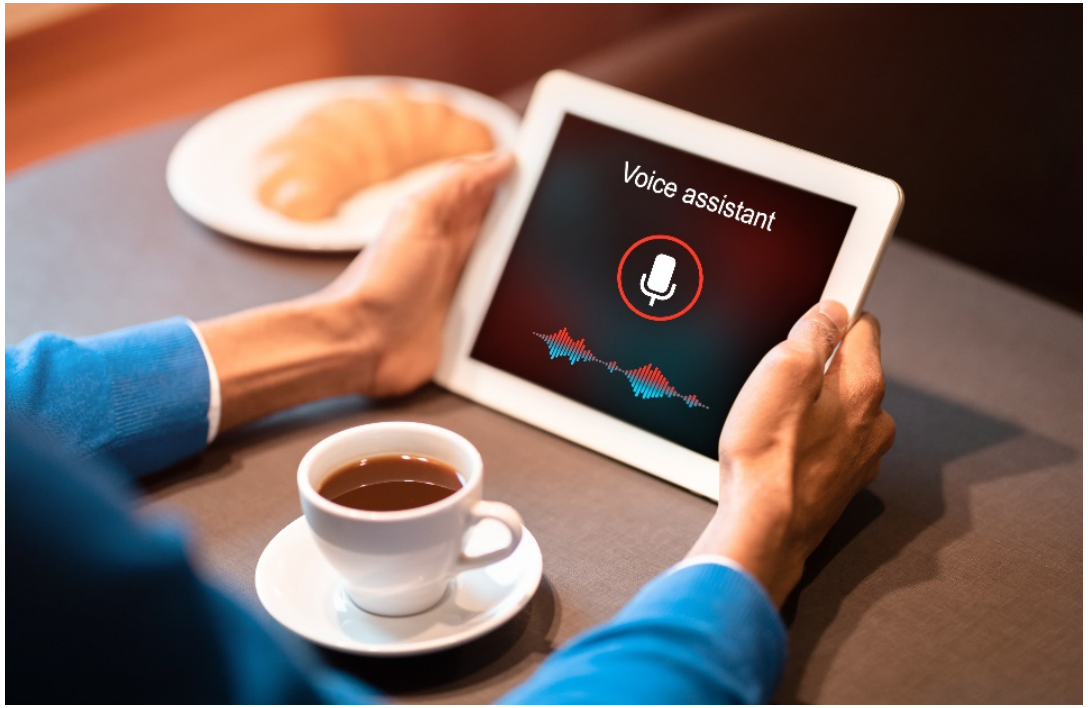
ML school audio track

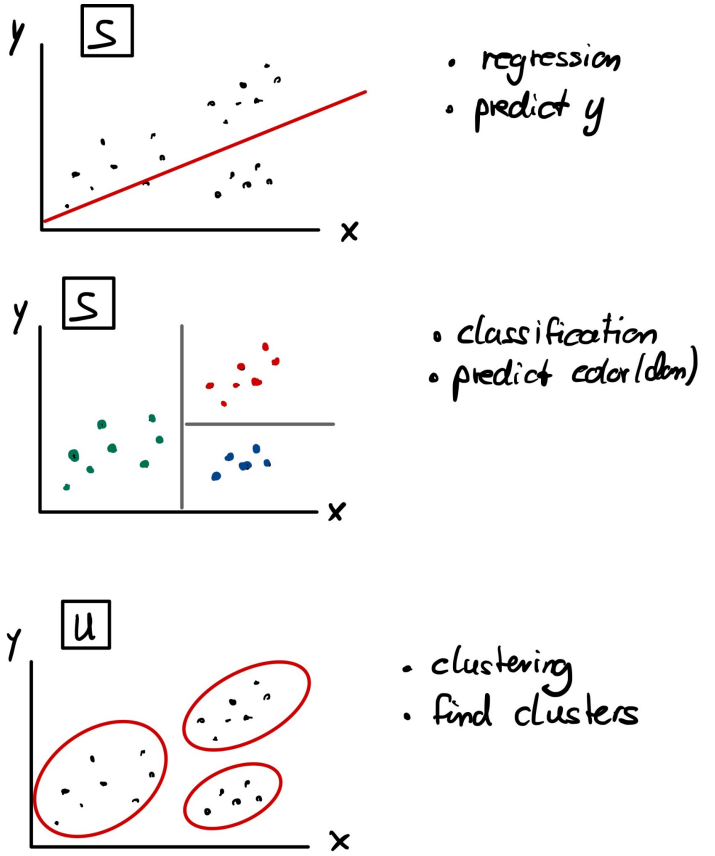
Deep Learning I

Dr. Paul Wallbott, Fraunhofer IAIS



Machine Learning

- Make predictions based on patterns found in data
- Often grouped by the availability of labels
  - Supervised learning (similar to function fitting)
    - Regression: labels continuous
    - Classification: labels discrete
  - Unsupervised learning
    - E.g., clustering



Mini Exercise

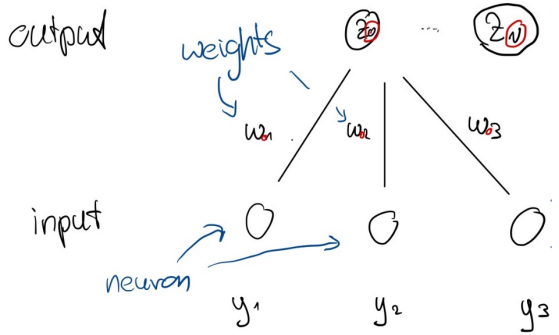
- Which type of ML problem are the following 3 examples?
  - Predict housing price based on historical transactions!
  - Which letter is written on the image?
  - Suggest user a video with similar content!
- How would you approach keyword spotting?

