ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

CENG499
Introduction to Machine Learning

Recitation I

Yusuf Mücahit ÇETİNKAYA

Nov 01, 2018
METU, Ankara

Agenda

1. Definition of some terms in ML pipeline
2. Bag of words
3. Tf-Idf
4. One Hot Encoding
5. NLTK
6. NN API
7. Assignment 1

Agenda

1. Definition of some terms in ML pipeline
2. Bag of words
3. Tf-Idf
4. One Hot Encoding
5. NLTK
6. NN API
7. Assignment 1

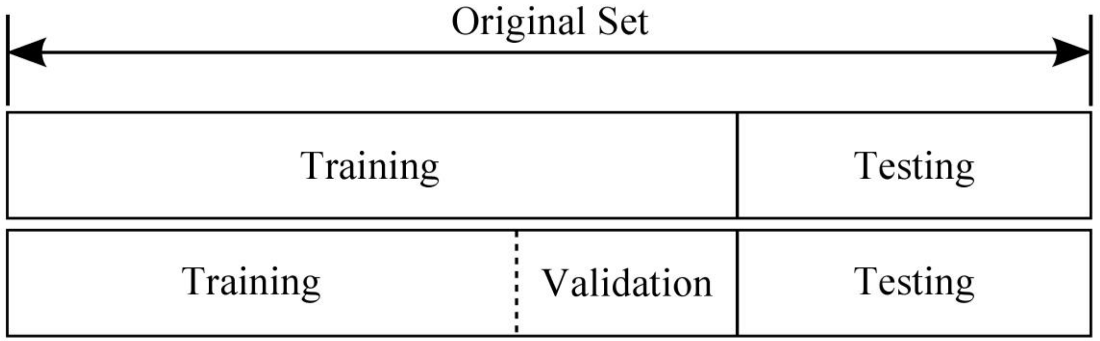
Some Terms

- Train set, test set, validation set
- Data Preprocessing
- Feature Extraction
- Feature Selection
- Cross-validation
- Confusion Matrix
- Vectorization (Data Representation)
- Perceptron
- Neural Network
- Activation Function
- Loss Function

Train set, test set, validation set

- ***The "training" data set is the general term for the samples used to create the model, while the "test" or "validation" data set is used to qualify performance.*** *(Applied Predictive Modeling, 2013)*

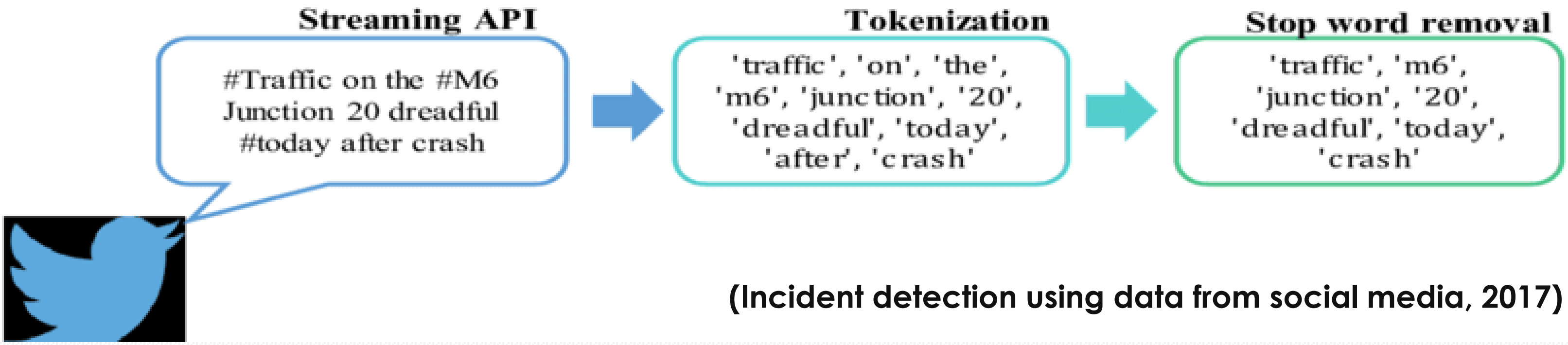
- **Training set: for model training**
- **Validation set: for evaluating performance during training**
- **Test set: for evaluating performance after training**



Data Preprocessing

- ***Data preprocessing is basically transforming raw data into a meaningful format because of various reasons.***
- **The main task of data preprocessing is to prune noisy and irrelevant data, and to reduce data volume for the pattern discovery phase** **(Web log cleaning for mining of web usage patterns, 2011)**

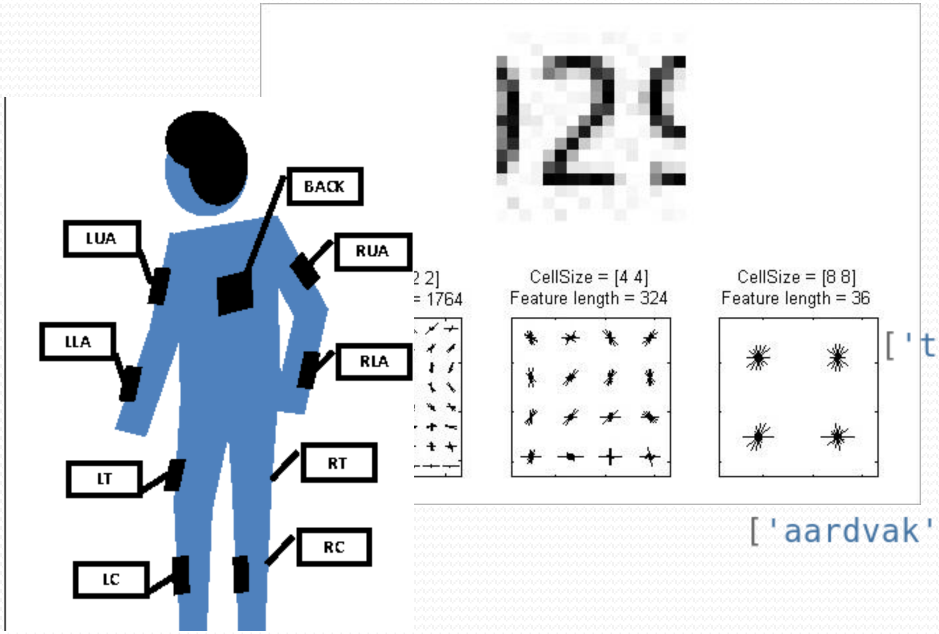
- **Data preprocessing includes *cleaning*, *instance selection*, *normalization*, *transformation*, *feature extraction* and *selection* etc.**

