

Data Science Survival Skills

Artificial Intelligence, Machine Learning, Deep Learning

Exercises Information

- DO NOT JUST COPY RANDOM CODE FROM GOOGLE!!!!

We want you to use the concepts that we just explained a couple of cells before!

EXAMPLE: first and last frame of a video: Example Code:

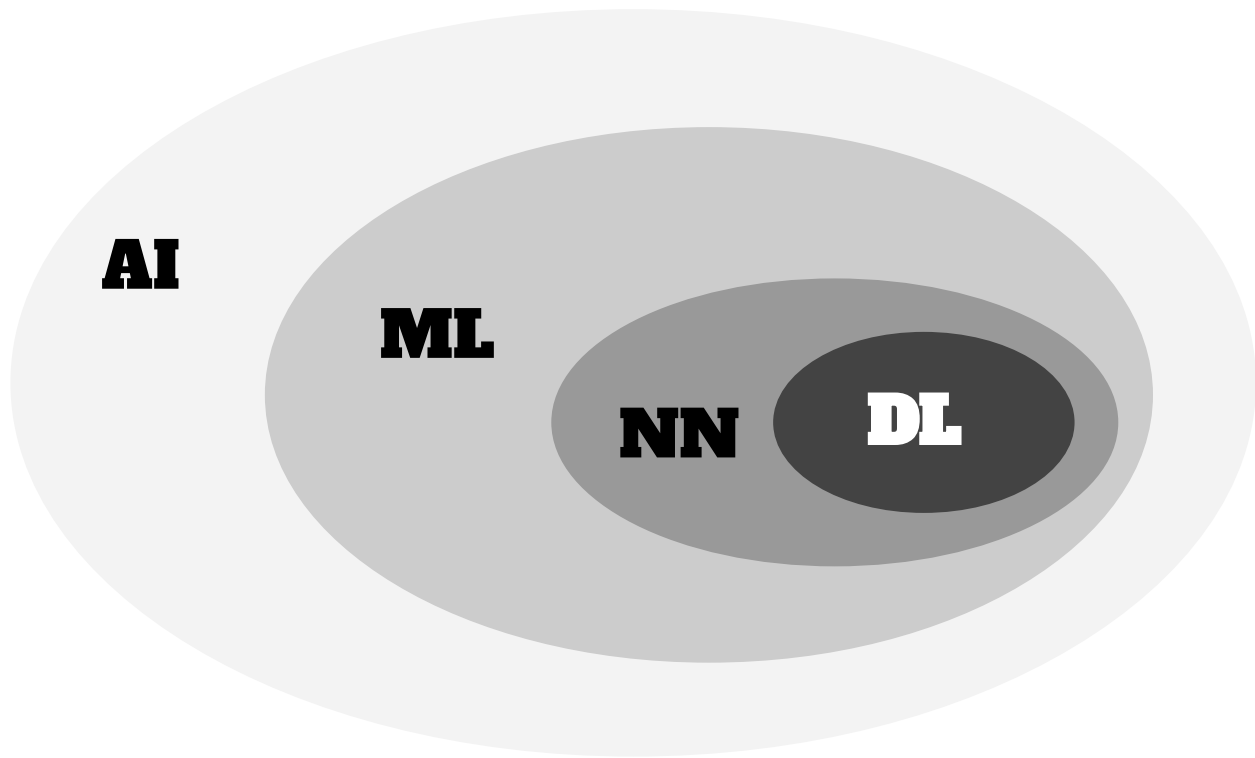
vid[0] and vid[-1], not

```
1. import cv2
2. import os
3.
4. def extractFrames(pathIn, pathOut):
5.     os.mkdir(pathOut)
6.
7.     cap = cv2.VideoCapture(pathIn)
8.     count = 0
9.
10.    while (cap.isOpened()):
11.
12.        # Capture frame-by-frame
13.        ret, frame = cap.read()
14.
15.        if ret == True:
16.            print('Read %d frame: ' % count, ret)
17.            cv2.imwrite(os.path.join(pathOut, "frame{:d}.jpg".format(count)), frame)
18.            count += 1
19.        else:
20.            break
21.
22.    # When everything done, release the capture
23.    cap.release()
24.    cv2.destroyAllWindows()
25.
26. def main():
27.     extractFrames('bigbuckbunny720p5mb.mp4', 'data')
28.
29. if __name__ == "__main__":
30.     main()
```

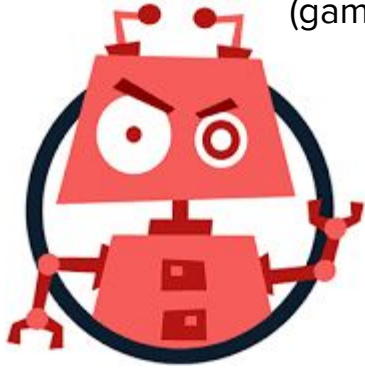
Agenda

- What is AI/ML/DL?
- How can we fit a line?
- How can we categorize two groups?
- Dive in Machine Learning
- Dive in Deep Learning

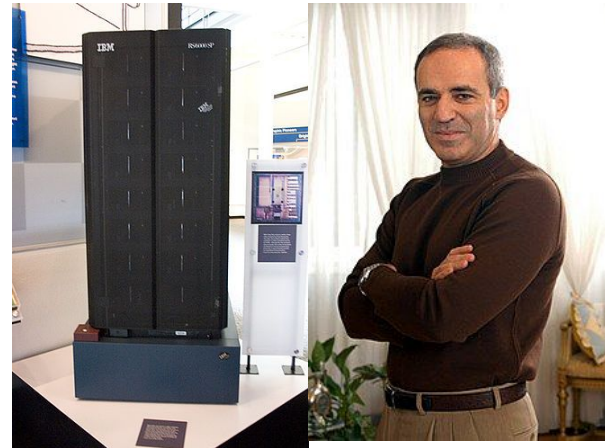
Wait! Aren't we doing AI? Or Deep Learning?



What is intelligence?



Computer
(game) bots



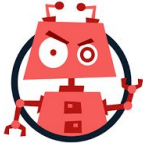
pages. The one exception was in a list of frequently asked questions, one of which was, “Does Deep Blue use artificial intelligence?” Even more surprisingly, IBM’s answer to this question was “no”! (IBM, 1997)



Artificial Intelligence

Narrow

Weak



SPECIFIC TASK

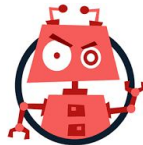
-- NOW --

General

Strong

HUMAN-LIKE

-- FUTURE --



Super

Strong

TRANSCENDENT

?????

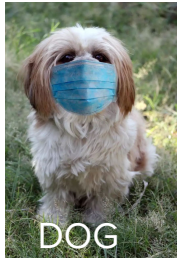


How can machines learn?

Learning = not explicitly programmed !!!

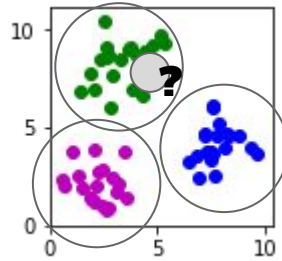
Supervised

e.g. classification, labelled data



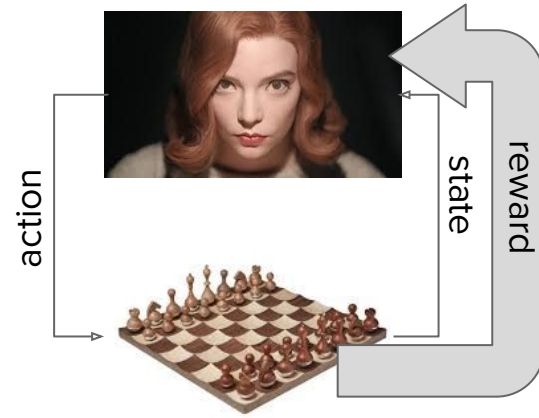
Unsupervised

e.g. clustering, unlabelled data

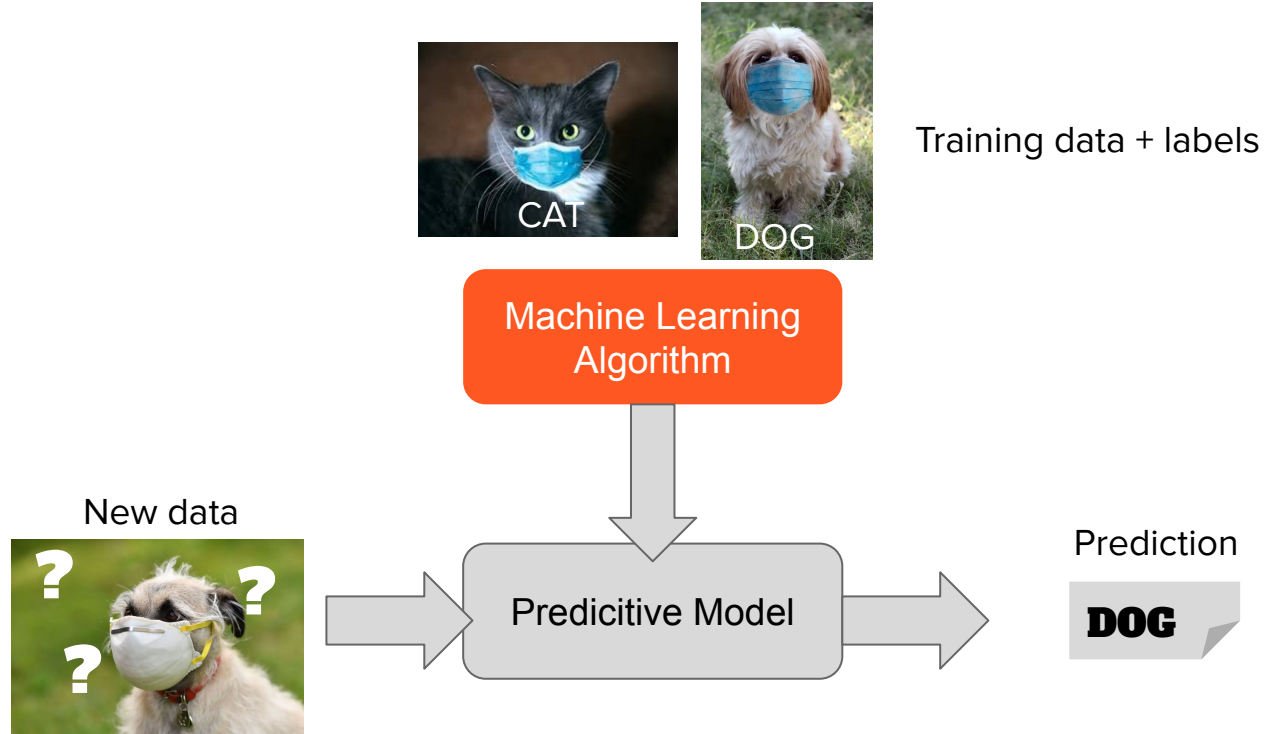


Reinforcement

e.g. complex games, like chess

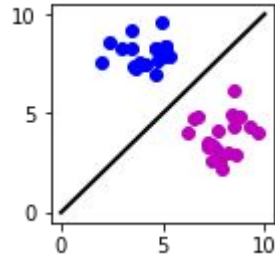


Supervised learning

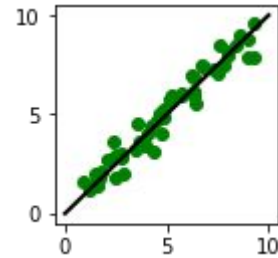


Two main types of problems (supervised learning)