Machine Learning problems

- Answers
  - Predict housing price based on transactions --> find price function
  - Which letter is written on the image? --> classify image into groups
  - Suggest user a video with similar content --> find clusters of videos
- One possibility
  - Classify words

Deep Neural Networks

- One type of machine learning model
- Model approximates dataset
- Structured in layers (see scatch)
- Many layers make up a deep neural network



biases

$$z_j = \sum_k w_{jk}\, y_k^{in} + b_j$$

↓ here

$$z = \sum w_k\, y_k^{in} + b$$

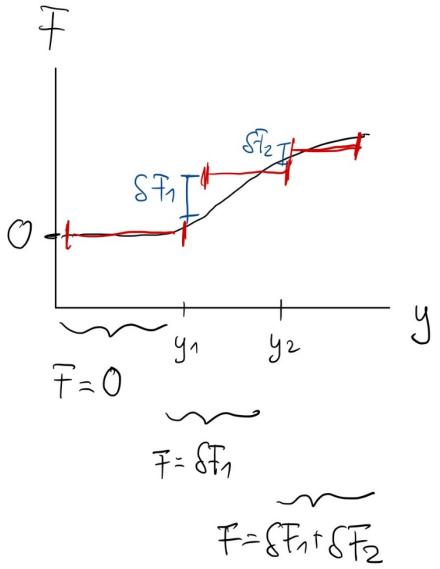
$$= w_1 y_1 + w_2 y_2 + w_3 y_3 + b$$

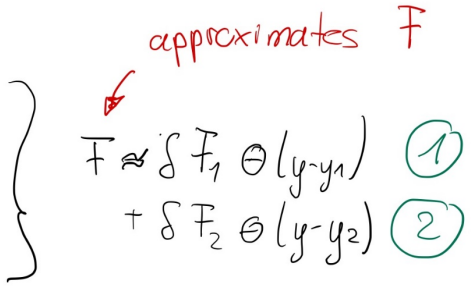
$$y^{out} = f(z)$$

activation function

Deep Neural Networks

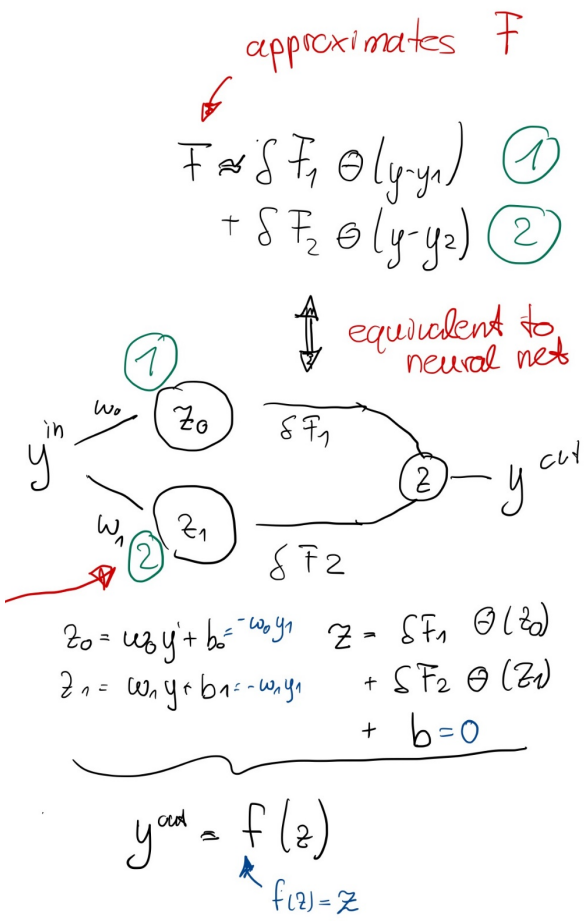
- Great approximators of non-linear behaviour
  - How do neural networks approximate?





