base.camp

base.camp Talks
29.07.2020

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Eugen Ruppert & Benjamin Heusser
ruppert at informatik.uni-hamburg.de

BASE.CAMP TALKS

PRACTICE SESSION DEEP LEARNING

Organization

base.camp Talks
Idea

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- lecture series on current and general topics for Computer Science
- practically oriented
- easy to follow
- interactive examples, not just theoretical knowledge

`https://www.inf.uni-hamburg.de/inst/basecamp/events/basecamp-talks.html`

Organization
Schedule 29.07.2020

16:00 Intro: Practical ML (ER)

16:40 Hands-On Session with BSI Brains (BH, BSI)

23:59 BSI Brains Accounts time out

Login to BSI Brains:

`https://partner.bsi-software.com/demo/bsiml_1_2/`

Organization
Zoom Nettiquette

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Turn on your video, if you can
- If available, use a headset
- Please mute your microphone when not in the conversation
  mute on/off: Alt + a – Push to Talk: Space
- We want to record the session and make it available for people who could not join today
  We take privacy seriously. Thus, we will blur/black out your faces, so don't worry about asking questions and giving feedback!

Practical Machine Learning

5 Core Steps of Applied ML

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

1. Exploratory Analysis
2. Data Cleaning
3. Feature Engineering
4. Algorithm Selection
5. Model Training

5 Core Steps of Applied ML
Exploratory Analysis

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- simple algorithm applied on current data
  - e.g. Desicion Tree algorithms
  - Random Forest can capture **non-linearity**
- Question: can we get any meaningful classification from our data?

5 Core Steps of Applied ML
Data Cleaning

- check data for errors, missing values
- work with domain expert:
  - can we provide default values?
  - how to detect outliers or wrong data?

5 Core Steps of Applied ML
Feature Engineering

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- check available data:
  - how can you scale/normalize the data meaningfully? e.g. does it have an exponential, logarithmic growth? – **feature scaling** important for training
  - can you combine multiple data points (e.g. history)?
  - can you create new features out of existing ones?
- bring additional data sources for better classification
- sanity check: better data and features should perform better on the simple algorithm

5 Core Steps of Applied ML
Algorithm Selection

- how much data do you have?
- how many training resources?
- what is an acceptable classification time?
- do you need live updates (online learning)?

5 Core Steps of Applied ML
Model Training

- split data into training and test split
  e.g. 80 % training, 20 % test
- train the actual model (model parameters)
- update and adjust hyperparameters
- monitor for overfitting

Training

Training Process
ML Objective

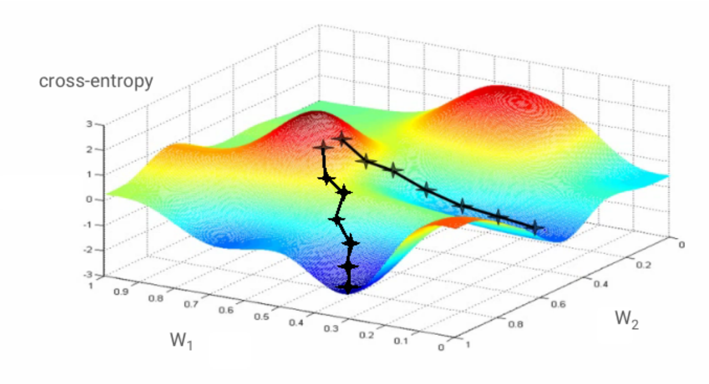
- find patterns in training data
- correspondence between input (features) and output (label)
- minimize Loss on the training data
- **assumption:** training data is drawn from the same distribution as test data

Loss/Cost Function
Loss or Cost Function

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

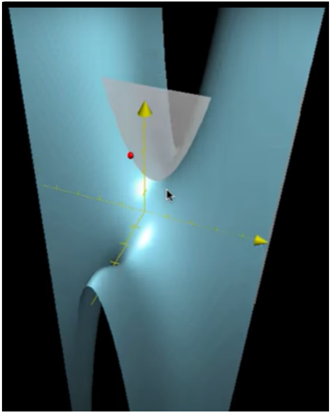
- missclassifications during training are accumulated into Loss
- ideal Loss functon scales with the magnitude of error
  e.g. (squared) distance to the correct value
- gradient of the error can be computed, so that a Gradient Descent algorithm can improve the model