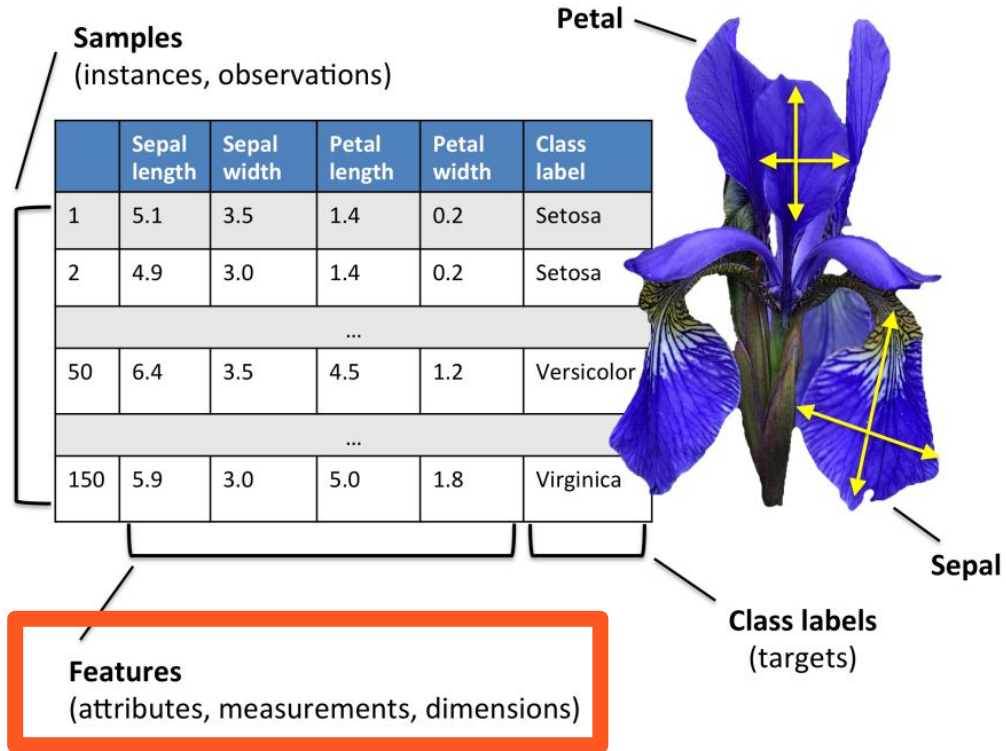
CLASSIFICATION



REGRESSION

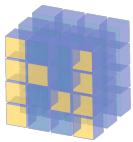


Terminology



Math and I/O packages

Processing



NumPy

Multi-dimensional image processing (scipy.ndimage)