Deep Neural Networks

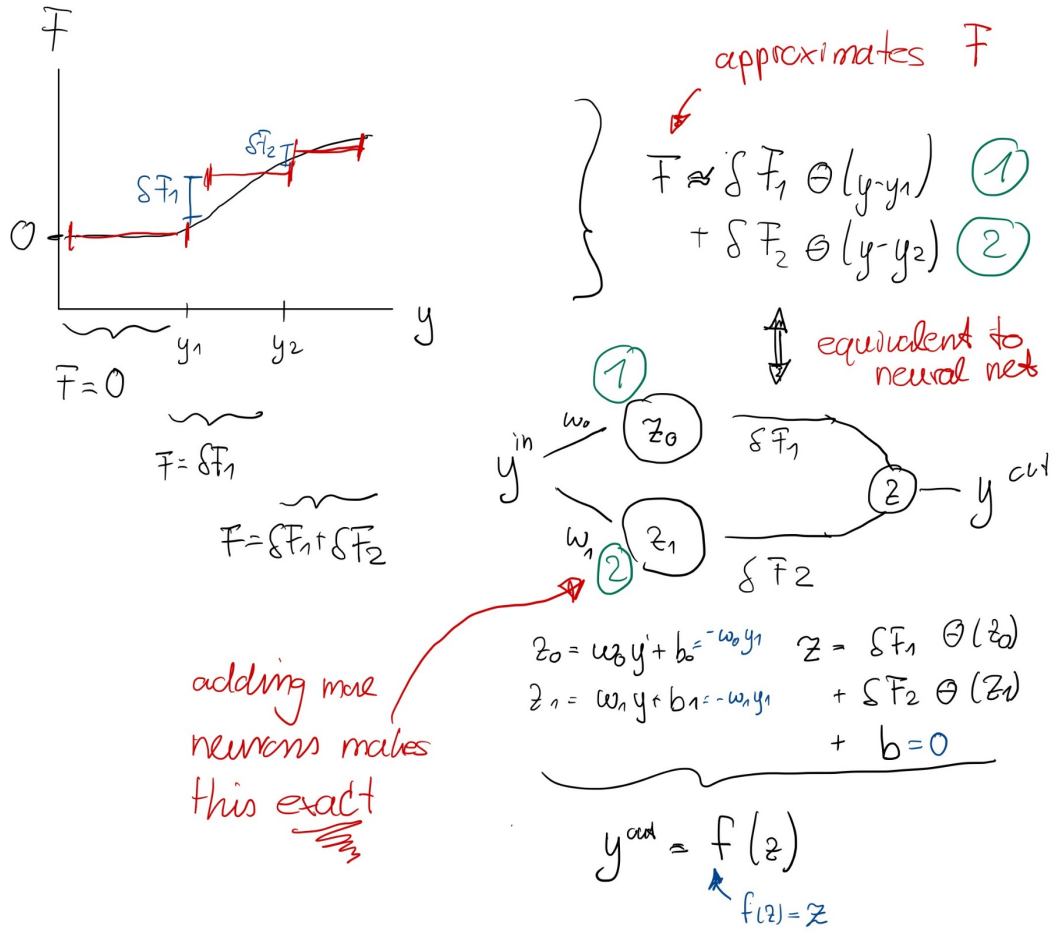
- Great approximators of non-linear behaviour
  - How do neural networks approximate?



Deep Neural Networks

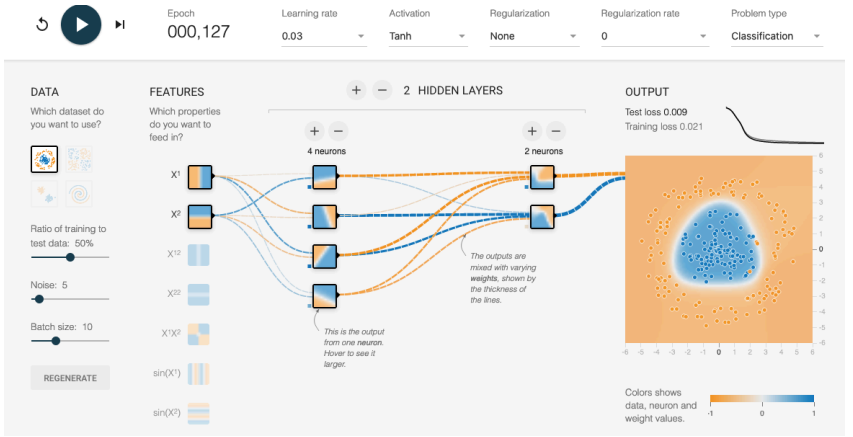
Frei nach Machine Learning for physicists 2020, Lecture Series by Florian Marquardt

- Great approximators of non-linear behaviour
  - How do neural networks approximate?



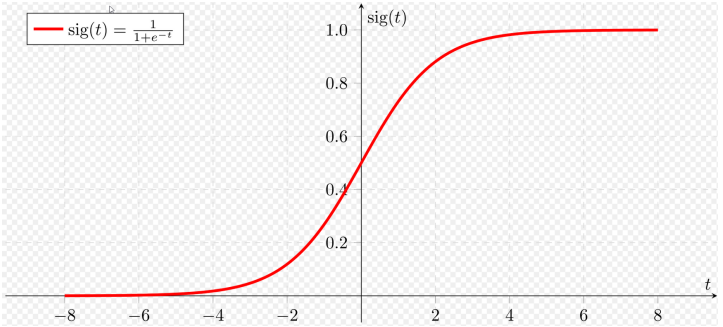
Neural networks

- We have just encountered a simple “feed forward” (FF) neural network
- It is called “deep” if it has many layers
- Prominent variants are:
  - Convolutional NNs or CNNs (later in depth)
  - Recurrent NNs or RNNs
- Let us see ourselves if they are good function approximators
  - [Playground]

