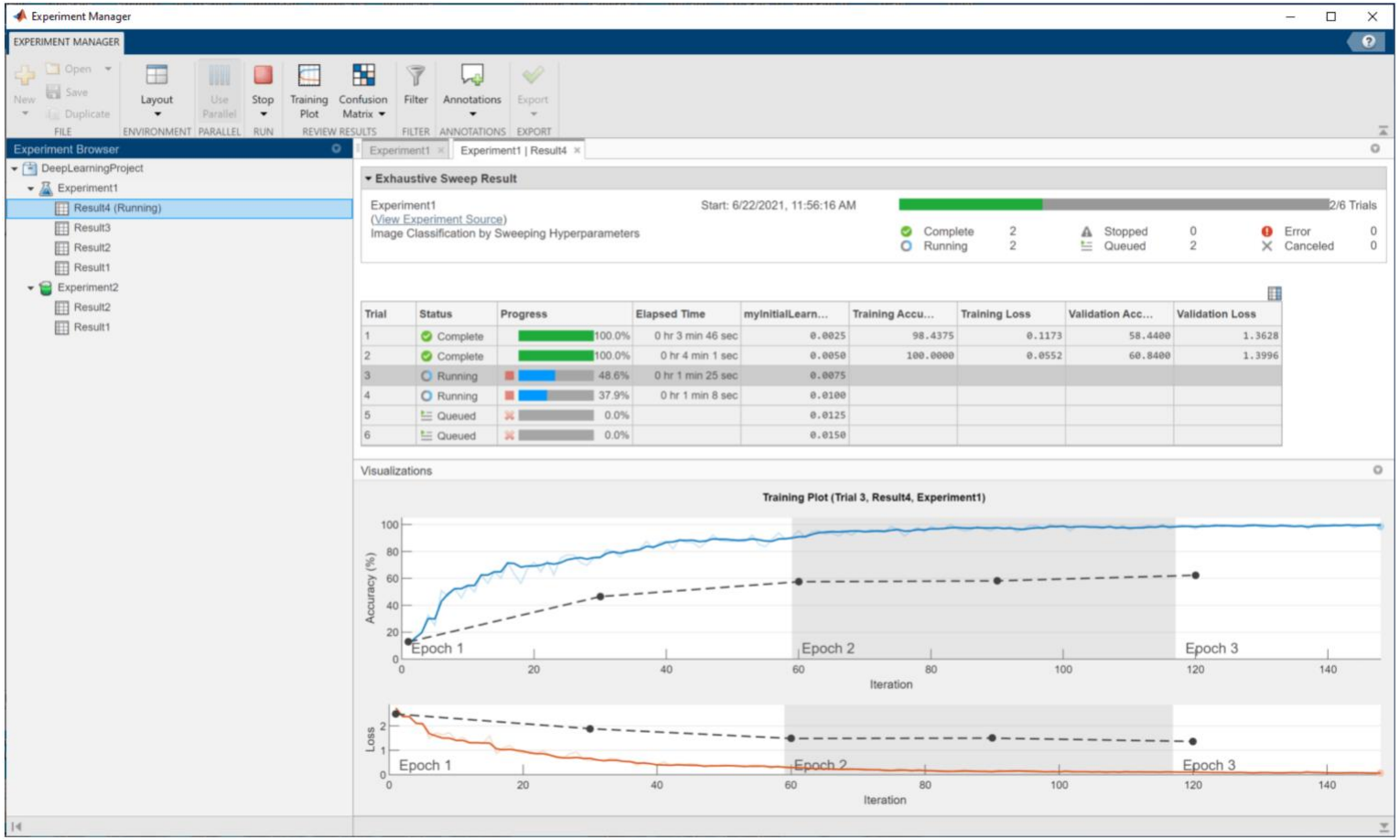


Festlegung der Hyperparameter

<u>WICHTIG:</u> Rückfragen zur Case Study werden nach dem 05.07.2022 bis zur finalen Abgabe nicht mehr beantwortet!