Reading/Saving

imageio - Python library for reading and writing image data



flammkuchen

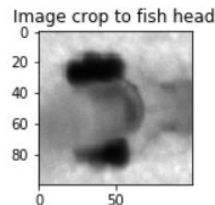


Plotting

matplotlib

seaborn: statistical data visualization

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 import imageio as io
5 # ImageIO to load an image (e.g. training data)
6 im = io.imread("zebrafish.png")
7 # Numpy to process image (e.g. crop)
8 im_crop = im[220:320, 100:200]
9 # Matplotlib to e.g. show data or training progress
10 plt.figure(figsize=(2,2))
11 plt.imshow(im_crop, cmap='gray')
12 plt.title("Image crop to fish head");
```



Machine learning packages (the big ones)



PCA
ICA
Linear Regression
SVMs
...



Optimizer
Filtering
Stats
...

Neural networks etc



Machine learning packages (the new kids on the block)



JAX: Autograd and XLA

Continuous integration [passing](#) [pypi](#) [40.2.0](#)

[Quickstart](#) | [Transformations](#) | [Install guide](#) | [Neural net libraries](#) | [Change logs](#) | [Reference docs](#) | [Code search](#)

News: JAX tops largest-scale MLPerf Training 0.7 benchmarks!

What is JAX?

JAX is Autograd and XLA, brought together for high-performance machine learning research.



Acme: a research framework for reinforcement learning

[Overview](#) | [Installation](#) | [Documentation](#) | [Agents](#) | [Examples](#) | [Paper](#) | [Blog post](#)

[python](#) [3.6](#) | [3.7](#) | [pypi package](#) [0.2.0](#) | [pytest](#) [failing](#)

Acme is a library of reinforcement learning (RL) agents and agent building blocks. Acme strives to expose simple, efficient, and readable agents, that serve both as reference implementations of popular algorithms and as strong baselines, while still providing enough flexibility to do novel research. The design of Acme also attempts to provide multiple points of entry to the RL problem at differing levels of complexity.