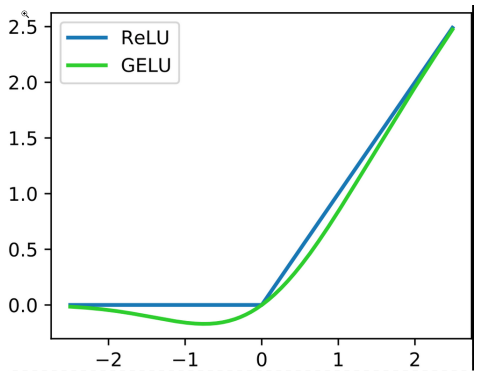


Activation functions

- Step function derivative: 0 or undefined…
  - Better use sigmoid / tanh
  - Or relu (better training properties)



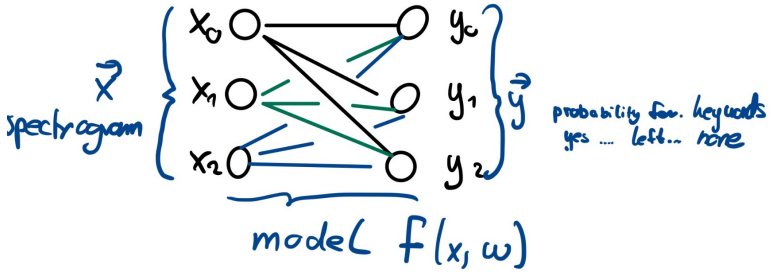
https://de.wikipedia.org/wiki/Sigmoidfunktion#/media/Datei:Sigmoid-function-2.svg



https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#Gaussian_Error_Linear_Unit_(GELU)

Components Deep Learning problem

- **Model**: Function f(x,w) mapping input x to output y with weights w
  - For now: Feed forward neural network
- **Data set**: (x,y) of **inputs** x with **labels** y
- **Loss**: L(x,y,f(x,w)) Function that describes how well the model describes the data
- **Goal**: Find w, such that loss is **optimal**
  - Not a trivial task
  - How can we find an optimum?