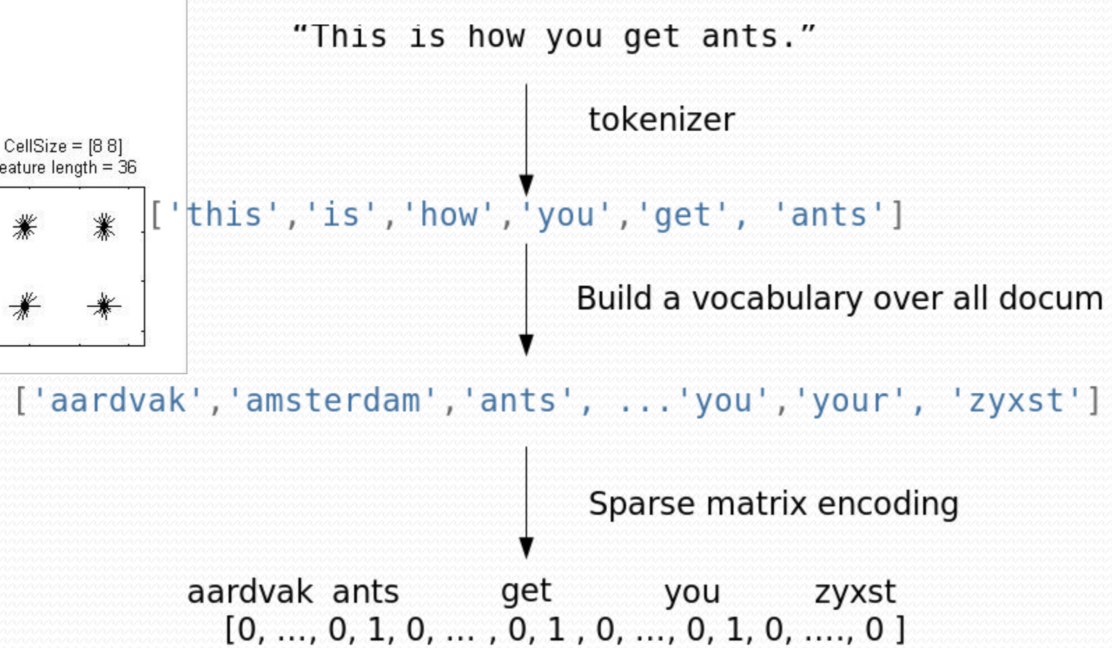


(Incident detection using data from social media, 2017)

Feature Extraction

- **Feature extraction is the name for methods that combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.** **(deepai.org)**



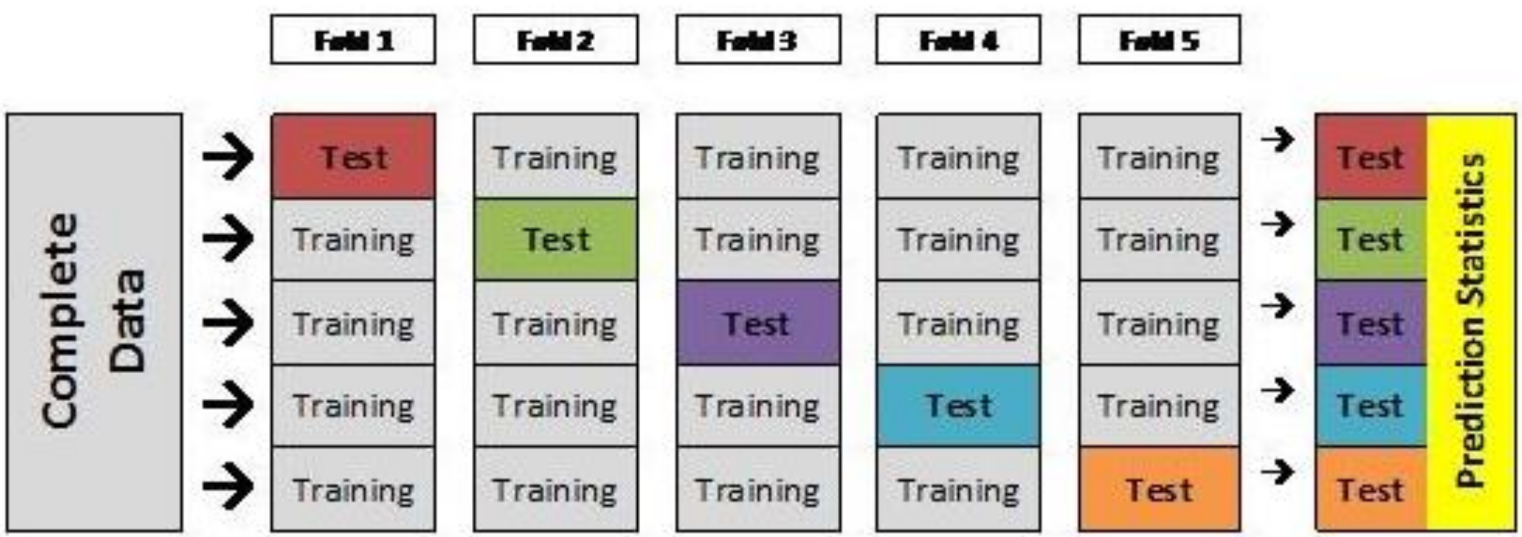


Feature Selection

- **Deciding on which features are most informative to increase model performance.**

