Mathematical techniques in data science
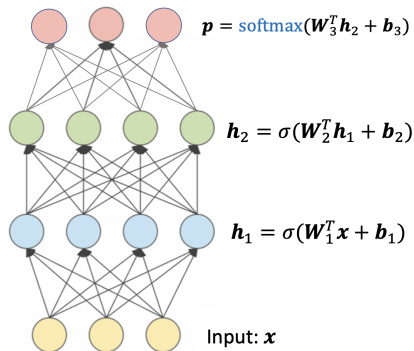
Lecture 6: Deep learning with Keras

Feed-forward neural networks

- Structure:
  - Graphical representation
  - Activation functions
  - Loss functions
- Training:
  - Stochastic gradient descent
  - Back-propagation

Settings

- Data:
  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- Model parameters:

  $\theta = (W_1, b_1, W_2, b_2, \ldots, W_L, b_L)$

- Training: Find the best value of θ that fits the data

$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{W}_3^T \boldsymbol{h}_2 + \boldsymbol{b}_3)$

$\boldsymbol{h}_2 = \sigma(\boldsymbol{W}_2^T \boldsymbol{h}_1 + \boldsymbol{b}_2)$

$\boldsymbol{h}_1 = \sigma(\boldsymbol{W}_1^T \boldsymbol{x} + \boldsymbol{b}_1)$

Input: $\boldsymbol{x}$

Stochastic gradient descent: terminology

- Mini-batch stochastic gradient descent
    - randomly shuffle examples in the training set, divide them into k mini-batches of data of size m
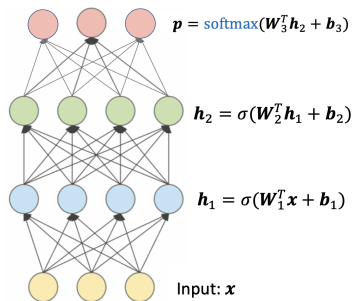   - for each batch I_i ($i=1, \ldots, k$), approximate the objective function by

    $$\hat{\ell}(\theta) = \frac{1}{m} \sum_{j \in I_i} L(\theta, x_j, y_j)$$

    and update θ

    $$\theta \leftarrow \theta - \rho \nabla \hat{\ell}(\theta)$$

    - Repeat until an approximate minimum is obtained or a maximum numbers M epochs are done
- Terminology:
    - m: batch-size
    - ρ: learning rate
    - M: number of epochs

$$\boldsymbol{p} = \text{softmax}(\boldsymbol{W}_3^T \boldsymbol{h}_2 + \boldsymbol{b}_3)$$

$$\boldsymbol{h}_2 = \sigma(\boldsymbol{W}_2^T \boldsymbol{h}_1 + \boldsymbol{b}_2)$$

$$\boldsymbol{h}_1 = \sigma(\boldsymbol{W}_1^T \boldsymbol{x} + \boldsymbol{b}_1)$$

Input: $\boldsymbol{x}$

- One forward pass to evaluate h_1, h_2, p, ℓ
- One backward pass to compute $\nabla \ell(\theta)$

`sklearn.neural_network`.MLPClassifier

class sklearn.neural_network.MLPClassifier(*hidden_layer_sizes=(100,), activation='relu', *,
solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001,
power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False,
warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10,
max_fun=15000*)

[source]

Multi-layer Perceptron classifier.

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

New in version 0.18.

Parameters:	**hidden_layer_sizes : *tuple, length = n_layers - 2, default=(100,)***
	The ith element represents the number of neurons in the ith hidden layer.
	activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
	Activation function for the hidden layer.
	• 'identity', no-op activation, useful to implement linear bottleneck, returns $f(x) = x$
	• 'logistic', the logistic sigmoid function, returns $f(x) = 1 / (1 + \exp(-x))$.
	• 'tanh', the hyperbolic tan function, returns $f(x) = \tanh(x)$.
	• 'relu', the rectified linear unit function, returns $f(x) = \max(0, x)$

Keras

Simple. Flexible. Powerful.

- High level API for deep learning
- More flexible to define network architecture than sklearn
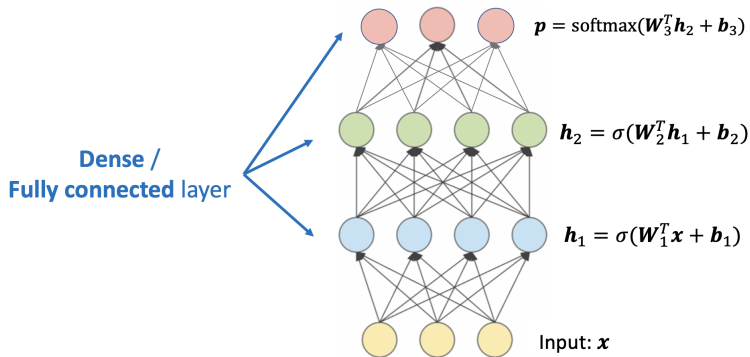
Define network architecture (1)

- Define a network as a Sequential object
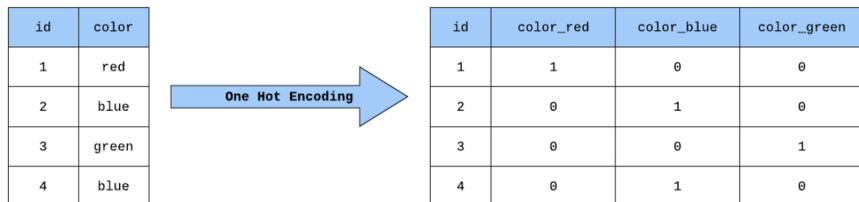- Add layers to it one-by-one