$$\{(\vec{x}^{(0)}, \vec{y}^{(0)}) \dots (\vec{x}^{(N-1)}, \vec{y}^{(N-1)})\}$$

dataset with N instances

all instances

prediction

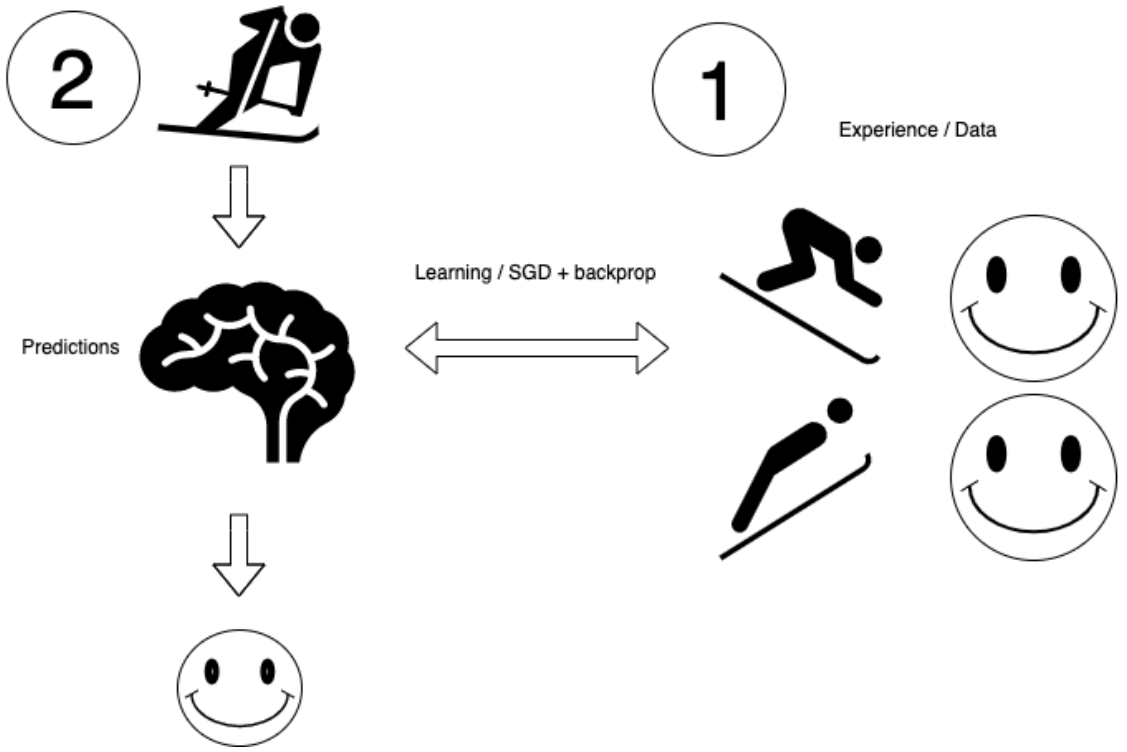
label of instance

$$L \sim \sum_{i=0}^{N-1} \left(f(\vec{x}^{(i)}, w) - \vec{y}^{(i)}\right)^2$$

distance

model parameters / weights

Components Deep Learning problem



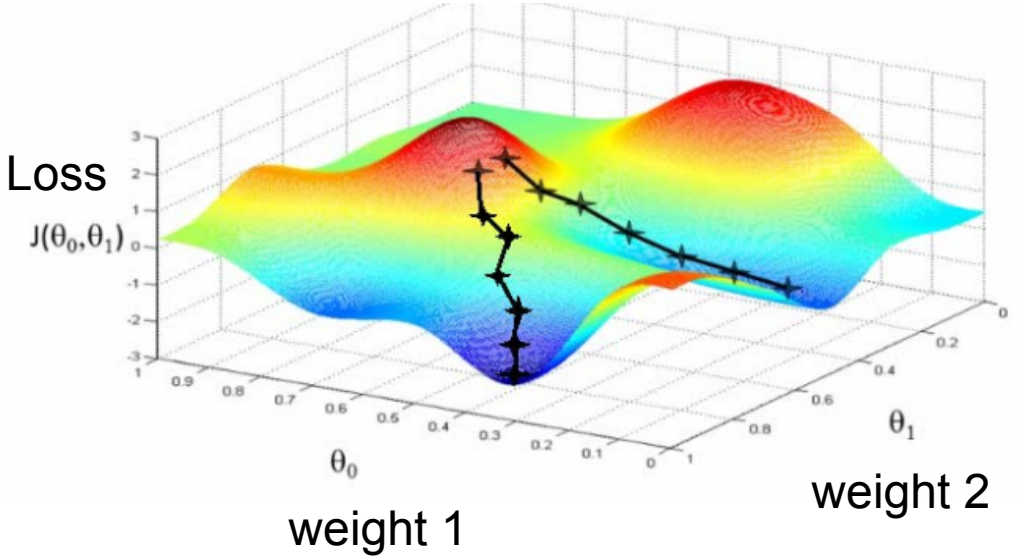
How to get the weights?

- Optimize model with respect to parameters w (weights)
- Constraint: Minimize loss function
- Scalar function in the space of parameters: dimension?
- Gradient descent! (GD)
  - “Ski down hill” aka in gradient direction
  - Need derivatives of L(x,w,f) and therefore f(x,w)

$$\frac{\partial L(x,w)}{\partial w} \sim R(x,w) \frac{\partial f(x,w)}{\partial w}$$

(V)

gradient w.r.t w — chain rule — model derivative

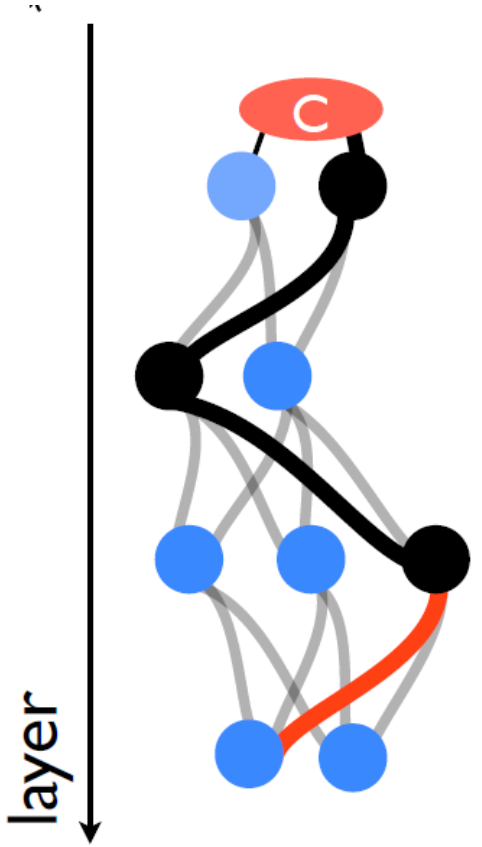


Source: Fraunhofer ML Ergebnisbericht

Back propagation

- NN is essentially chain of function applications
  - Sum over all paths effecting output y
  - Multiply derivatives along a path (chain rule)
- Doing this efficiently: backpropagation
- Intuition enough for now
- This made Deep Learning scale!

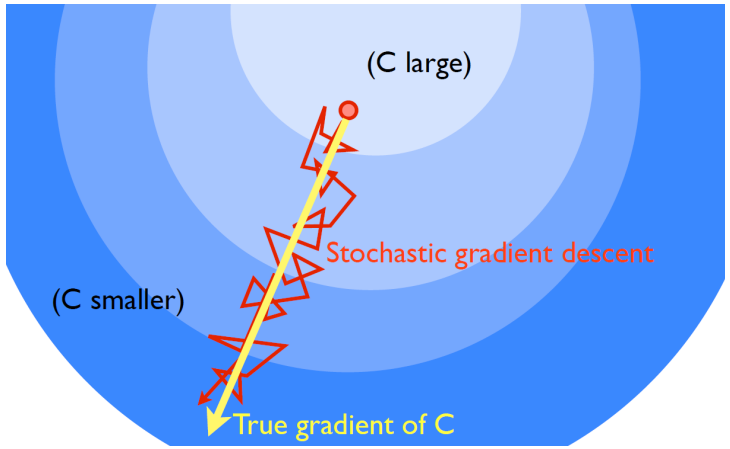
*Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986a). "Learning representations by back-propagating errors". Nature. **323** (6088): 533–536*



Machine Learning for physicists 2020, Lecture Series by Florian Marquardt

How to get the weights?