Cross Validation

- **Model evaluation method to ensure the stability of trained model.**



Confusion Matrix

- **A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.**

Truth

Predicted	Asphalt	Concrete	Grass	Tree	Building	Total
Asphalt	2385	4	0	1	4	2394
Concrete	0	332	0	0	1	333
Grass	0	1	908	8	0	917
Tree	0	0	0	1084	9	1093
Building	12	0	0	6	2053	2071
Total	2397	337	908	1099	2067	6808

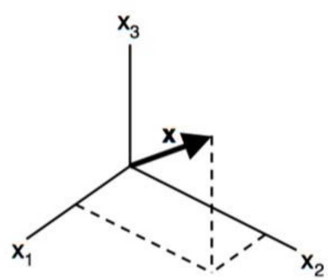
https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Vectorization (Data Representation)

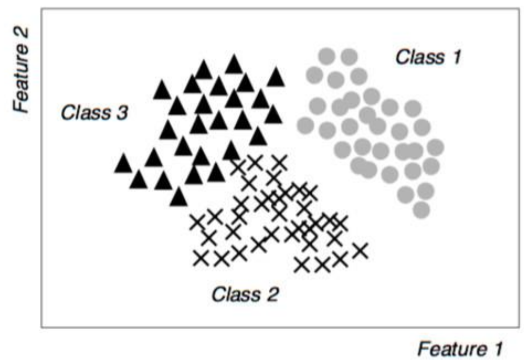
- **Representing an object with numeric or symbolic characteristics, called features, with a mathematical, easily analyzable way.**

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

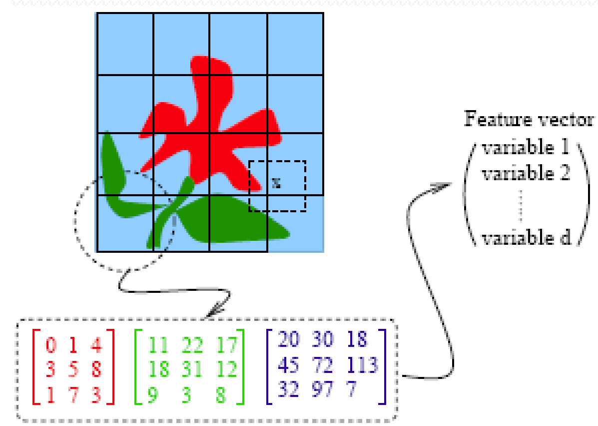
Feature vector



Feature space (3D)

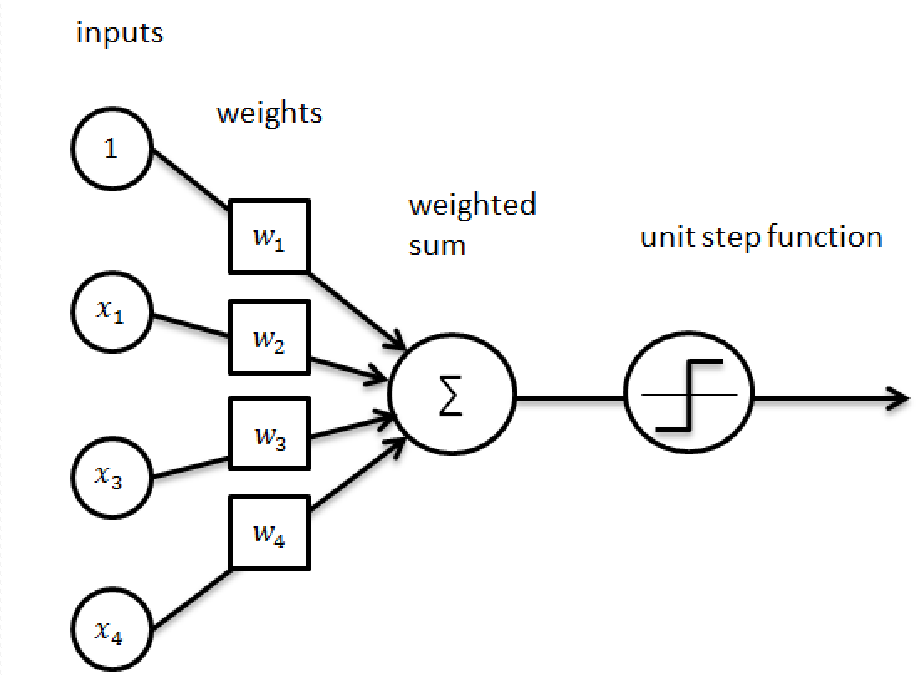


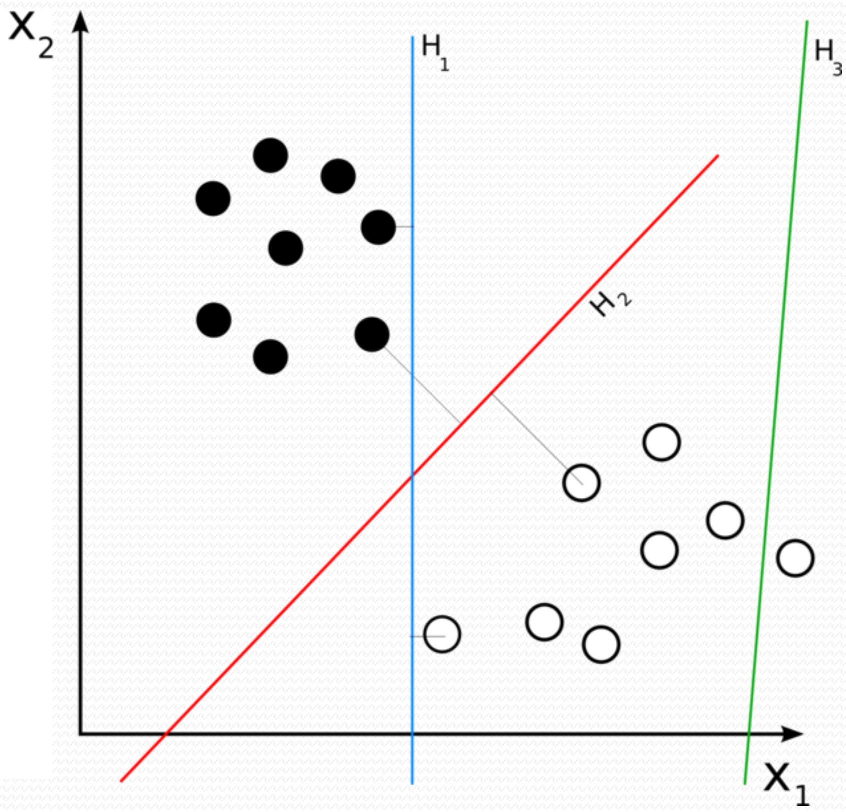
Scatter plot (2D)