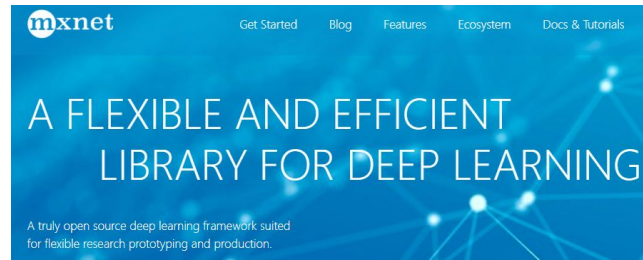
Flax: A neural network library and ecosystem for JAX designed for flexibility

[Build](#) [passing](#) [coverage](#) [81.6%](#)

[Overview](#) | [Quick install](#) | [What does Flax look like?](#) | [Documentation](#)

See our [full documentation](#) to learn everything you need to know about Flax.

Flax was originally started by engineers and researchers within the Brain Team in Google Research (in close collaboration with the JAX team), and is now developed jointly with the open source community.



Haiku: Sonnet for JAX

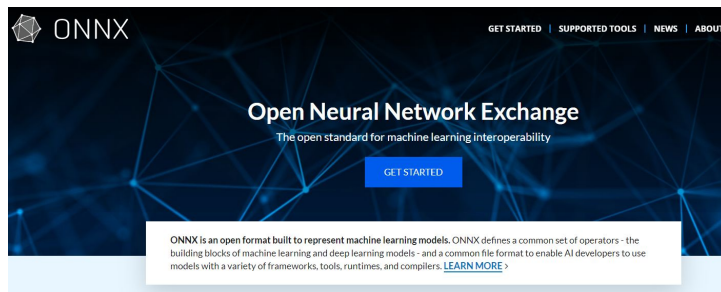
[Overview](#) | [Why Haiku?](#) | [Quickstart](#) | [Installation](#) | [Examples](#) | [User manual](#) | [Documentation](#) | [Citing Haiku](#)

[pytest](#) [passing](#)

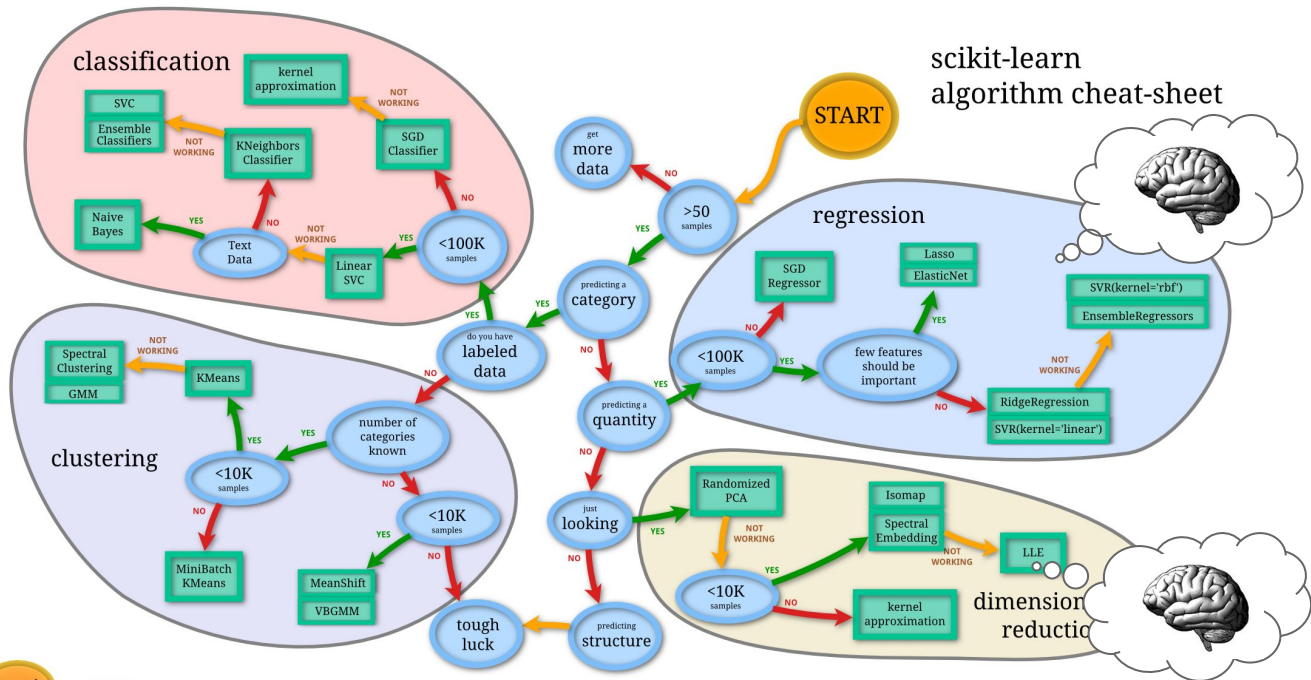
What is Haiku?

Haiku is a tool
For building neural networks
Think: "Sonnet for JAX"

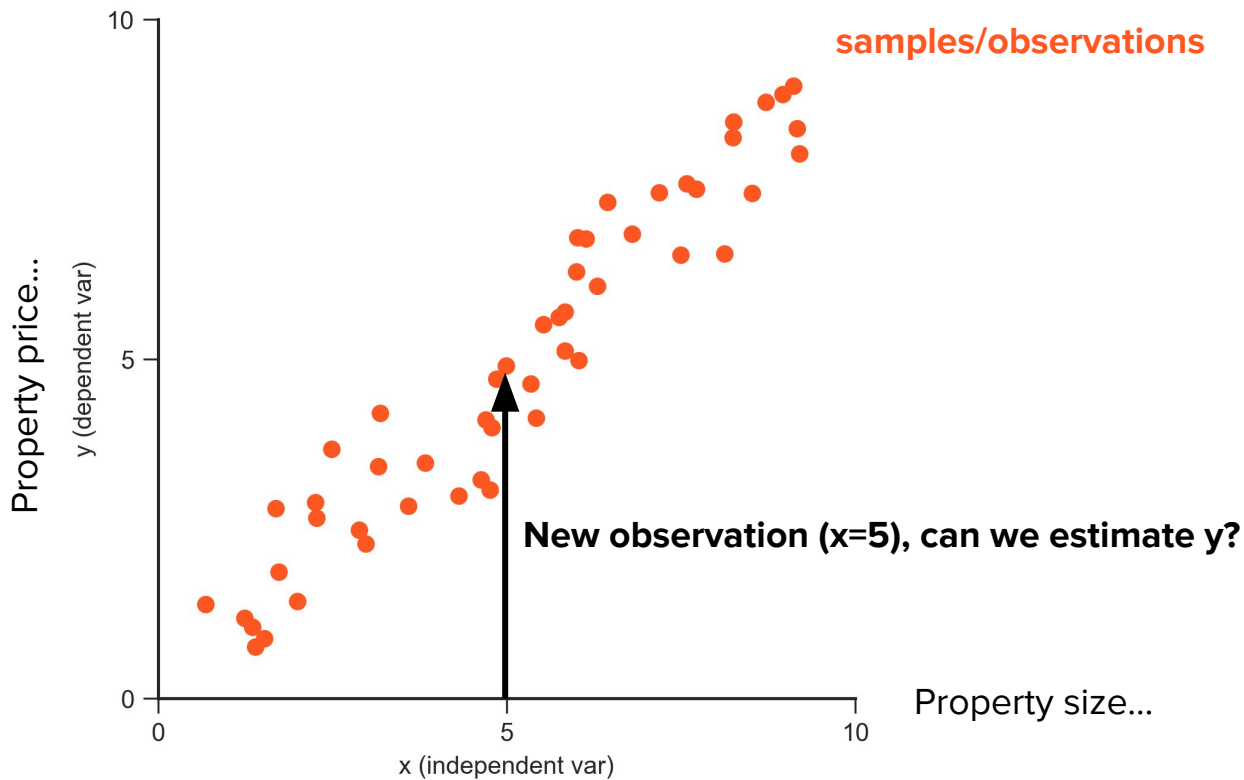
Haiku is a simple neural network library for JAX developed by some of the authors of Sonnet, a neural network library for TensorFlow.



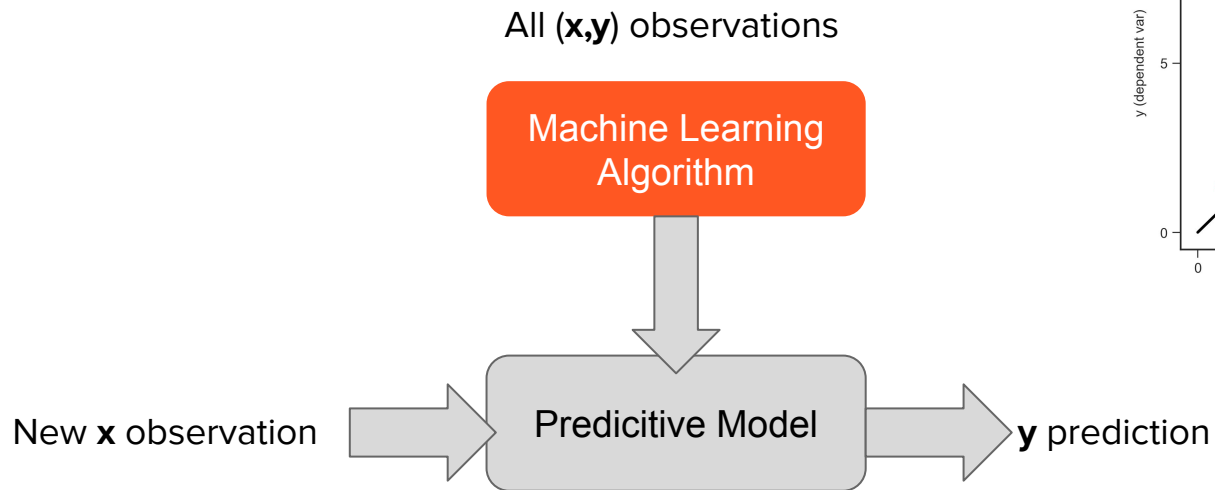
Roadmap for machine learning



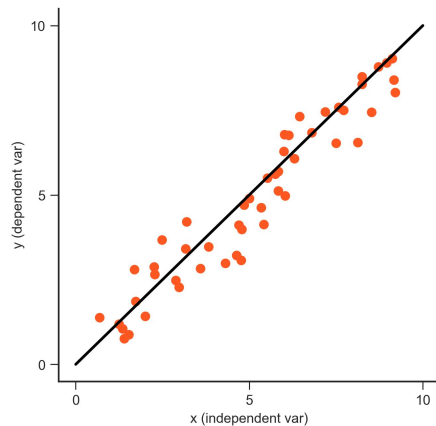
Linear regression



We need a model!

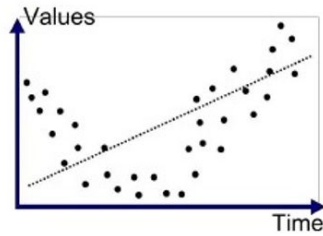


We could use a line as estimation!

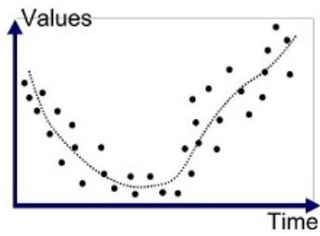


But... why a line?!

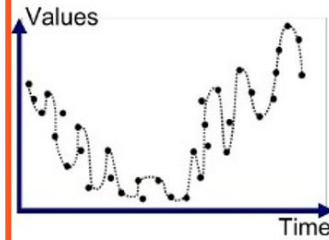
GOOD TRADE-OFF



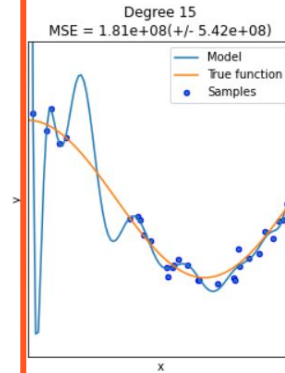
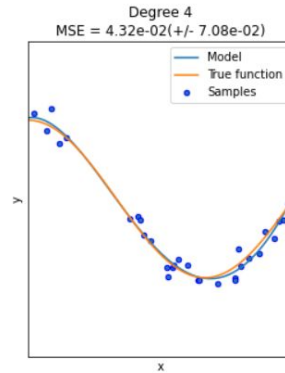
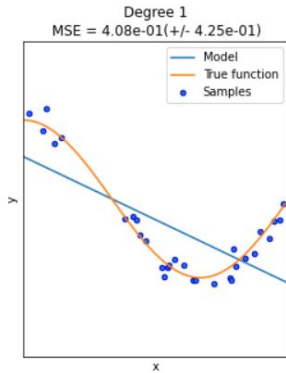
Underfitted



Good Fit/Robust



Overfitted



Linear regression

slope

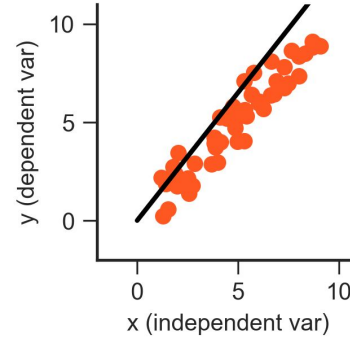
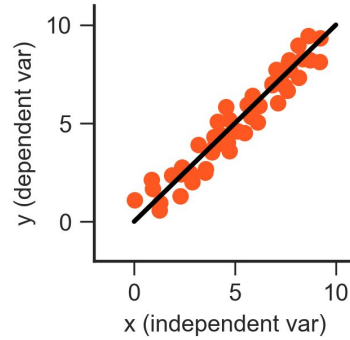
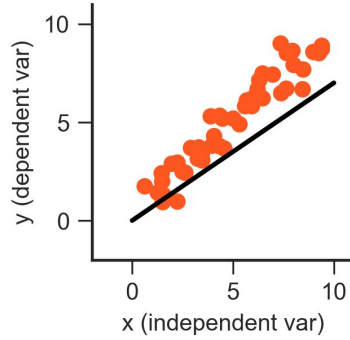