- Still often to big to calculate the gradient for all samples at the same time
- Stochastic gradient descent (SGD): Use mini batches aka “batches”
- On average go in gradient direction, if
  - Learning rate (eta) has right size
  - w —> w – eta dL/dw
- Smart strategy improvement
  - ADAM: smarter updating strategy adaptive method
  - More exist: https://arxiv.org/pdf/1609.04747.pdf



Machine Learning for physicists 2020, Lecture Series by Florian Marquardt

Mini Exercise

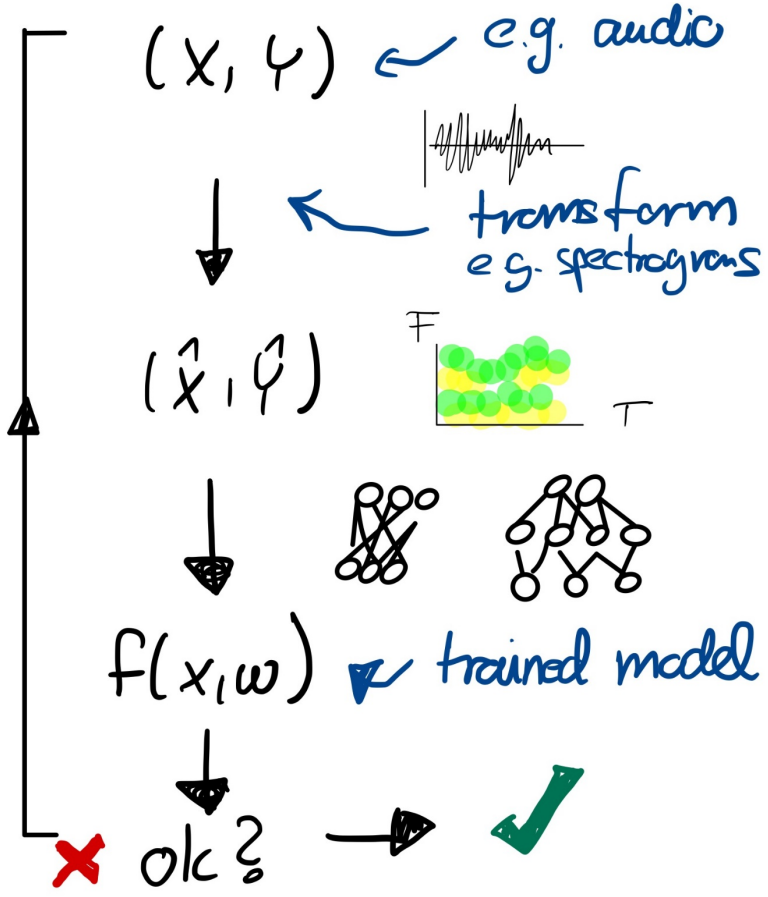
- What are the key components in a deep learning problem?
- How is the optimisation problem solved?
- What is the relationship between SGD and backpropagation?

Mini Exercise

- What are the key components in a deep learning problem?
  - Model, dataset, loss function
- How is the optimisation problem solved?
  - SGD
- What is the relationship between SGD and backpropagation?
  - SGD is a method to find the minimum which needs to calculate gradients
  - That is expensive (chain rule) but backpropagation helps