Perceptron

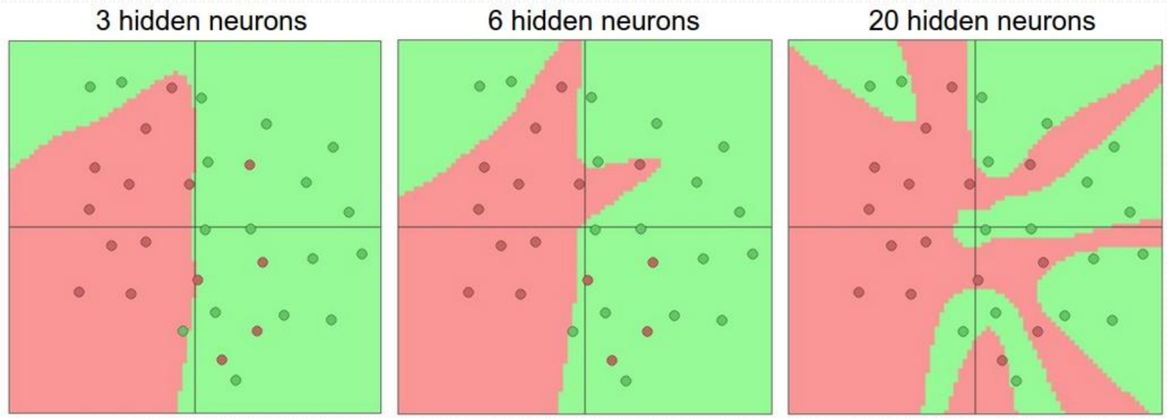
- **A simple linear classifier for binary classification.**

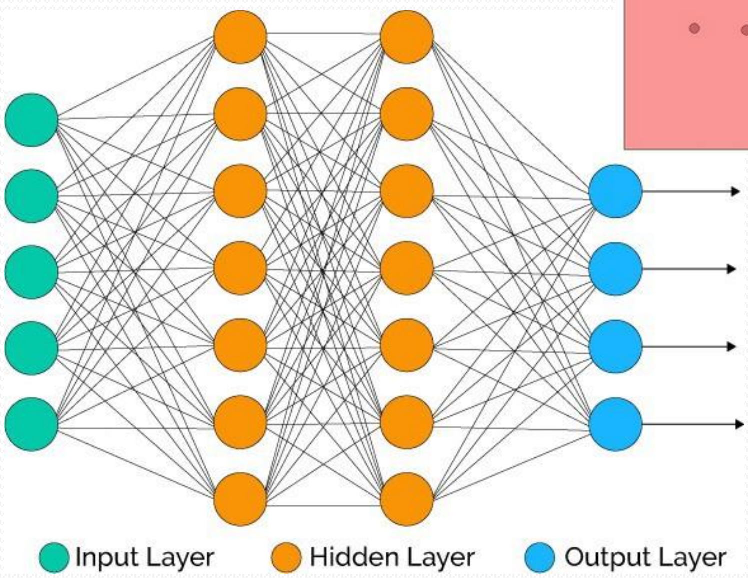




Neural Network

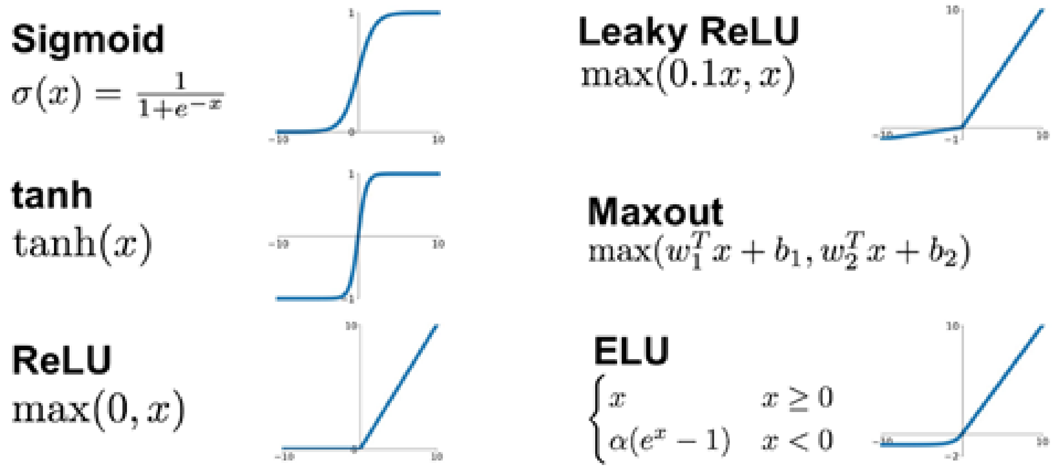
- **Multi-layer Perceptrons**
- **Universal function approximators**