intercept

$$y = m * x + b$$

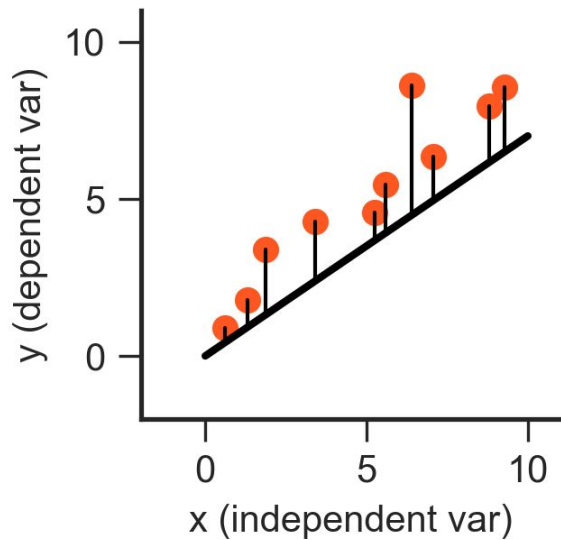
How do we find **m** and **b**?

Or, first of all, what are we aiming for?!?!

What is a good line?!

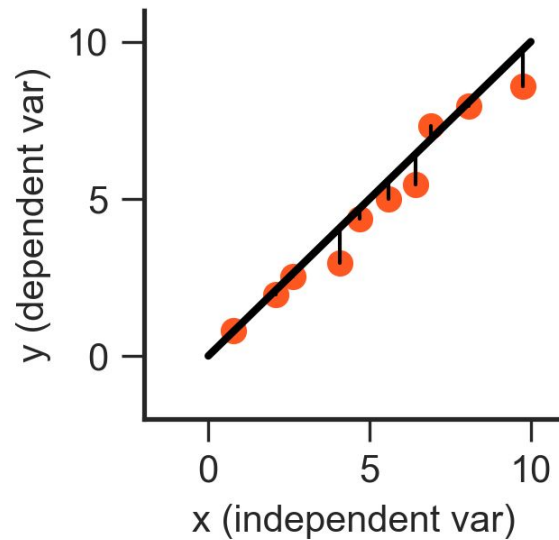


Formalizing our aim



Minimize residuals!

$$\min \sum_i^n (y_i - \hat{y}_i)^2$$



Who is minimizing your residuals?

- Optimization algorithm.

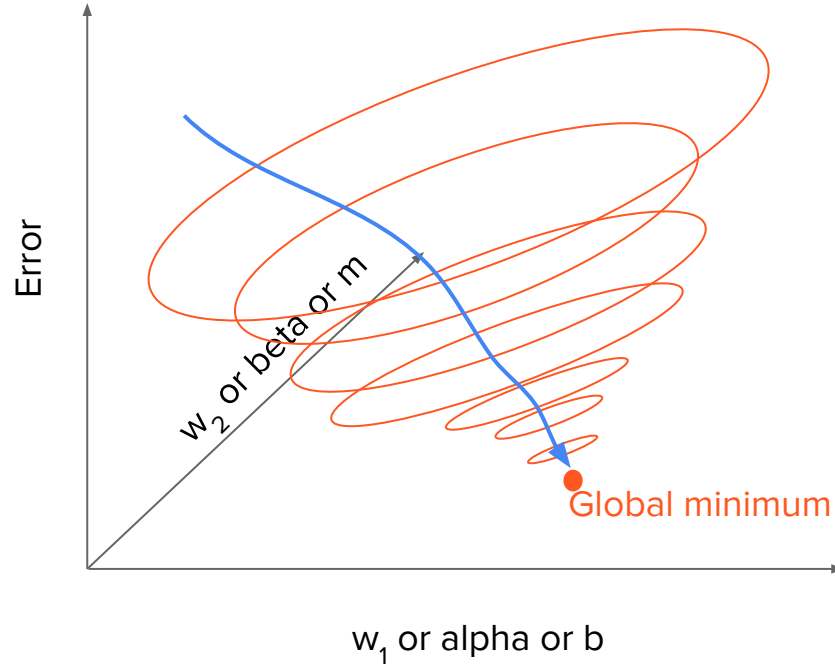
These ones you should now:

- Levenberg-Marquardt (damped least-squares, interpolation of Gauss/Newton method + grad. desc)
- Nelder-Mead (downhill simplex, heuristic for non-linear opt. w/o knowing the derivative)
- Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm
(improves a guess of the Hessian matrix, and does not use matrix inversion → $O(n^2)$ vs. $O(n^3)$)

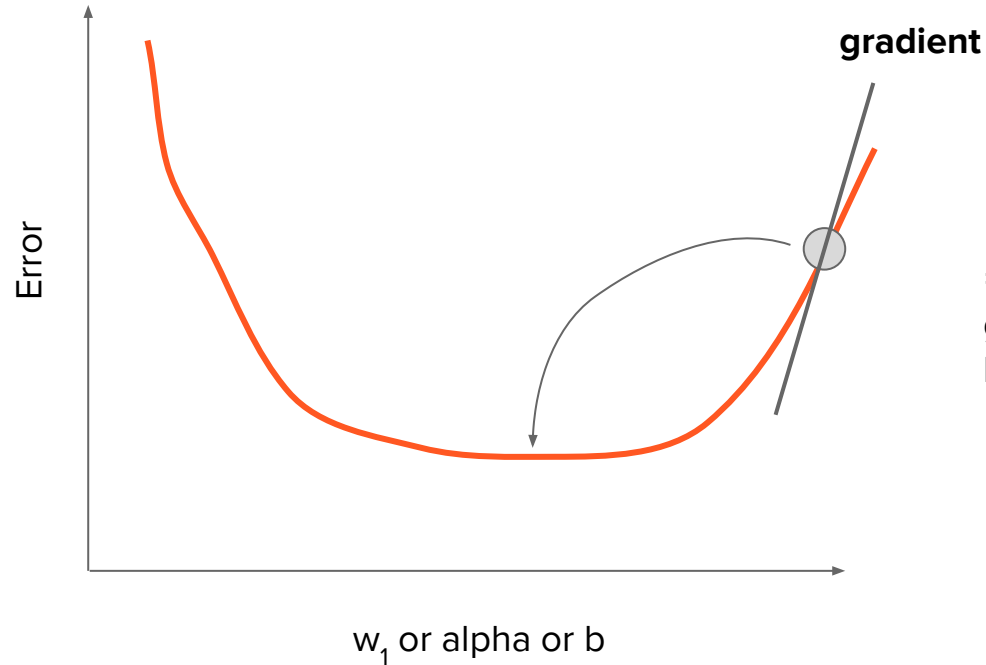
- A general method is **GRADIENT DESCENT**.

Optimization through gradient descent

$$y = m * x + b$$

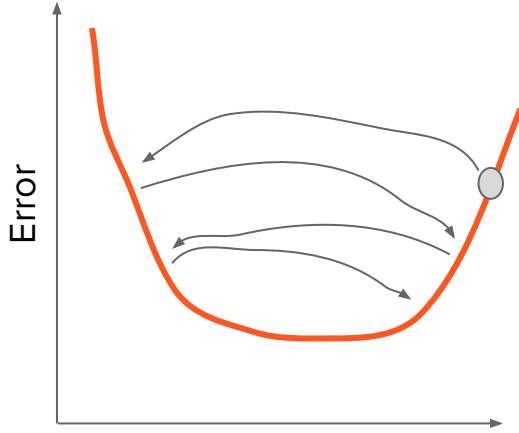


Optimization with gradient descent

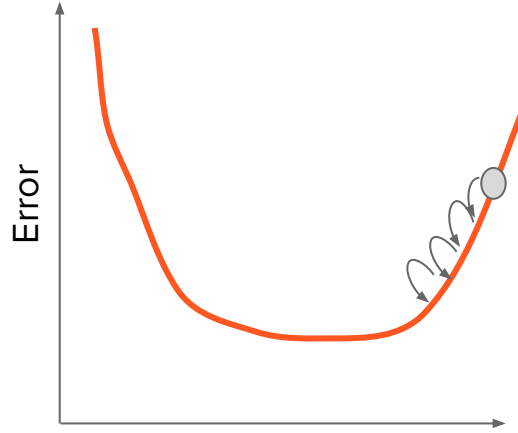


⇒ walk AGAINST the gradient → gradient DESCENT :-)

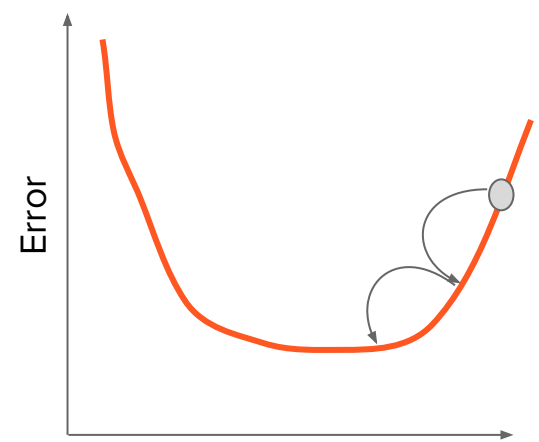
Optimization with gradient descent



Step size TOO LARGE



TOO SMALL



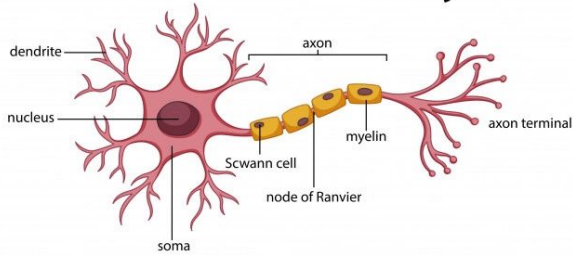
JUST RIGHT

Step size = **LEARNING RATE**

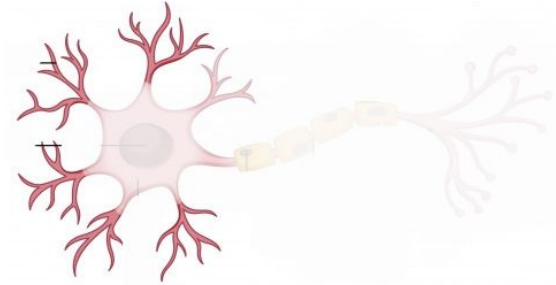
Computing the gradient



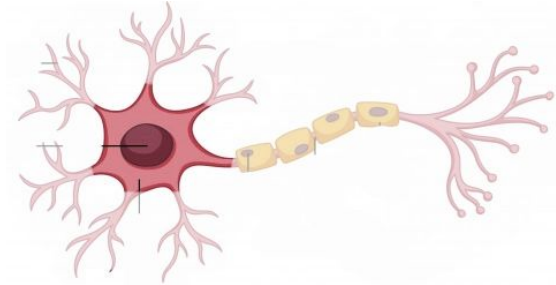
Perceptrons



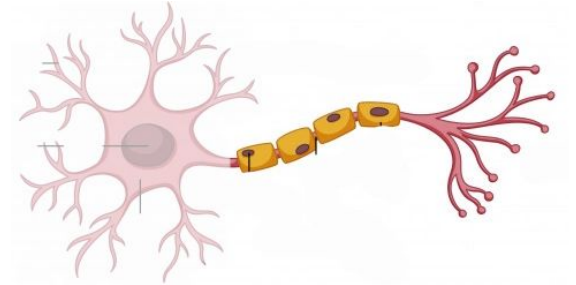
INPUT



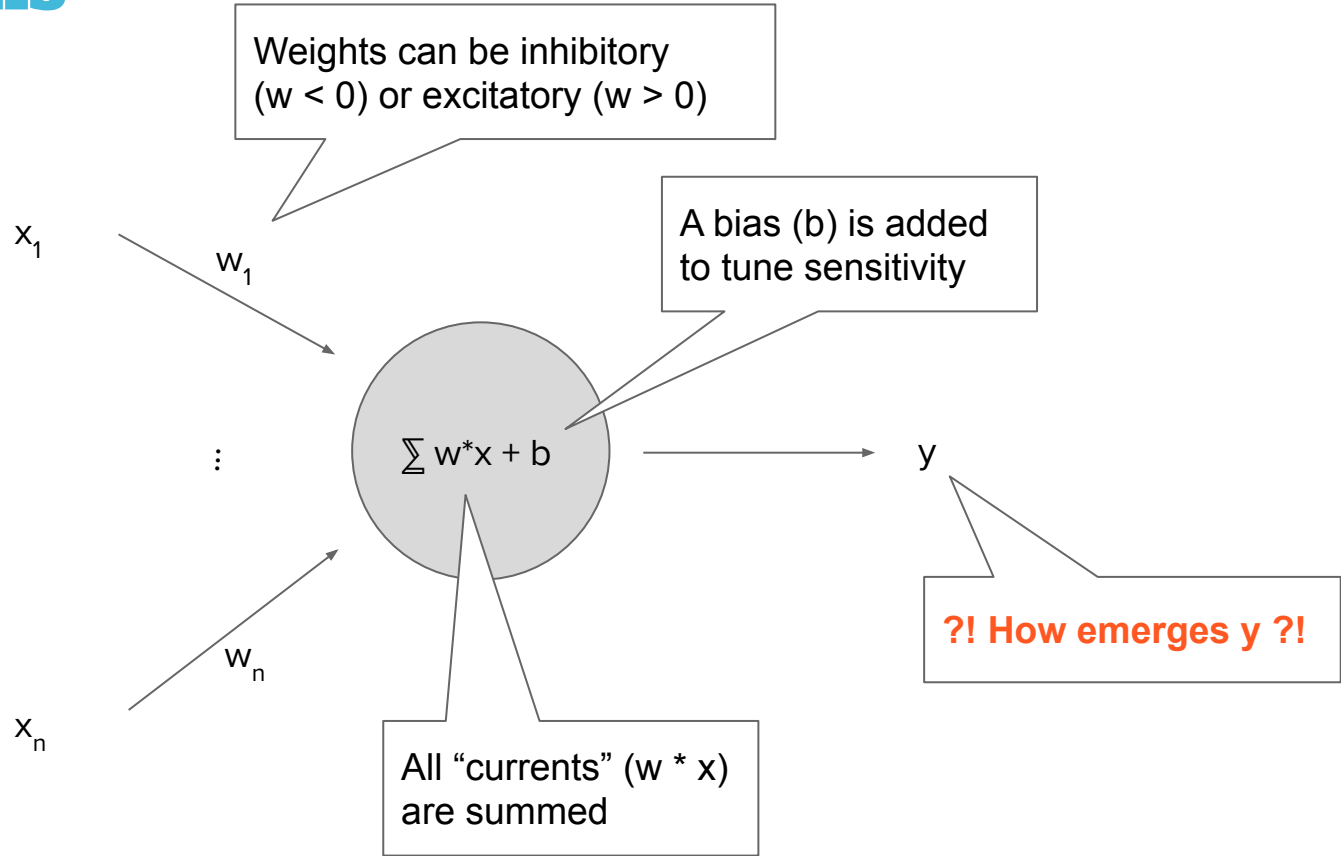
COMPUTATION



OUTPUT



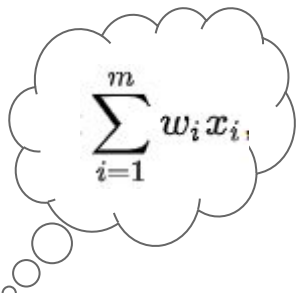
Perceptrons

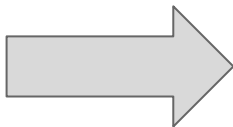


Perceptrons

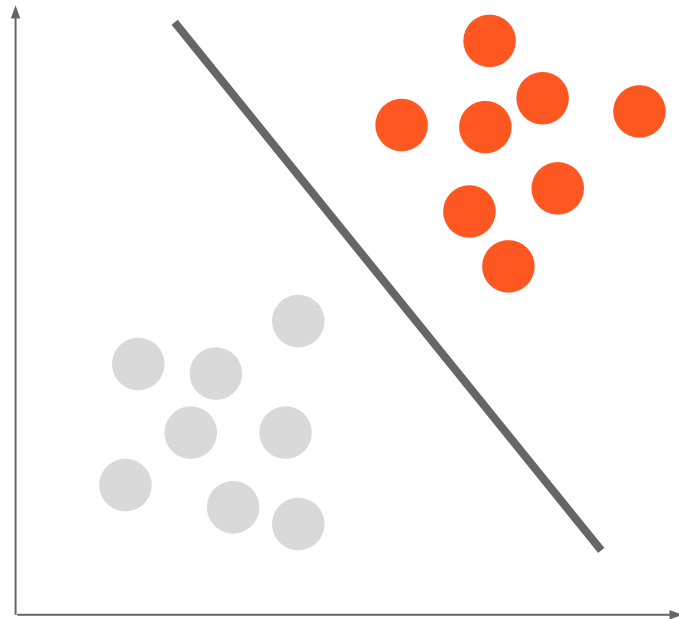
Threshold
step function:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$


$$\sum_{i=1}^m w_i x_i$$



Perceptrons are **LINEAR CLASSIFIER!**