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(50, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Dense / Fully connected layer

$$\boldsymbol{p} = \text{softmax}(\boldsymbol{W}_3^T \boldsymbol{h}_2 + \boldsymbol{b}_3)$$

$$\boldsymbol{h}_2 = \sigma(\boldsymbol{W}_2^T \boldsymbol{h}_1 + \boldsymbol{b}_2)$$

$$\boldsymbol{h}_1 = \sigma(\boldsymbol{W}_1^T \boldsymbol{x} + \boldsymbol{b}_1)$$

Input: $\boldsymbol{x}$

One-hot encoding

id	color
1	red
2	blue
3	green
4	blue

One Hot Encoding →

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Labels in Keras are usually encoded as one-hot vectors

Some intros to computer vision

Computer vision

A field that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs

- Image classification/object recognition
- Object detection
- Image segmentation
- Image generation
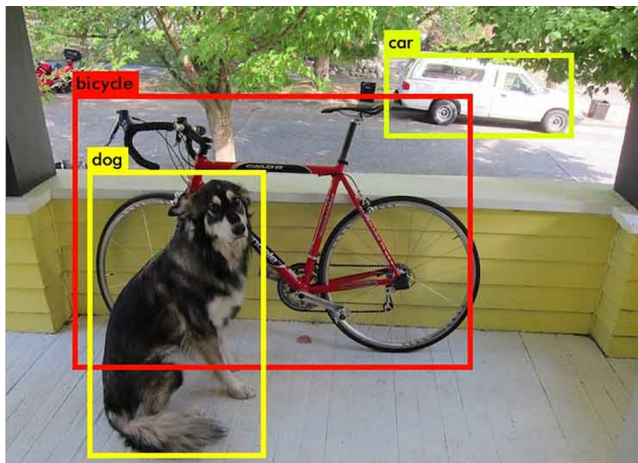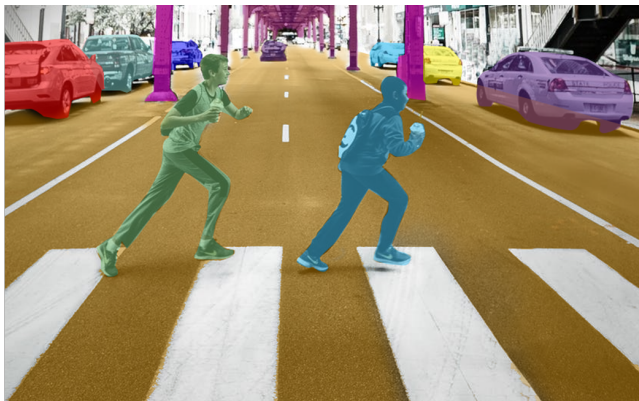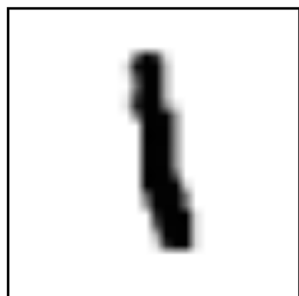
Image classification
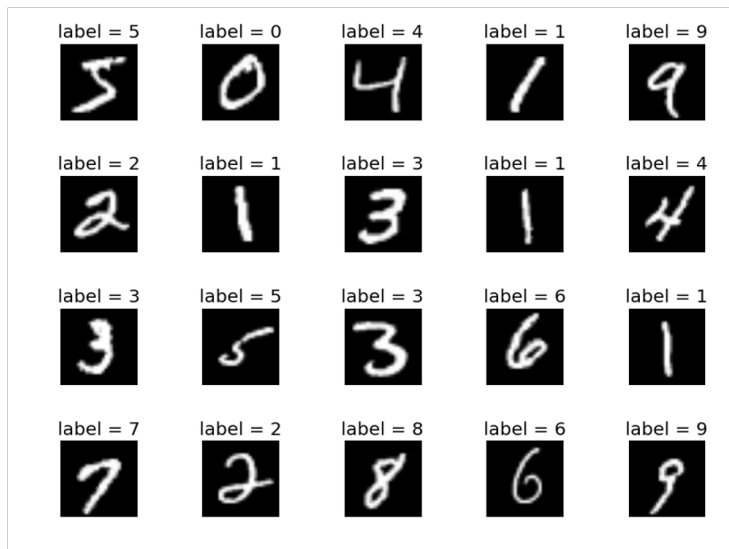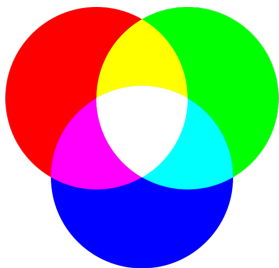
Object detection

Image segmentation

Grayscale image representation

$$\cong \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Color image representation

- Use RGB color mode
- Represent a color by 3 values: R (Red) G (Green) B (Blue)
- There are other color modes

Image representation
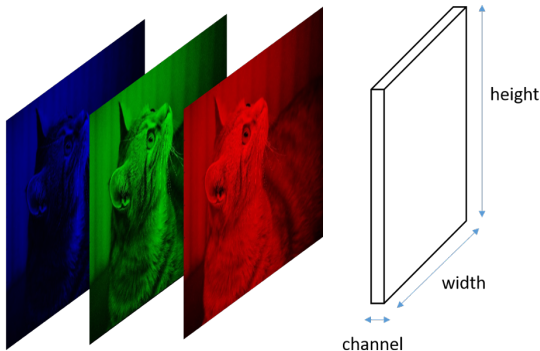
- An image is an H x W x C matrix: H (height), W (width), C (depth or number of channels)
- Grayscale image: $C = 1$
- RGB image: $C = 3$

Demo: train an MLP using Keras