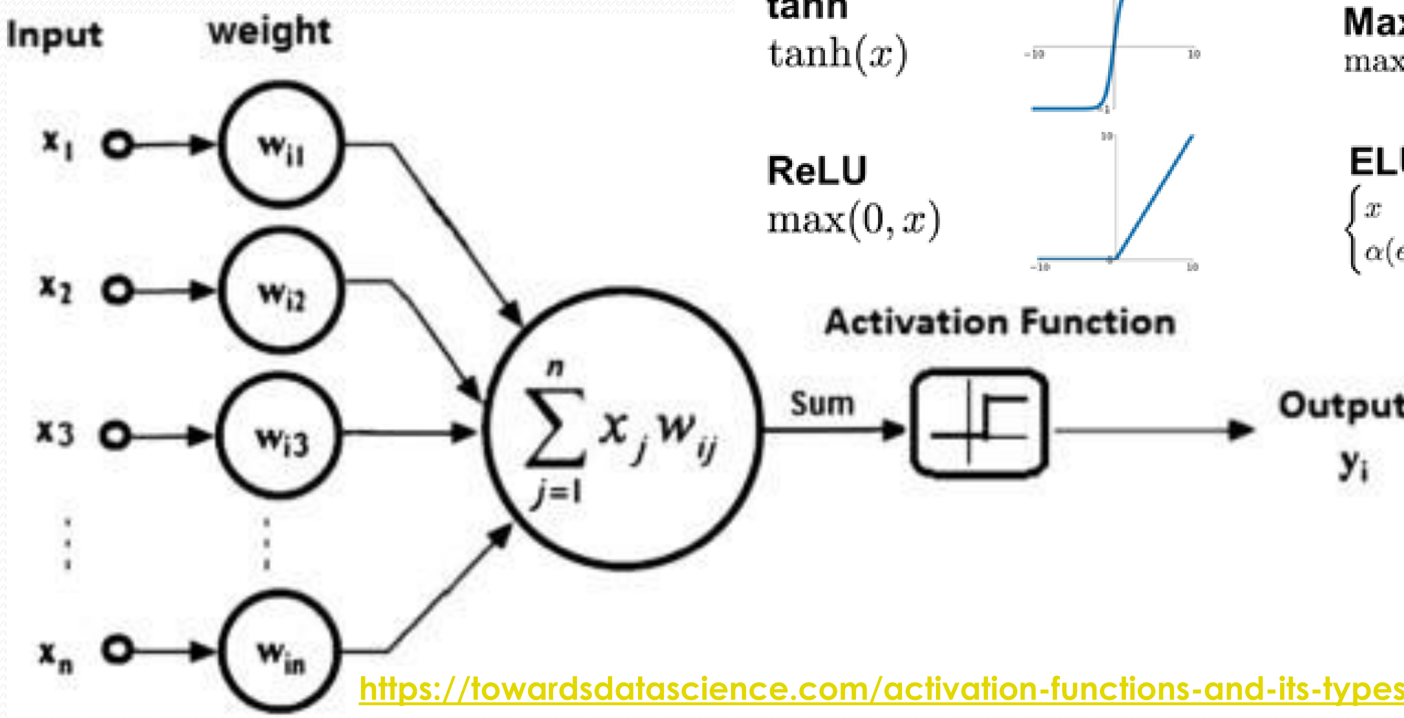




Activation Function

- **Just a function that takes the output of the neuron as input.**
- **Ex: tanh, relu, sigmoid etc.**





https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Loss Function

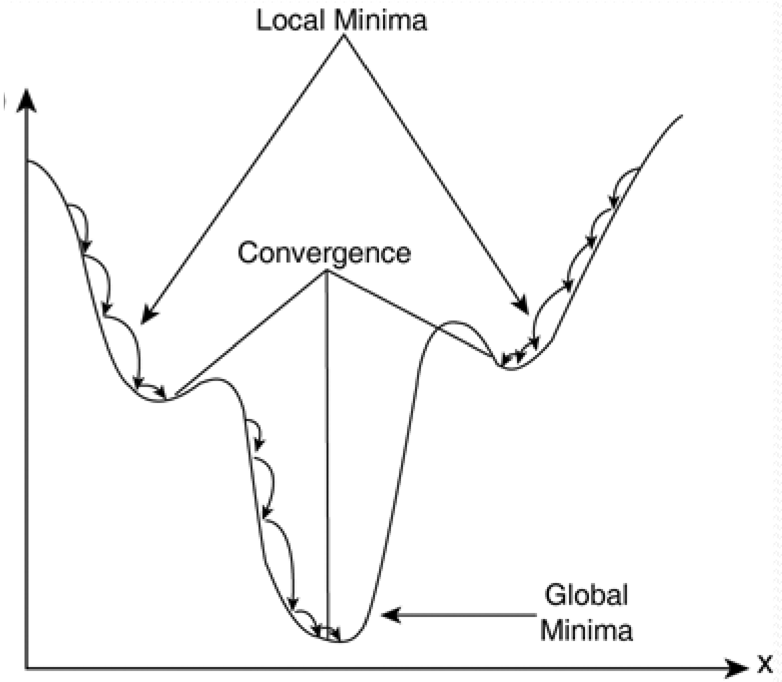
- **Measure of the model performance**

Crossentropy Loss

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

Hinge Loss

$$\ell(y) = \max(0, 1 - t \cdot y)$$



Agenda

1. **Definition of some terms in ML pipeline**
2. **Bag of words**
3. **Tf-Idf**
4. **One Hot Encoding**
5. **NLTK**
6. **NN API**
7. **Assignment 1**

Bag of Words

- **An algorithm to represent text data in a vectorized format.**

- **Ex:**
- **Doc 1: «I live in Ankara»**
- **Doc 2: «Ankara is good place to live good»**