Teaching and Learning

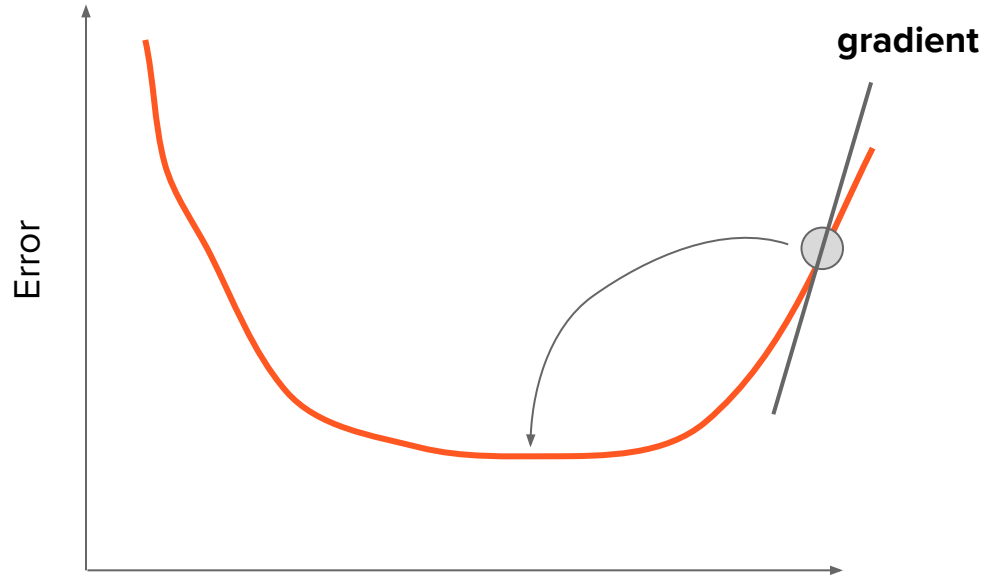
What are the parameters one could tune?

WEIGHTS

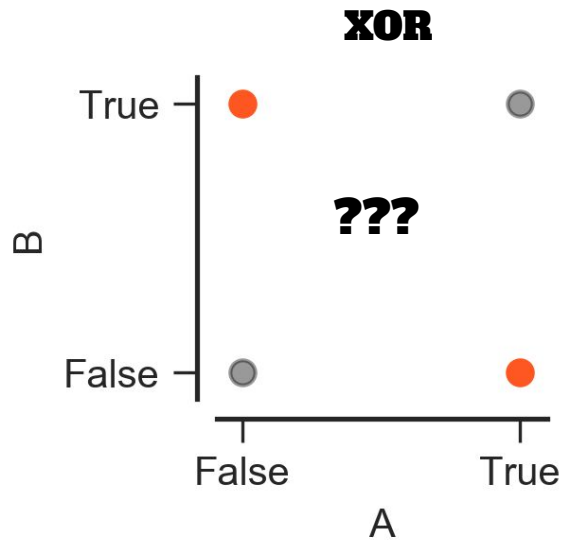
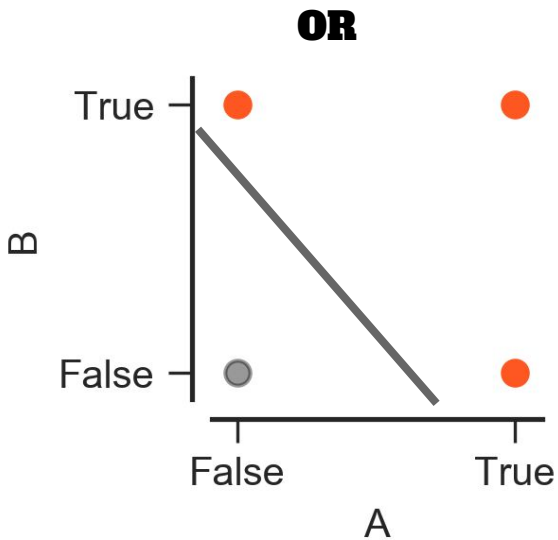
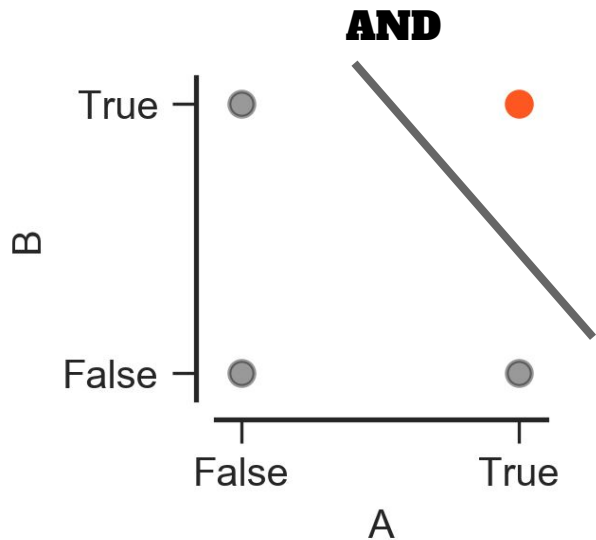
BIAS

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

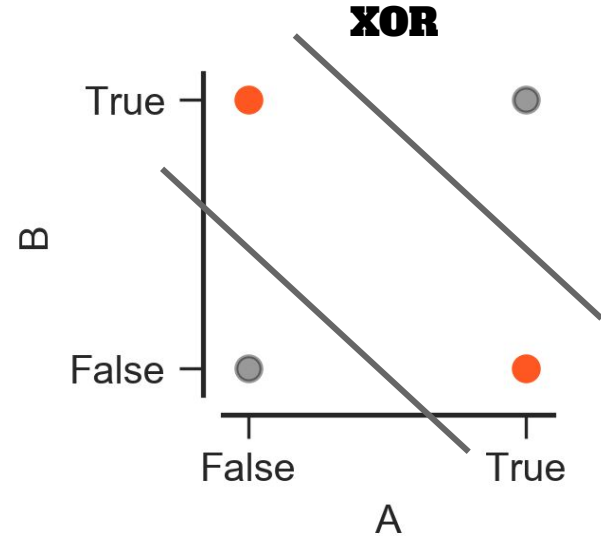
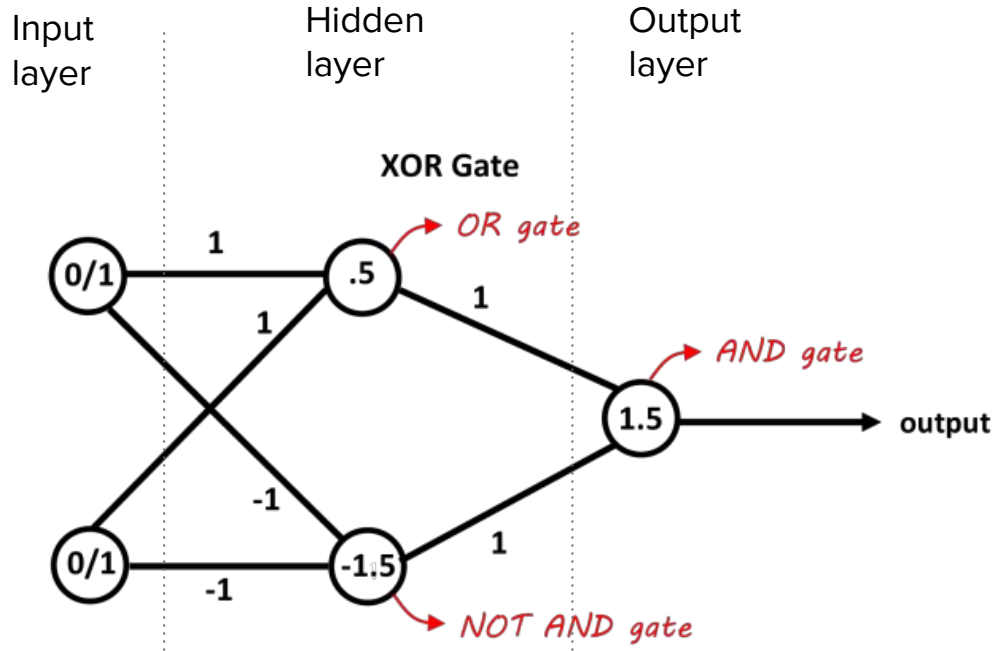
Learning weights with gradient descent



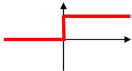
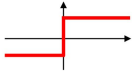
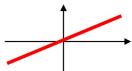

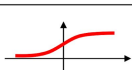



Why perceptrons were abandoned...



But there is a solution: The multilayer perceptron!

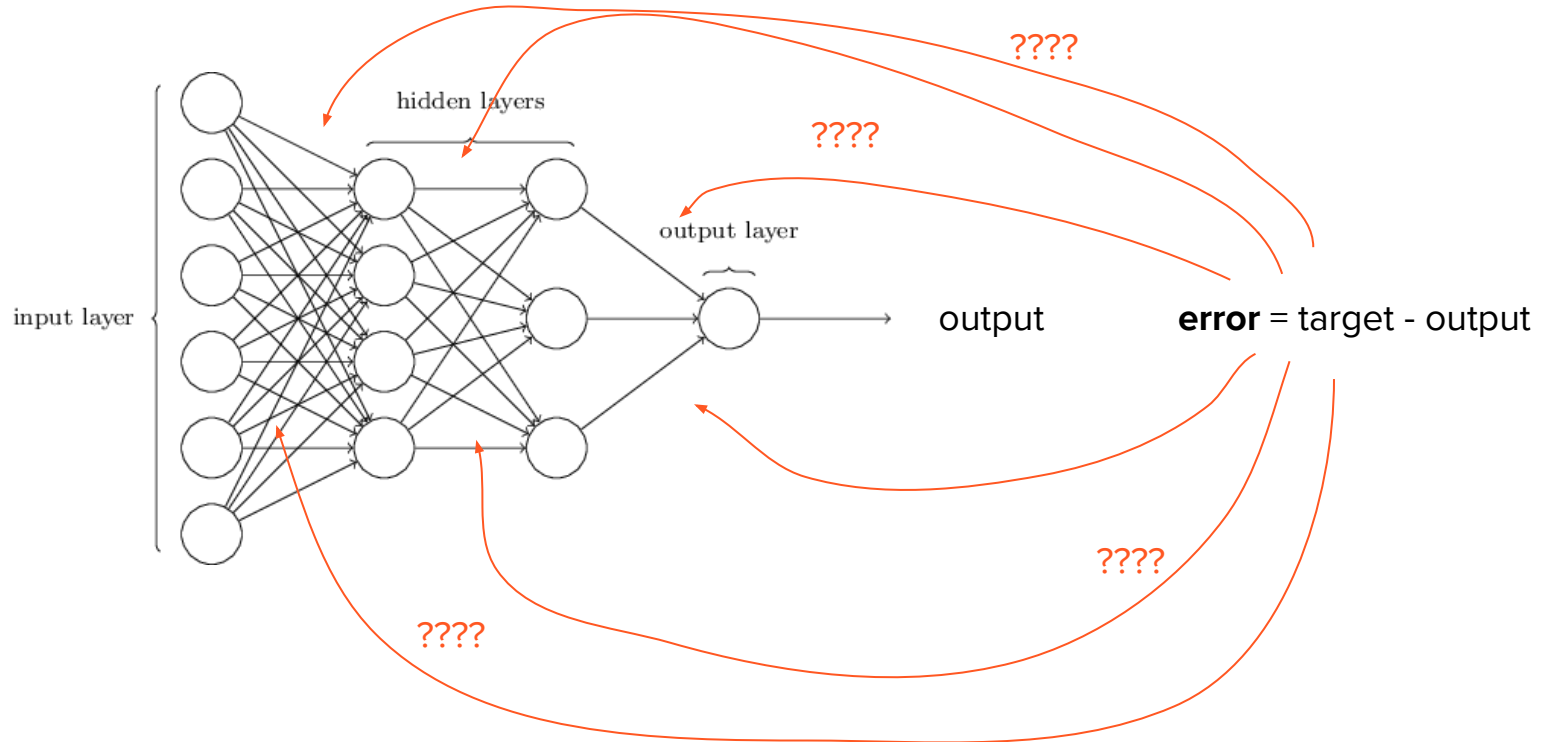


Activation functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Assigning the error

→ Something called backpropagation.
It is done automatically, but you should
KNOW WHAT IT IS DOING!



Yes you should understand backprop



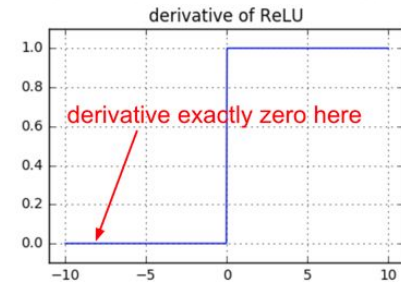
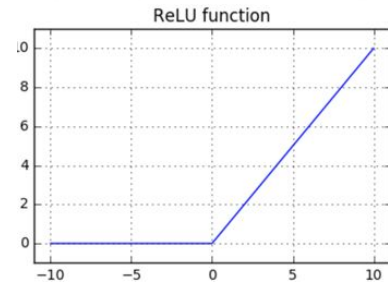
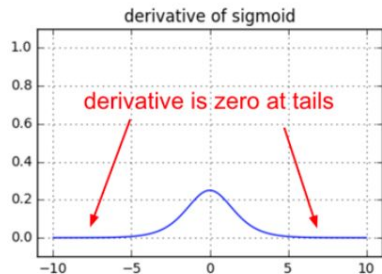
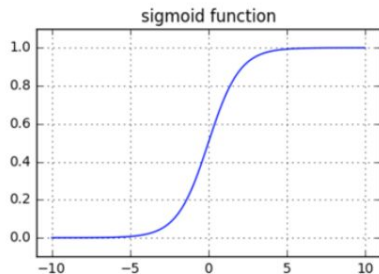
Andrej Karpathy Dec 19, 2016 · 7 min read



17K



46



In conclusion

Backpropagation is a leaky abstraction; it is a credit assignment scheme with non-trivial consequences. If you try to ignore how it works under the hood because “TensorFlow automatically makes my networks learn”, you will not be ready to wrestle with the dangers it presents, and you will be much less effective at building and debugging neural networks.

<https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>

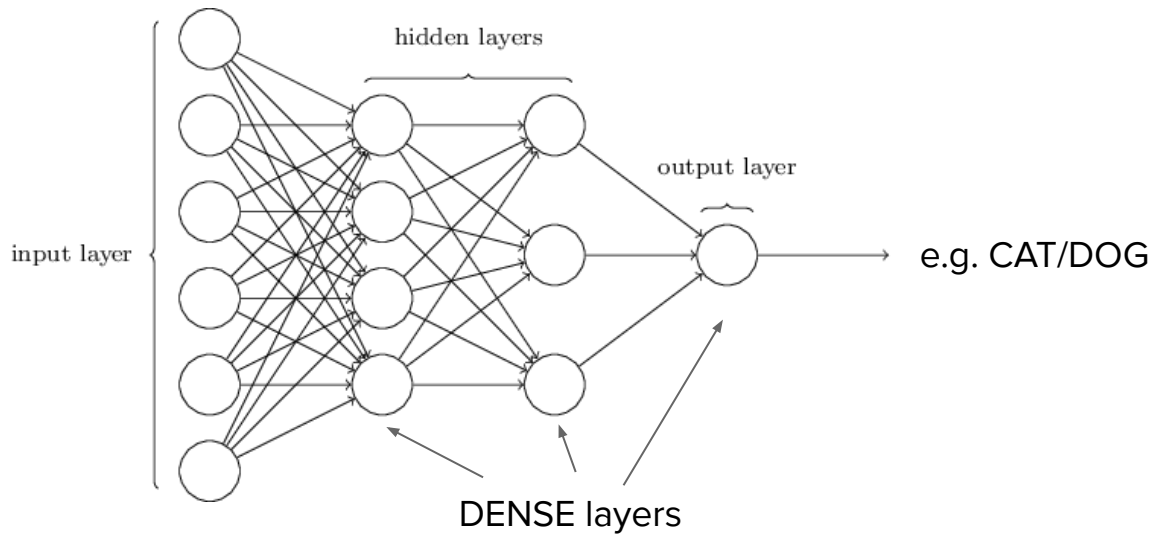
TensorFlow/Keras



1	1	0
4	2	1
0	2	1



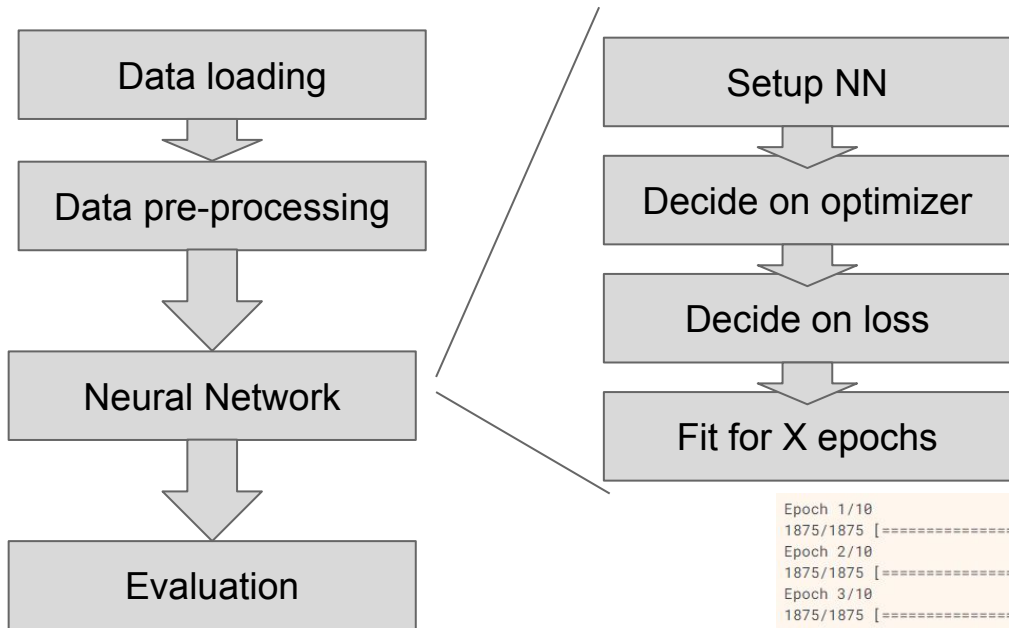
1
1
0
4
2
1
0
2
1



INPUT layer

FLATTEN layer

How to create a NN (in Keras)