(Stochastic) Gradient Descent



Stochastic Gradient Descent

Partial Derivatives



https://towardsdatascience.com/
a-quick-introduction-to-derivatives-for-machine-learning-people

Training
Learning Rate and Optimizer

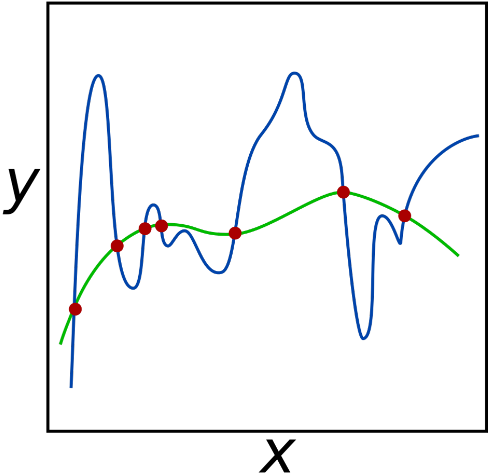
Learning Rate hyperparameter, trade-off:

- high LR (fast, but skipping over optimal solutions)
- low LR (more accurate, but can remain in local minima)
- nice: decaying LR (start fast, slow down gradually)

Optimizer

- SDG robust
- Adam is usually faster
- hyperparameters need to be tuned for both

Overfitting and Regularization



Regularization

- training optimizes on the training data (overfitting)
- we want robustness for classification of unseen data
  - model complexity fed into the Loss function
  - λ: tradeoff between training accurracy and robustness
- best way for regularization: more data!

$$Loss_R(f) = \sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f)$$

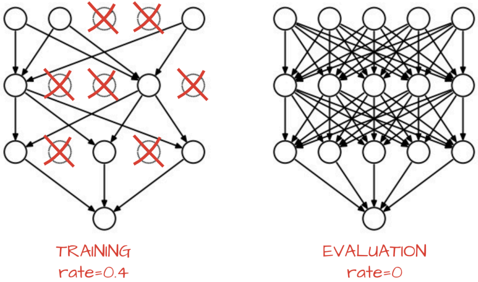
Regularization
L1, L2 Regularization

- **L1** minimize overall model complexity
  can introduce sparsity: many values are zero
- **L2** higher values are penalized more
  reduction of high values, many values become close to zero

$$L_1(f) = \sum_{i=1}^{n} |w(x_i)|$$

$$L_2(f) = \sum_{i=1}^{n} w(x_i)^2$$

Regularization
Dropout



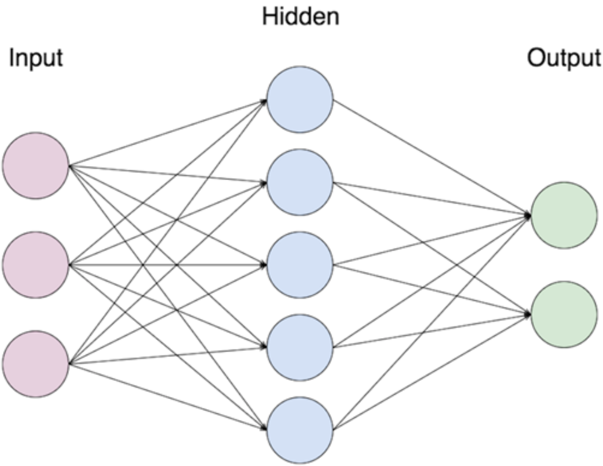
- in deep networks, neurons can provide ‘bad’ output e.g. overfitted on training data
- dropout deactivates random nodes during training
- network becomes more robust and more general, not depending on individual nodes