- **Doc 3: «Where to live in Ankara»**
- **Doc 4: «My place is in Ankara»**

- **Vocabulary :**
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

Bag of Words

- **An algorithm to represent text data in a vectorized format.**

- **Ex:**
- **Doc 1: «I live in Ankara»** [1, 1, 0, 0, 0, 1, 0, 1, 0, 0]
- **Doc 2: «Ankara is good place to live good»**
  [1, 0, 0, 0, 1, 0, 1, 1, 1, 1]
- **Doc 3: «Where to live in Ankara»** [1, 0, 0, 1, 0, 1, 0, 1, 0, 1]
- **Doc 4: «My place is in Ankara»** [1, 0, 1, 0, 0, 1, 1, 0, 1, 0]

- **Vocabulary :**
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

Bag of Words

- **Can use number of occurences in the text.**

- **Ex:**
- **Doc 1: «I live in Ankara»** [1, 1, 0, 0, 0, 1, 0, 1, 0, 0]
- **Doc 2: «Ankara is good place to live good»**
  [1, 0, 0, 0, 2, 0, 1, 1, 1, 1]
- **Doc 3: «Where to live in Ankara»** [1, 0, 0, 1, 0, 1, 0, 1, 0, 1]
- **Doc 4: «My place is in Ankara»** [1, 0, 1, 0, 0, 1, 1, 0, 1, 0]

- **Vocabulary :**
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

Agenda

1. **Definition of some terms in ML pipeline**
2. **Bag of words**
3. **Tf-Idf**
4. **One Hot Encoding**
5. **NLTK**
6. **NN API**
7. **Assignment 1**

Tf-Idf

- **Very popular algorithm to represent text data in a vectorized format.**
- **Computes also the importance of a term in the collection(IDF).**

- **Tf : Term Frequency**
- **Idf: Inverse Document Frequency**

$$\text{TF-IDF}(term,\, document) = \text{TF}(term,\, document)^{*}\text{IDF}(term)$$

$$\text{TF}(term,\, document) = \frac{\text{Number of times } term \text{ appears in } document}{\text{Total number of terms in } document}$$

$$\text{IDF}(term) = log(\frac{1 + \text{Total number of documents}}{1 + \text{Total number of documents where } term \text{ contained}})$$

Tf-Idf

- Ex:
- Doc 1: «I live in Ankara»
- Doc 2: «Ankara is good place to live good»
- Doc 3: «Where to live in Ankara»
- Doc 4: «My place is in Ankara»

- Vocabulary :
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

- Number of documents terms contained:
  {'I': 1, 'good': 1, 'Ankara': 4, 'is': 2, 'Where': 1,
  'place': 2, 'in': 3, 'My': 1, 'live': 3, 'to': 2}

Tf-Idf

- **Vocabulary :**

  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

- **Number of documents terms contained:**

  **{'I': 1, 'good': 1, 'Ankara': 4, 'is': 2, 'Where': 1,
  'place': 2, 'in': 3, 'My': 1, 'live': 3, 'to': 2}**

- **Idfs:**

  {'I': 0.398, 'good': 0.398, 'Ankara': 0.0, 'is': 0.222, 'Where': 0.398, 'place': 0.222, 'in': 0.097, 'My': 0.398, 'live': 0.097, 'to': 0.222}

Tf-Idf

- **Ex:**
- **Doc 1: «I live in Ankara»**

- **Vocabulary :**
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

- **TF('I', Doc 1) = 1 / 4 = 0.25**
  IDF('I') = 0.398
  TF-IDF('I', Doc 1) = 0.25 * 0.398 = 0.0995
- **TF('live', Doc 1) = 1 / 4 = 0.25**
  IDF('live') = 0.097
  TF-IDF('live', Doc 1) = 0.25 * 0.097 = 0.0242

Tf-Idf

- Ex:
- Doc 1: «I live in Ankara»

- Vocabulary :
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

- TF('in', Doc 1) = 1 / 4 = 0.25
  IDF('in') 0.097
  TF-IDF('in', Doc 1) = 0.25 * 0.097 = 0.0242
- TF('Ankara', Doc 1) = 1 / 4 = 0.25
  IDF('Ankara') = 0
  TF-IDF('Ankara', Doc 1) = 0.25 * 0 = 0

Tf-Idf

- **Ex:**
- **Doc 1: «I live in Ankara»**

- **Vocabulary :**
  ['Ankara', 'I', 'My', 'Where', 'good', 'in', 'is', 'live', 'place', 'to']

- **TF-IDF(Doc 1) = [0, 0.0995, 0, 0, 0, 0.0242, 0, 0.0242, 0, 0]**

Agenda

1. Definition of some terms in ML pipeline
2. Bag of words
3. Tf-Idf
4. One Hot Encoding
5. NLTK
6. NN API
7. Assignment 1

One Hot Encoding

- **Encoding categorical features as a one-hot numeric array.**
- **Derives the categories based on the unique values in each feature.**

color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

One Hot Encoding

- **If there is no ordinal relationship between categories can't use enumeration.**
  - **It seperates the dimensions of each label.**
- **Integer encoding can be used which is only one dimension for ordinal data;**
  - **Baby -> 0, Teen -> 1, Young Adult -> 2, Adult ->3**

Agenda

1. **Definition of some terms in ML pipeline**
2. **Bag of words**
3. **Tf-Idf**
4. **One Hot Encoding**
5. **NLTK**
6. **NN API**
7. **Assignment 1**

NLTK

- **Natural Language Toolkit**
- **A suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.**
- www.nltk.org

- **A token is the technical name for a sequence of characters.**

Slide Type -

```
from nltk.tokenize import word_tokenize

words = word_tokenize('''Nice book! Though it is lack of advanced topics.
                         it's still good for beginners.'''.lower())
print(words)
```

```
['nice', 'book', '!', 'though', 'it', 'is', 'lack', 'of', 'advanced', 'top
ics', '.', 'it's', 'still', 'good', 'for', 'beginners', '.']
```

NLTK

- **By using lower(), we have normalized the text to lowercase. Often we want to go further than this, and strip off any affixes, a task known as stemming.**
- **NLTK includes several off-the-shelf stemmer;**
  - **PorterStemmer, LancasterStemmer SnowballStemmer etc.**

```
stems = []
stemmer = SnowballStemmer("english")
for token in tokens:
    token = stemmer.stem(token)
    if token != "":
        stems.append(token)
```

Agenda

1. Definition of some terms in ML pipeline
2. Bag of words
3. Tf-Idf
4. One Hot Encoding
5. NLTK
6. NN API
7. Assignment 1

Tensorflow

- TensorFlow is an open source software library for high performance numerical computation.
- Provides primitives for defining functions on tensors and automatically computing their derivatives.
- www.tensorflow.org
- https://cow.ceng.metu.edu.tr/Courses/download_courseFile.php?id=9730
- It used to have only these primitive types.