```
model = tf.keras.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(128, activation='relu'),  
    tf.keras.layers.Dense(10)  
])
```

```
model.compile(optimizer='adam',  
              loss=SparseCategoricalCrossentropy(from_logits=True),  
              metrics=['accuracy'])
```

```
model.fit(train_images, train_labels, epochs=10)
```

313/313 - 1s - loss: 0.3637 - accuracy: 0.8693

Test accuracy: 0.8693000078201294

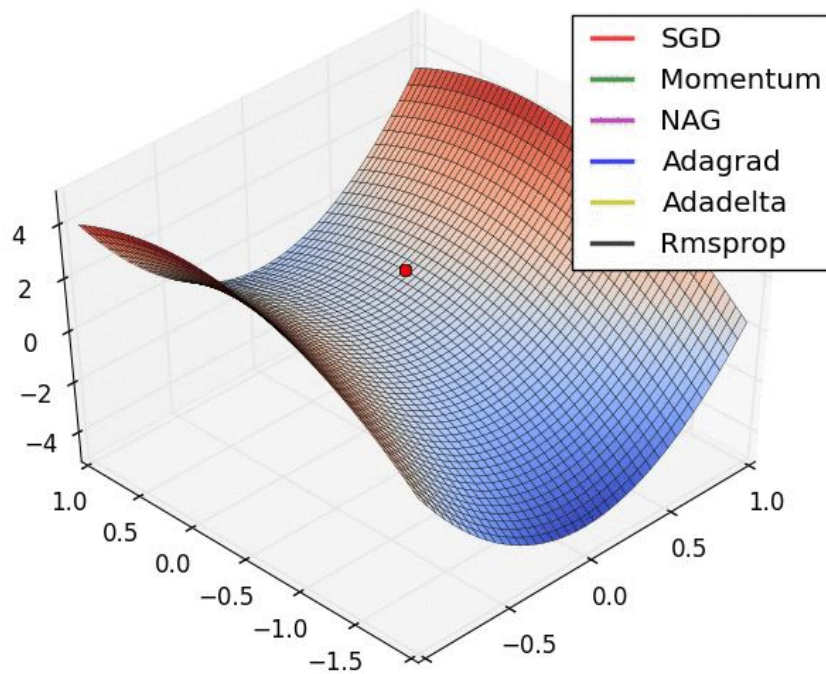
```
Epoch 1/10  
1875/1875 [=====] - 3s 1ms/step - loss: 0.4917 - accuracy: 0.8277  
Epoch 2/10  
1875/1875 [=====] - 3s 1ms/step - loss: 0.3702 - accuracy: 0.8674  
Epoch 3/10  
1875/1875 [=====] - 3s 2ms/step - loss: 0.3328 - accuracy: 0.8793  
Epoch 4/10  
1875/1875 [=====] - 3s 1ms/step - loss: 0.3106 - accuracy: 0.8859
```

Optimizers

Strategies to optimize the network and perform gradient descent:

- Stochastic gradient descent (SGD)
- SGD + momentum
- RMSprop
- Adadelata
- Adam
- ...

Try SGD(momentum=.9) and Adam



The loss function

Important:

your objective function should represent what you would like to optimize!!

Regression Losses

Mean Square Error/Quadratic Loss/L2 Loss

Mathematical formulation :-

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

Mean Squared Error

Cross Entropy Loss/Negative Log Likelihood

This is the most common setting for classification problems. Cross-entropy loss increases as the predicted probability diverges from the actual label.

Mathematical formulation :-

$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

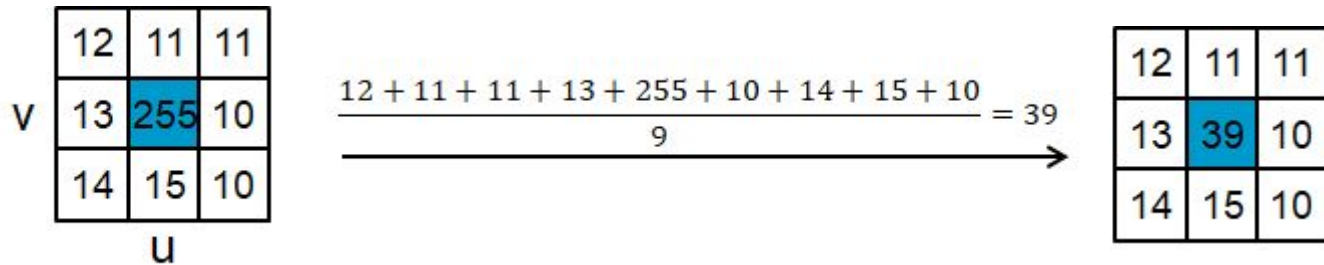
Cross entropy loss

Code for an MLP

