, typically a supervised learning one

Example of a confusion matrix:

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

Precision and Recall for multi-class classification problem

The confusion matrix (also called error matrix) is useful to visualize a performance of an algorithm, typically a supervised learning one

Example of a confusion matrix:

Labels from the dataset

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

Precision and Recall for multi-class classification problem

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

- The above table assumes that you have 3 possible output labels: A, B & C.
- The diagonals contain the true positives for each label (= TP_X).
- The sum of a column would be total number of instances that should have label X
- The sum of a row would be total number of instances predicted as a particular label X
- Given all of this the precision of a label x is computed as:
- `TP_X/(TotalPredicted_X)`
- The recall of a label x is computed as:
- `TP_X/(TotalGoldLabel_X)`

Neural-network only process numbers

```
5.7,2.9,4.2,1.3,Iris-versicolor
7.3,2.9,6.3,1.8,Iris-virginica
4.6,3.4,1.4,0.3,Iris-setosa
6.7,2.5,5.8,1.8,Iris-virginica
6.2,2.9,4.3,1.3,Iris-versicolor
5.0,3.4,1.5,0.2,Iris-setosa
```

These tags are not numbers

We therefore need to transform these textual tags into numbers

One-hot encoding

```
5.7,2.9,4.2,1.3,Iris-versicolor
7.3,2.9,6.3,1.8,Iris-virginica
4.6,3.4,1.4,0.3,Iris-setosa
6.7,2.5,5.8,1.8,Iris-virginica
6.2,2.9,4.3,1.3,Iris-versicolor
5.0,3.4,1.5,0.2,Iris-setosa
```

One hot encoding is a very simple process for which categorial variables (e.g., flower name) is *converted into numerical values*

One-hot encoding

```
5.7,2.9,4.2,1.3,Iris-versicolor
7.3,2.9,6.3,1.8,Iris-virginica
4.6,3.4,1.4,0.3,Iris-setosa
6.7,2.5,5.8,1.8,Iris-virginica
6.2,2.9,4.3,1.3,Iris-versicolor
5.0,3.4,1.5,0.2,Iris-setosa
```

One hot encoding is a very simple process for which categorial variables (e.g., flower name) is *converted into numerical values*

Here is a simple recipe:

- We have N labels
- Each tag is encoded into N numerical values
- Each numerical value is either 0 or 1

One hot encoding example

```
Iris-versicolor = [1,0,0]
 Iris-virginica = [0,1,0]
    Iris-setosa = [0,0,1]
```

As a consequence, a neural network to properly classify Iris needs to have 4 inputs (each dataset row has 4 features) and 3 outputs (because of the one-hot encoding)

Determining a one hot encoding is very simple. If you have an ordered collection `{versicolor, virginica, setosa}`, then encoding label L is a vector of `[0, 0, 0]` with 1 at the index of L in the collection

Measuring error

Previously we discuss about good and bad classification

A complementary metric is the *mean squared error* (MSE)

MSE is a number representing the error made by an algorithm

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

n = number of examples
Y_i = prediction of the network
^Y_i = gold results (labels contained in the dataset)

Computing the error

In the file NeuralNetwork.py

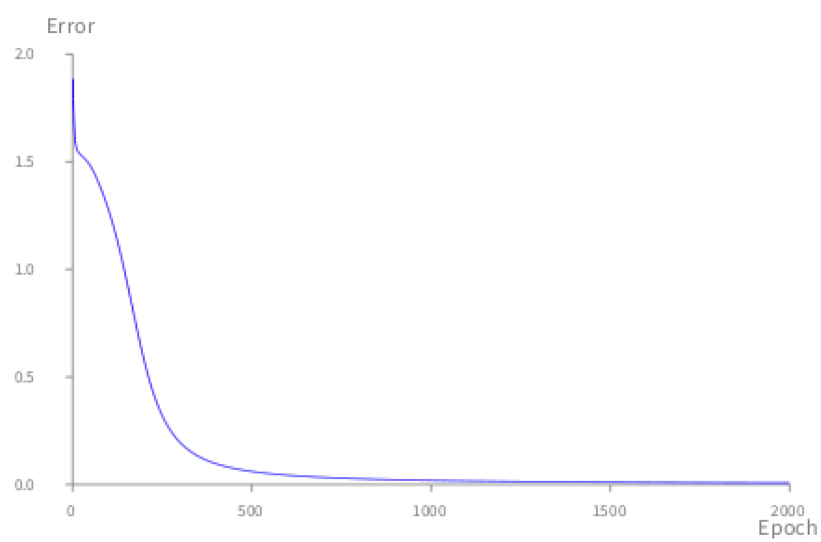
```
def calculate_cost(A2, Y):
    # m is the number of examples
    cost = np.sum((0.5 * (A2 - Y) ** 2).mean(axis=1))/m
    return cost
```

Normalization?