Tensorflow – Keras API

- **Now it includes a high-level API to build and train models.**

```
import tensorflow as tf
from tensorflow import keras
```

- **Builds the model, layer by layer with predefined classes.**
    - **The most common type of model is a stack of layers: the <u>tf.keras.Sequential</u> model.**

```
model = keras.Sequential()
```

```
layers.Dense(<Number of nodes in the layer>, activation=<activation function>)
```

Tensorflow – Keras API

- **Each layer can be created seperately and added to the model later.**

```
model = keras.Sequential()
# Adds a densely-connected layer with 64 units to the model:
model.add(keras.layers.Dense(64, activation='relu'))
# Add another:
model.add(keras.layers.Dense(64, activation='relu'))
# Add a softmax layer with 10 output units:
model.add(keras.layers.Dense(10, activation='softmax'))
```

Tensorflow – Keras API

- **After model is ready, it is compiled with learning configurations.**

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- **Can get customized metrics as parameter.**

```
my_metric = tf.keras.metrics.top_k_categorical_accuracy
```

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='categorical_crossentropy',
              metrics=['accuracy', my_metric])
```

Tensorflow – Keras API

- **After model is compiled, it is ready to train. It takes numpy arrays as train, test data and labels.**

```
import numpy as np

data = np.random.random((1000, 32))
labels = np.random.random((1000, 10))

model.fit(data, labels, epochs=10, batch_size=32)
```

- **epoch: one iteration over the entire dataset**
- **batch_size: number of samples to calculate loss and update weights**

Tensorflow – Keras API

- **Can follow current performance according to metrics, during training with validation data.**
- **Validation data is not used while updating weights, in other words while training.**

```
import numpy as np

data = np.random.random((1000, 32))
labels = np.random.random((1000, 10))

val_data = np.random.random((100, 32))
val_labels = np.random.random((100, 10))

model.fit(data, labels, epochs=10, batch_size=32,
          validation_data=(val_data, val_labels))
```

Tensorflow – Keras API

- **After training is finished *evaluate* method can be used to compute the performance of the model on a given dataset.**
- **Returns an array with the size of metrics. Order is same with the given metrics array during compilation.**

```
model.evaluate(x, y)
```

Tensorflow – Keras API

- **To get predictions on a given dataset, predict method of model can be used.**
- **Returns a numpy array with shape of (sample_count, number_of_labels)**

```
Y_predict = model.predict(x)

y_predict = np.argmax(Y_predict, axis=1)
```

Tensorflow – Keras API

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/_index.ipynb

Tensorflow – Keras API

- To visualize and trace the live learning process, tensorboard can be used.
- Create a callback for tensorboard.

```
tbCallBack = tf.keras.callbacks.TensorBoard(log_dir="my_log_dir",
                  histogram_freq=0, write_graph=True, write_images=True)
```

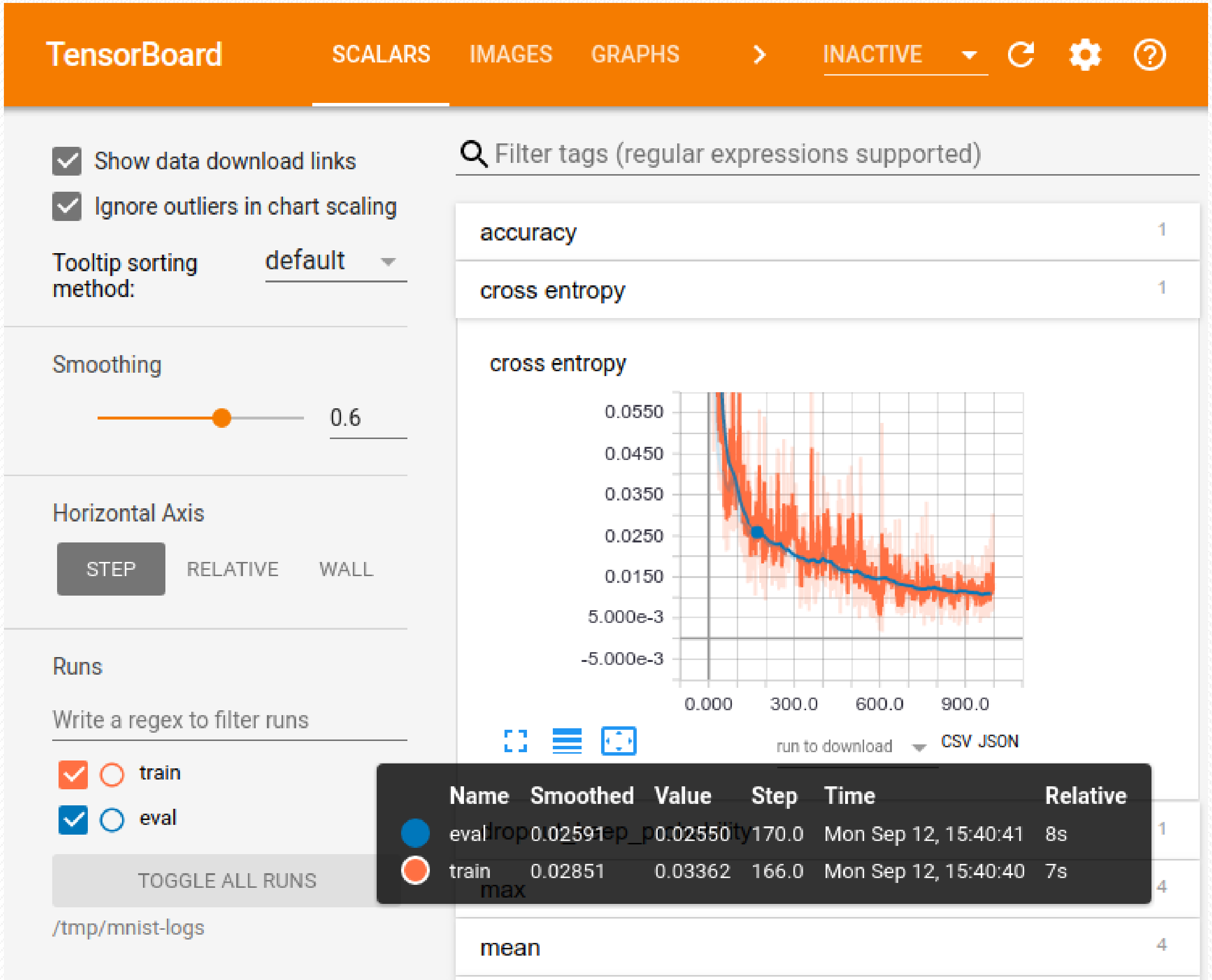
- Give this callback to ***callbacks*** parameter of ***fit*** method.
  - It runs these callbacks after each epoch.