```
model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(128, activation='relu'),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation='softmax')  
) # Multilayer perceptron
```


Changing gears...

Imagine a noisy pixel



$$I'(u, v) = \frac{\sum_{i=-1}^1 \sum_{j=-1}^1 I(u+i, v+j)}{9}$$

1	1	1
1	1	1
1	1	1

Applying a linear filter across a whole image



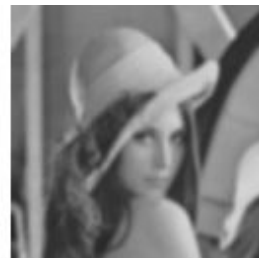
1	1	1
1	1	1
1	1	1



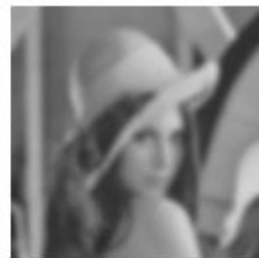
Kernel size effect



$$\otimes \square =$$



$$\otimes \square =$$



$$\otimes \square =$$



Increase of
receptive field

Edge detection linear filters

Prewitt-Filter

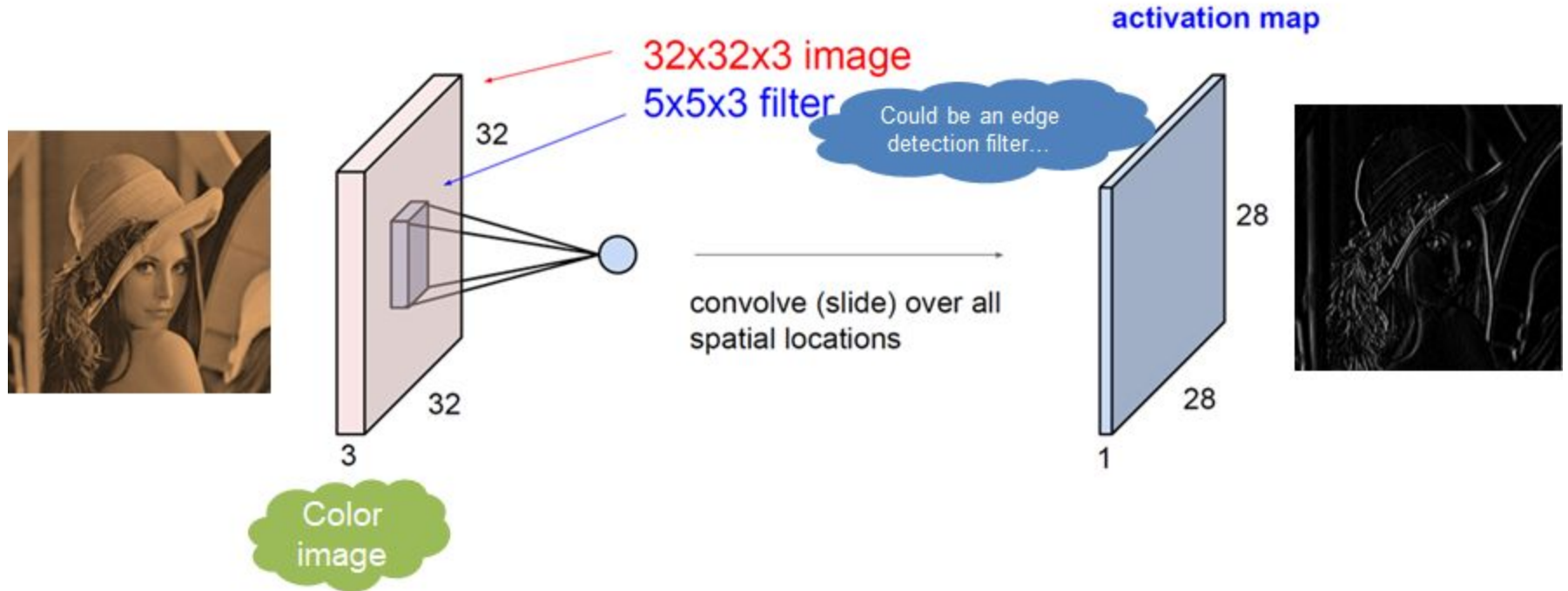
$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel-Filter

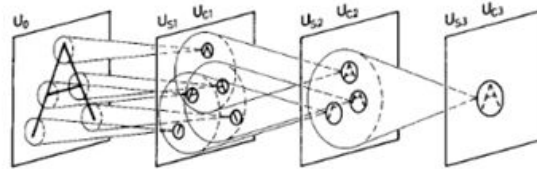
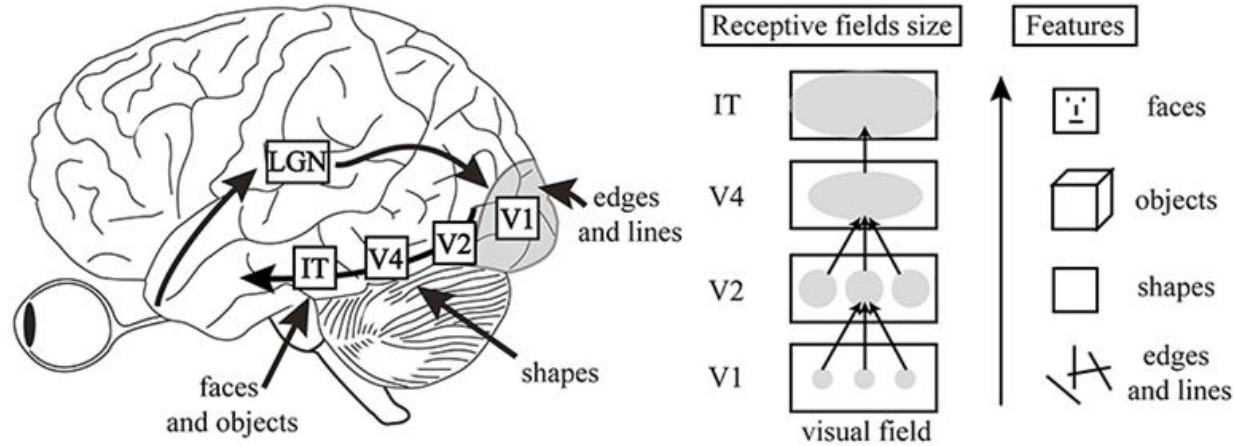
$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



Basis of convolutional neural networks



How does our visual cortex work?



Convolutional NNs (1998)

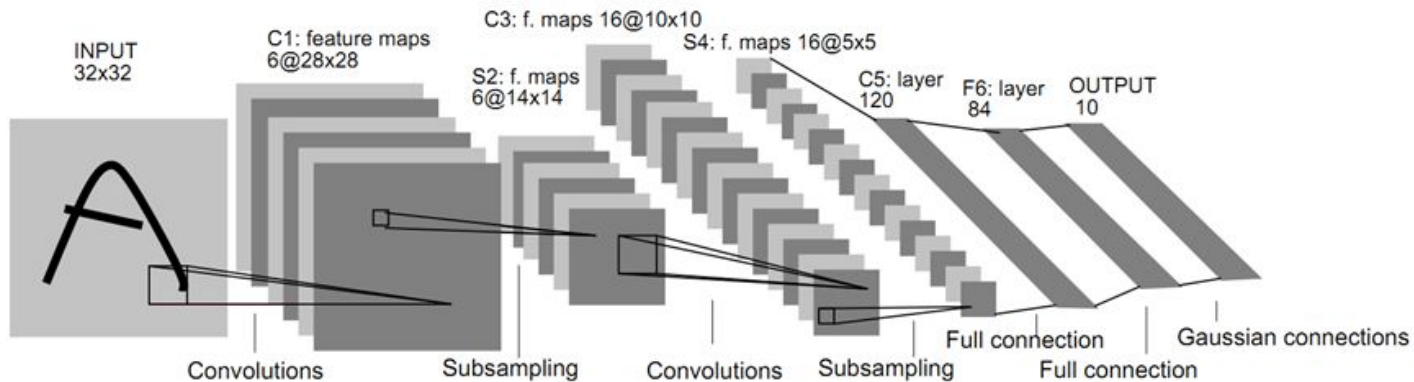


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

3 elements:

- Convolutions
- Subsamplings
- Fully Connected/Dense/Perceptron layers

Missing the last bits....

Filters/
Activation maps

$$H_x^p = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^p = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

3x3

0.81	0.32	0.07
0.75	0.28	0.81
0.53	0.88	0.34

5x5

0.07	0.64	0.66	0.86	0.05
0.28	0.06	0.89	0.86	0.51
0.64	0.07	0.06	0.36	0.03
0.04	0.31	0.24	0.57	0.06
0.87	0.46	0.07	0.71	0.86

I

```
tf.keras.layers.Conv2D(8, (3,3), input_shape=(28,28,1), padding='same', activation='relu'),
3 tf.keras.layers.MaxPool2D(),
4 tf.keras.layers.Conv2D(16, (3,3), padding='same', activation='relu'),
5 tf.keras.layers.MaxPool2D(),
6
7 tf.keras.layers.Flatten(),
8 tf.keras.layers.Dense(128, activation='relu'),
9 tf.keras.layers.Dropout(0.2),
10 tf.keras.layers.Dense(10, activation='softmax')
11 ]
```

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

2 x 2 Max-Pool

20	30
112	37

Same as in MLPs...

The last slide

- Gradient descent is a common principle for adjusting weights in a model
- Artificial neural networks are loosely related to real, biological networks
- Multilayer perceptrons (MLPs) are the classic ANNs consisting of simple neural units (perceptrons)
- Convolutional neural networks are loosely related to our visual cortex and are superior in image processing (in many tasks)

Exercise

Description of the exercise

In this exercise, we will review the Linear/Logistic regression and Correlation. Additionally, we will have an introduction to Machine and Deep Learning.

As an example, we provide an introduction with a sigmoid function using **Scipy.optimize**. Then, your task is to use **Scikit-learn** and **curve_fit/minimize** to execute a linear regression.



MLPs and CNNs

You need to get MLPs and CNNs working.

In particular, we will ask you the following:

- What do some of the arguments of the functions represent?
- How does it affect the way we fit our model?
- How can we obtain a higher performance?



These are some of the questions that you will ponder during this exercise, and programming errors that you need to solve