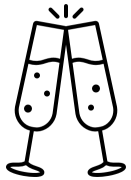
Machine Learning

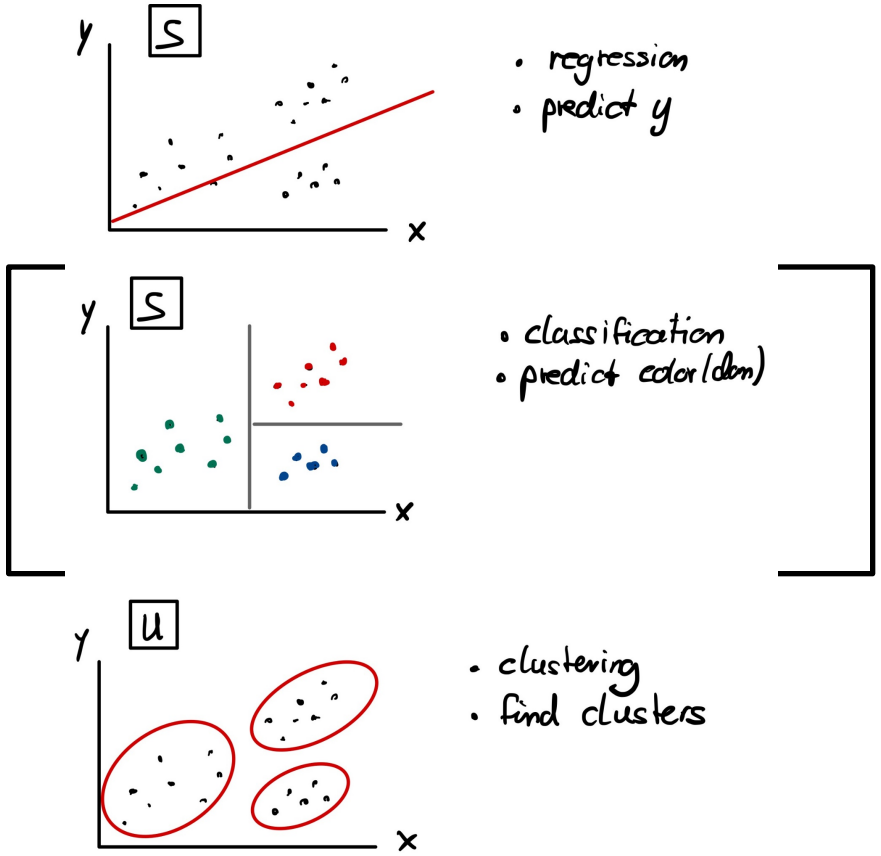
- Practical steps
  - Data gathering, pre-processing
  - Feature engineering: What are good features?
  - Training of several (deep neural) **models**
  - Evaluation on new data --> will see more systematic ways later



Let’s put our knowledge to use!

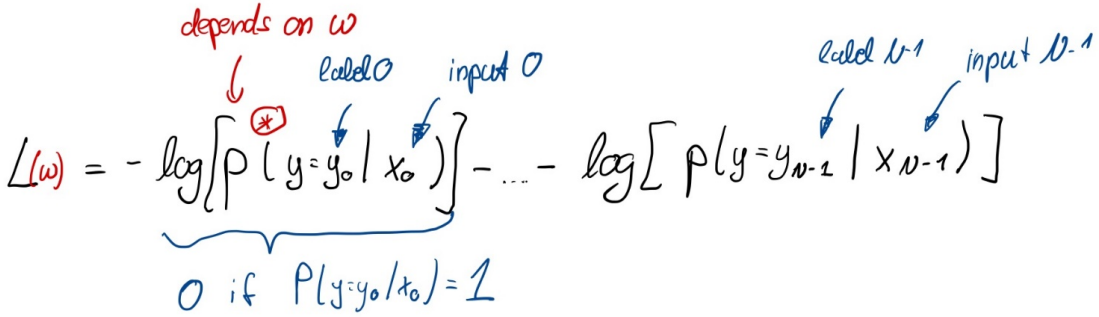
- We know theoretically how DL works now!
- Let us look at **classification**
  - Useful for keyword spotting later
  - Used very often in practice (e.g., medical diagnosis)





Logistic Regression

- Classification algorithm: Instances are grouped into *previously known* classes
- As a deep learning problem
  - **Model** f(x,w): What is the likelihood of word y given recording x?
  - **Dataset**: 1 second recordings x, class labels y=[1, 0, .., 0], "*one-hot encoded": which word was said?*
  - P(y|x) = softmax(w.x + b): logistic regression
  - **Loss**: Categorical entropy. Log likelihood of the trainingset being classified correctly



Metrics for classification

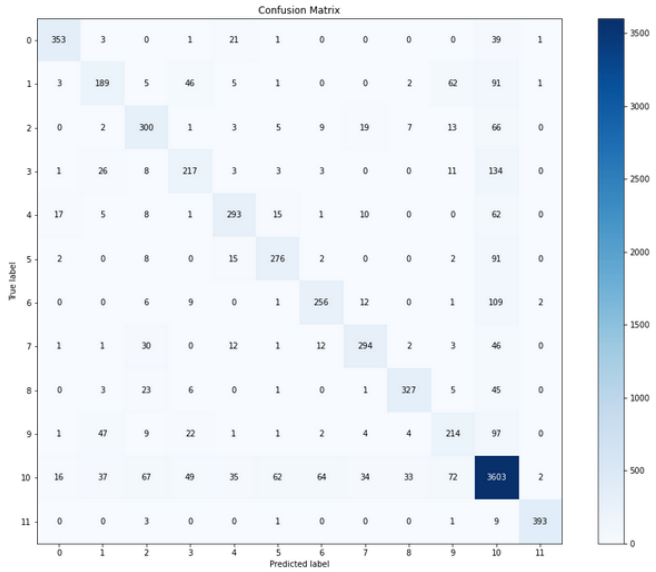
- Translate loss to real performance?
- Need proper metrics
  - Acc = (TP + TN) / (P + N) [0,1]
  - Recall = TP / P [0,1]
  - Precision = TP / PP [0,1]
  - F1 = 2 TP / (2 TP + FP + FN)
- Equivalent measures FPR, FNR
- Tradeoff Precision Recall (FPR, FNR)
  - Can be balanced by thresholds