Example of training a network with the AND logical gate

```
data := {{0 . 0 . 0} .
         {0 . 1 . 0} .
         {1 . 0 . 0} .
         {1 . 1 . 1}}.

n := NeuralNetwork new.
n configure: 2 numberOfHidden: 1 nbOfOutput: 2.
n train: data nbEpoch: 2000.
n
```

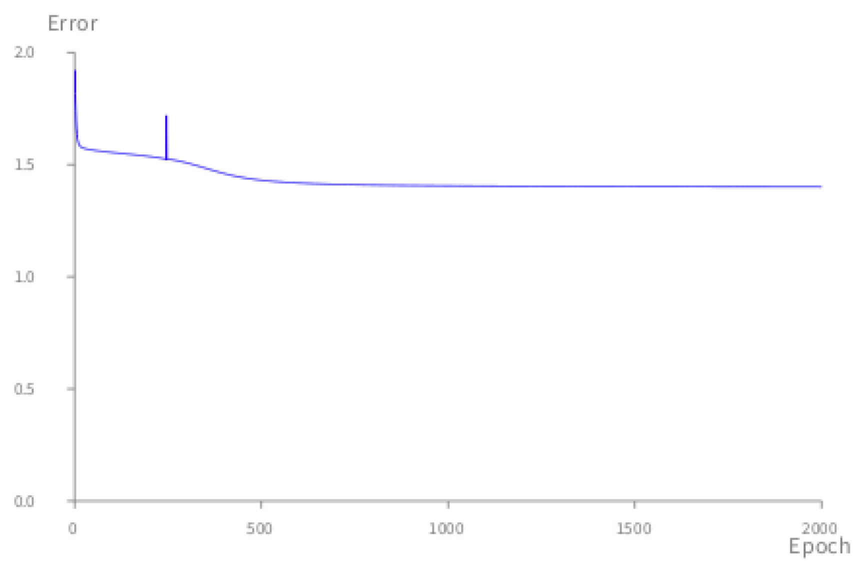


Normalization?

Replacing each 1 in the input by 50

```
data := {{0 . 0 . 0} .
         {0 . 50 . 0} .
         {50 . 0 . 0} .
         {50 . 50 . 1}}.

n := NeuralNetwork new.
n configure: 2 numberOfHidden: 1 nbOfOutput: 2.
n train: data nbEpoch: 2000.
n
```

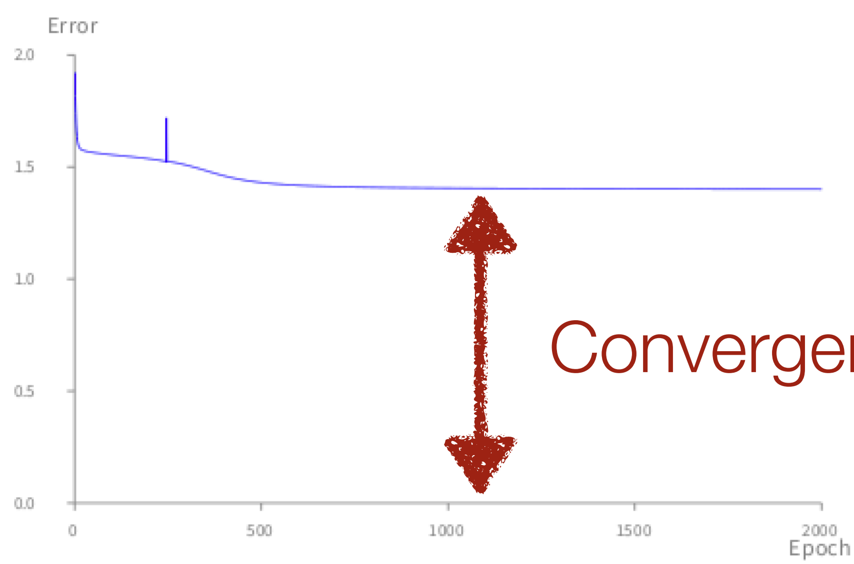


Normalization?

Replacing each 1 in the input by 50

```
data := {{0 . 0 . 0} .
         {0 . 50 . 0} .
         {50 . 0 . 0} .
         {50 . 50 . 1}}.

n := NeuralNetwork new.
n configure: 2 numberOfHidden: 1 nbOfOutput: 2.
n train: data nbEpoch: 2000.
n
```



Convergence to 0 is difficult

Need to normalize data

The sigmoid function returns a value between 0 and 1

Having the same range for the input improves the learning performance.

Each input should therefore *be between 0 and 1*

The process of transforming data from an arbitrary range to a restricted range is called *normalization*

Normalization

A bit of maths (but nothing terrible)

$$f(x) = \frac{(x - d_L)(n_H - n_L)}{(d_H - d_L)} + n_L$$

f(x) normalizes a value x

The variable d represents the high and low values of the data

N represents the high and low normalization range desired

Normalization

So, if a neuron input is between -10 and 10, then it has to be transformed as:

$$f(\text{input}) = (\text{input} - -10)\ (1 - 0)\ /\ (10 - -10) + 0$$

$$= (\text{input} + 10)\ /\ 20$$

Denormalization

When a neural network is used for regression, returned values are normalized. We therefore need to "denormalize" them

$$f(x) = \frac{(d_L - d_H)x - (n_H \cdot d_L) + d_H \cdot n_L}{(n_L - n_H)}$$

f(x) denormalizes a value x

The variable d represents the high and low values of the data

N represents the high and low normalization range desired

Prediction

Traditional way is to have the number of outputs the same size than the different class values

Consider a network that consists in classifying elements within N categories

The network works better with N outputs. The category corresponds to the output neuron with the maximum value

Outline

1. Performance of a neural network
2. **Tarea 1**

Training a NN over a dataset

To complete Tarea 1, you need:

1 - Implement a way to chart the cost functions during the training

2 - pick one dataset

3 - Implement the normalization

4 - Implement the one hot encoding transformation

5 - Produce the confusion matrix to represent the model test result

Bonuses

- use the k-Fold Cross-Validation. You can pick k = 3, 5, or 10

- have more than one dataset

- try different configuration of your network by varying the number of neurons in the hidden layer

- have all this in a programming language that is not Python

Datasets

You can pick the iris dataset, the seed dataset, or any other dataset available on

https://archive.ics.uci.edu/ml/datasets/seeds

https://archive.ics.uci.edu/ml/datasets

Fecha de entrega

Friday 9, October 2020