Deep Learning

Deep Learning

- ML with Deep Neural Networks
- inspired by neurons in the brain
- ‘automatic’ feature extraction and combination
- several layers of processing, e.g. image recognition:
  lines – shapes – eyes – face – person
- large parameter space – benefits greatly from GPUs

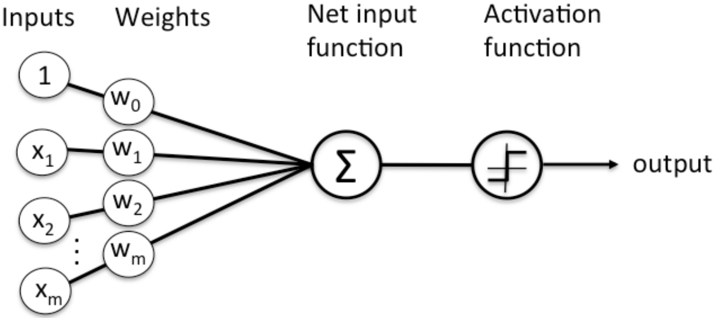
Neural Network



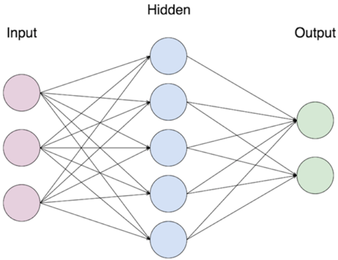
Neuron / Perceptron

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- weighted sum of inputs
- bias
- activation function

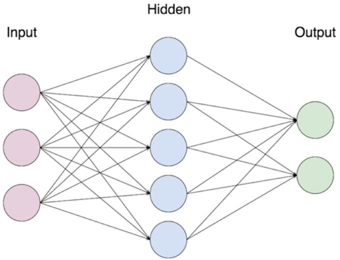


Forward and Back Propagation



- input is fed through the network (forward propagation)
- error is computed on the output layer
- error is back-propagated through all layers
- gradient of the error is identified, weights are updated

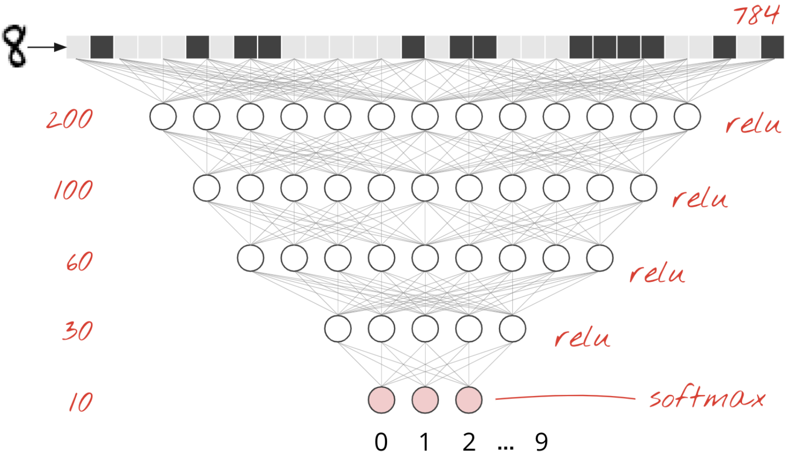
Batches and Epochs



- we compile loss on batches of input elements
  e.g. 1 batch = 100 input examples
- more robustness, stronger direction of the gradient
- 1 epoch = all input examples have been processed
- with complex neural nets, many epochs are required

Network Structure

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

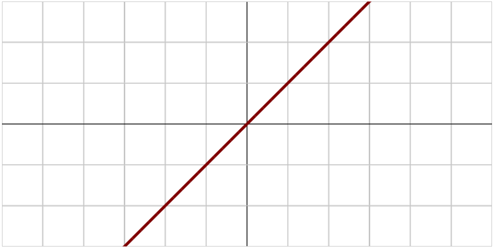


Network Structure

- funneling structure to force feature combinations
- high number of layers leads to many combination options – good for large datasets
- **BUT:** high number of layers increases memory capacity the network learns all the training examples "by heart" – strong overfitting