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

https://en.wikipedia.org/wiki/Confusion_matrix

Metrics for classification

- This is called confusion matrix
- We will plot it later for multiple classes



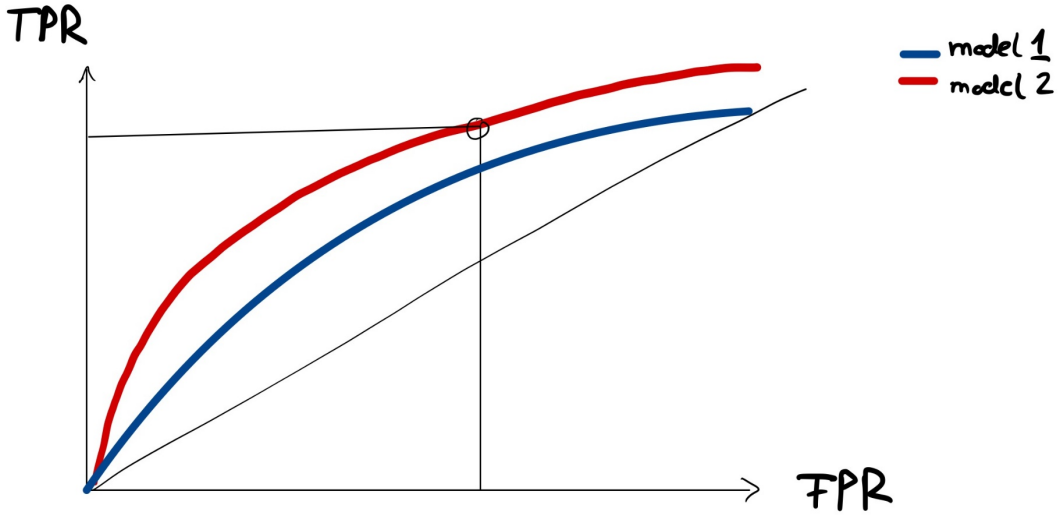
Multi class

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

binary

ROC curves

- Receiver operating characteristics
  - Visualize tradeoff between metrics
  - Example here: TPR, FPR
- Requirements decide which metrics matter
  - E.g. Covid test —> high TPR
  - Choose parameter such that TPR high enough vs. FPR low as possible
- Models can be compared by their ROC curve
  - Which one is better here?



?

Mini exercise

- Discuss with your neighbour / group:
  - Recap what the confusion matrix of a keyword spotter can tell you
  - For a keyword spotter, which two metrics would you use to judge the error and performance tradeoff? Which one should be optimized to what value?
  - Experts: Show that the softmax function automatically fulfils the criterion that the probability of being in one of the 2 classes from our example is one (100%)

Mini exercise

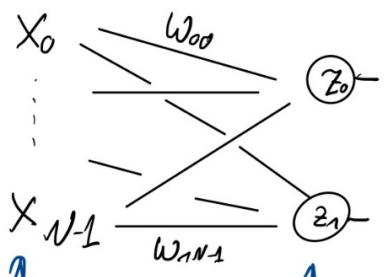
- Recap what the confusion matrix of a keyword spotter can tell you
  - Which words are confused with each other, e.g. “go” and “no”

Mini exercise

- For a keyword spotter, which two metrics would you use to judge the error and performance tradeoff? Which one should be optimized to what value?
  - False positive rate —> very annoying
  - False negative rate —> also very annoying

https://pixabay.com/de/photos/w%c3%bctend-ver%c3%a4rgern-person-frau-2514031/

Solution



N input values (pixels)

2 classes 0, 1

$$z_0 = \sum w_{0k} x_k + b^0_k$$

$$z_1 = \sum w_{1k} x_k + b^1_k$$

$$y_0 = p(y_1 \mid x) = \frac{e^{z_0}}{e^{z_0} + e^{z_1}}$$

$[0,1]$

$$y_1 = p(y_2 \mid x) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$

probability of class 2 given input x

activation function: softmax

$$p(y_1 \mid x) + p(y_2 \mid x) = \frac{e^{z_0} + e^{z_1}}{e^{z_0} + e^{z_1}} = 1$$

probability

Exercise 3: Practical tips using keras

- We create keras models like this:
  - Pass a list of „layers“
  - Start with an input layer
- Compile the model with model.compile
  - Provide optimizer, metrics, loss function
- The training is then conducted using model.fit
- A dataset can be passed via
  - A dataloader aka a tf.keras.utils.Sequence
  - A numpy array
- Costumisation is done via callbacks
  - E.g. write out confusion matrices

```
model = tf.keras.models.Sequential(
```

```
tf.keras.Input(name='input_layer', shape=(n_max_frames, n_mfccs)),
```

```
model.fit(x=train_data[0], y=train_data[1],
          steps_per_epoch=int(np.floor(len(train_data[0]) / batch_size)),
          epochs=n_epochs,
          callbacks=get_callbacks(output_dir, val_data, model, patience=patience),
          validation_data=val_data,
          shuffle=False)
```

Exercise 3: Training our first NN

- Work through the second notebook exercise 3a
  - It makes sense to copy the notebook and edit the copy!
  - In the copy you can delete the solutions if you like so you do not tempt yourself
  - !!! Execute only one notebook at a time !!!