```
model.fit(...inputs and parameters...,callbacks=[tbCallBack])
```

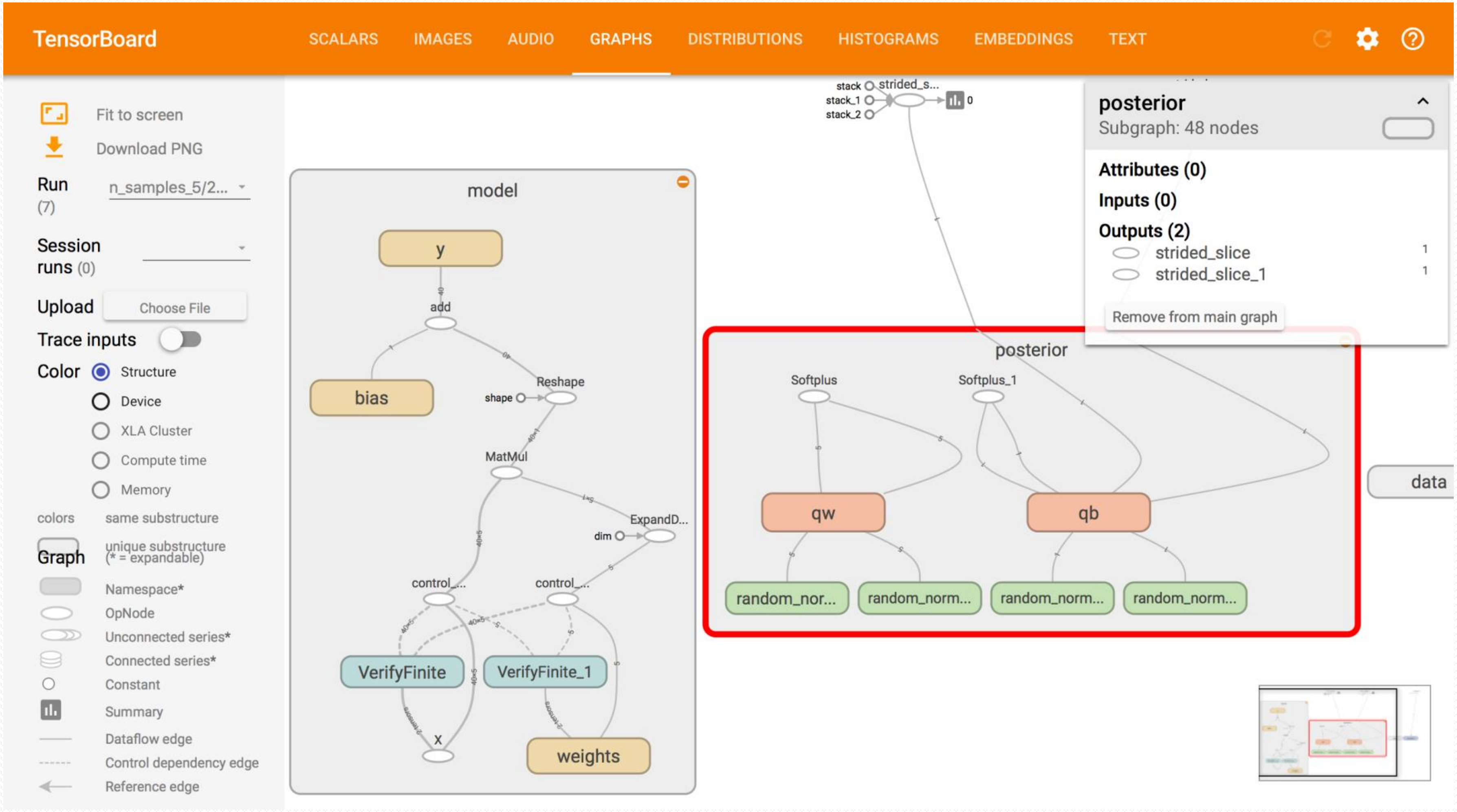
- Run tensorboard by giving log directory from command line.

```
tensorboard --logdir=path/to/log-directory
```

Tensorflow – Keras API



Tensorflow – Keras API



Agenda

1. Definition of some terms in ML pipeline
2. Bag of words
3. Tf-Idf
4. One Hot Encoding
5. NLTK
6. NN API
7. Assignment 1

Thank you for your attention.