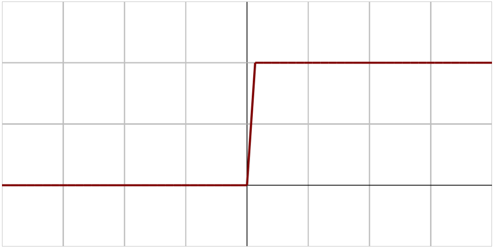
Activation Functions

Linear/Identity Activation



- \+ easy derivation
- – uniformity does not help differentiation
- – correct classifications contribute to learning
  $\rightarrow$ overfitting

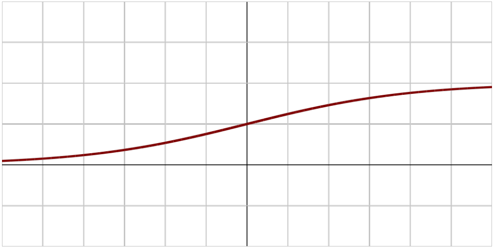
Step Activation



- \+ error class explicitly defined
- \+ only error class activates the neuron
- – no learning is possible, gradient has no direction

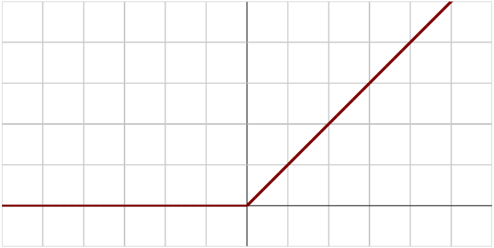
Sigmoid Activation

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



- \+ non-linear
- \+ strong differentiation at decision boundary
- \+ similar to brain neuron activation
- – gradient becomes zero on large errors

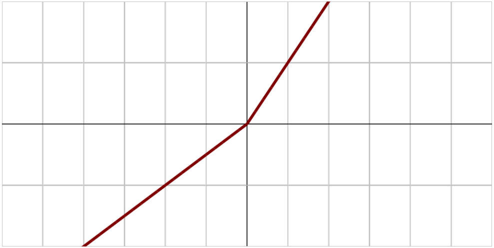
Rectified Linear Unit (ReLU) Activation



- + non-linear
- + can handle all errors margins
- + usually faster conversion than sigmoid
- – cannot learn from positive examples

Leaky ReLU Activation

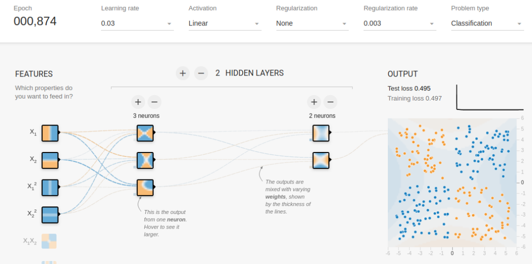
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



+ non-linear
+ can handle all errors margins
+ similar to brain neuron activation
+ usually faster conversion than sigmoid
+ learning from positive examples at a lower rate

Universality

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



- non-linear data requires non-linear activation functions
- non-linear data requires non-linear feature representation
- you can **approximate any non-linear function** with non-linear activation functions

`http://playground.tensorflow.org/`

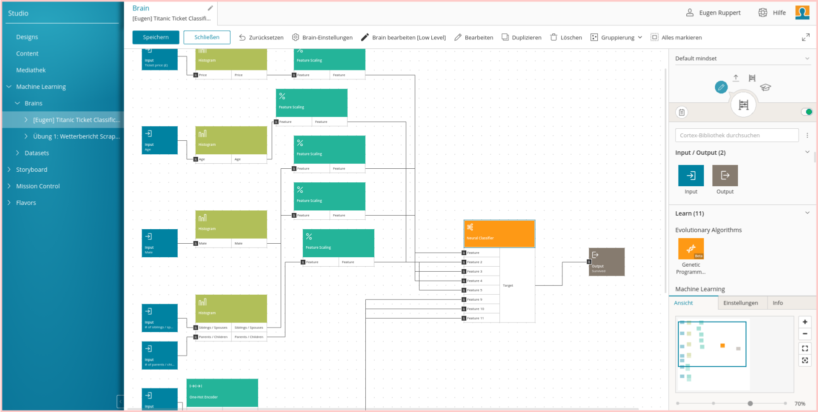
Conclusion

Conclusion

Take-away points:

- how to do practical ML
- understanding network engineering and hyperparameter tuning
- knowledge to aid in understanding papers and lectures
- motivation to try out neural networks

BSI Brains
Overview

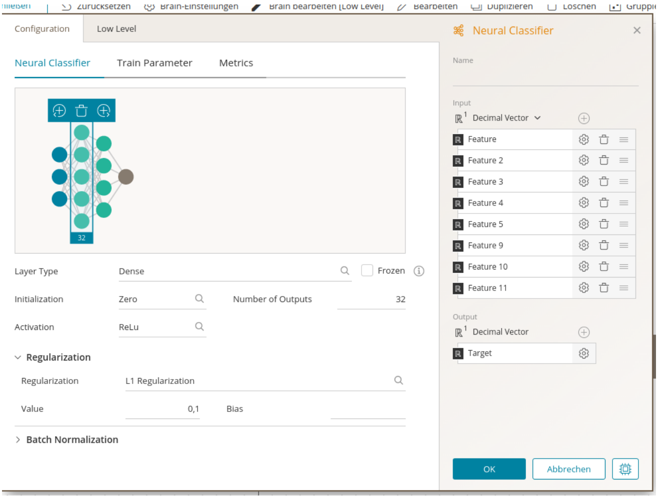


BSI Brains

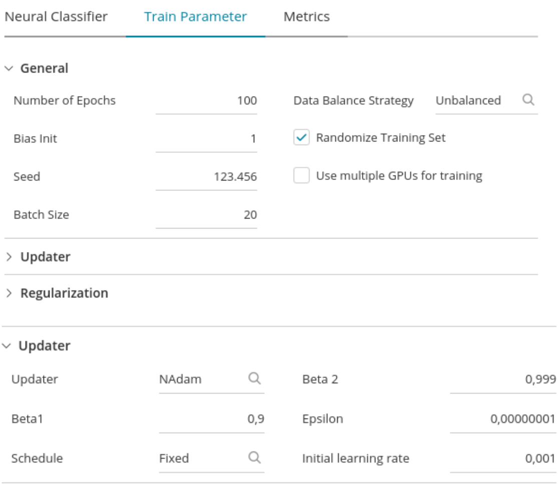
Put ideas into practice!

- Online Tool for data handling and ML experiments
  `https://partner.bsi-software.com/demo/bsiml_1_2/`
- backend based on DeepLearning4J
  `https://deeplearning4j.org/`
- easily configurable network structures
- hyperparemeter settings

BSI Brains
Network Configuration



BSI Brains
Hyperparameters



Resources

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Tensorflow Playground
  `http://playground.tensorflow.org/`
- Tensorflow Tutorial
  `https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/`
  Code: `https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/tree/master/tensorflow-mnist-tutorial`
- *Neural Networks and Deep Learning* book
  `http://neuralnetworksanddeeplearning.com/`

Fin

Evaluation

		Actual value	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Evaluation

- Accuracy = TP / TP + FP + TN + FN
- Precision = TP / TP + FP
- Recall = TP / TP + FN
- F-score = 2 * Pr * Rec / Prec + Rec

Baselines

- complex ML models need to be justified
- baselines can provide justification
- simple baselines
  - random
  - most frequent class
- strong baselines
  - perceptron
  - related work
  - simple neural networks

Softmax Activation

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Predictions   Images   Weights   Biases

Y[100, 10]   X[100, 784]   W[784,10]   b[10]

$$Y = softmax(X.W + b)$$

applied line by line   matrix multiply   broadcast on all lines

tensor shapes in [ ]

Softmax Activation

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- on output layer
- transforms activations into probability distribution
- exponential funciton to increase contrast and give the highest activation a value close to 1.0