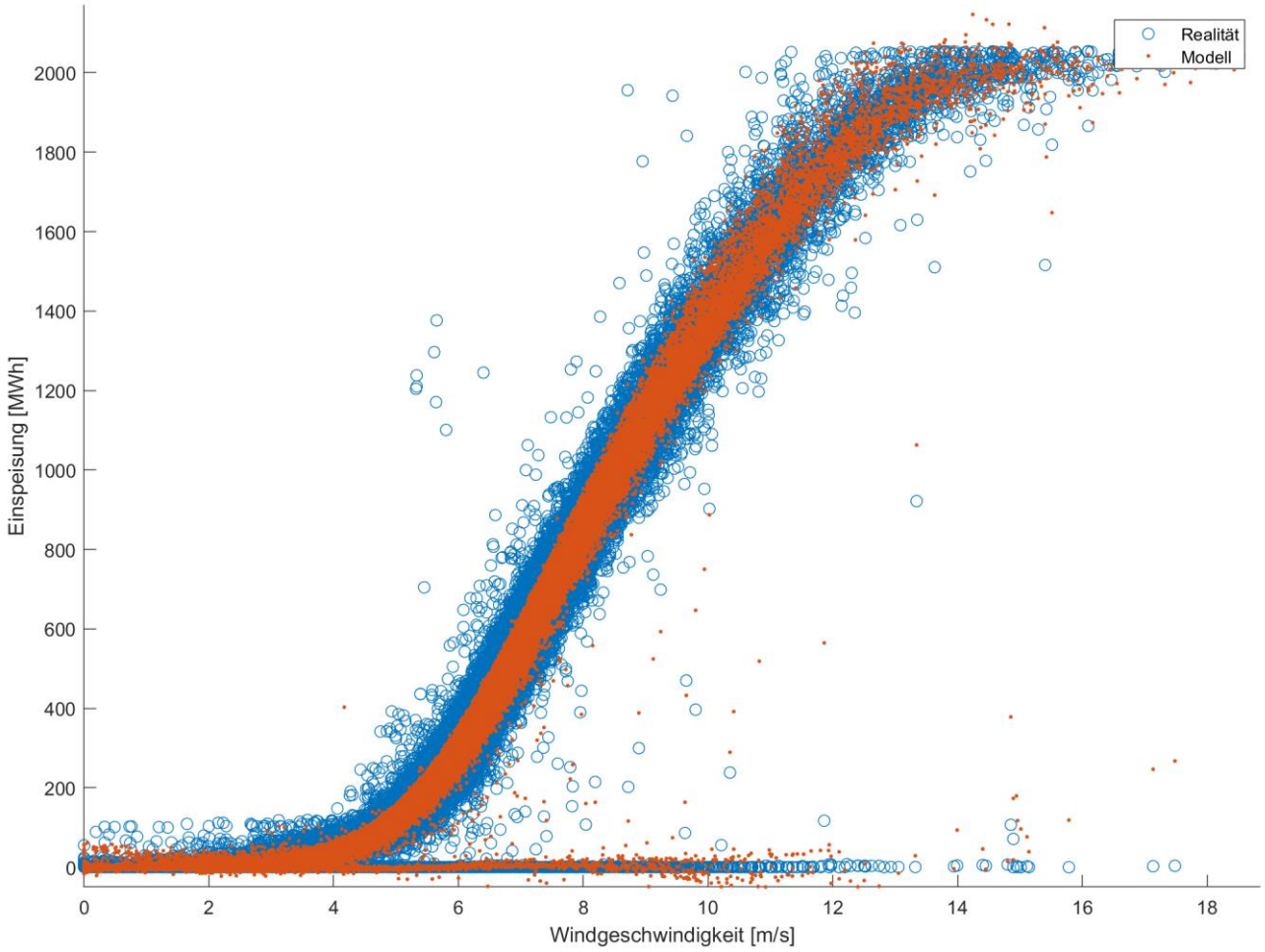
VL	Termin	Inhalt
29.03.2022	1	Einleitung / deskriptive Analyse/ Lin. Regr.
05.04.2022	2	Vertiefende Anwendungen Lineare Regression
12.04.2022	3	Einführung in die Modellauswahl
19.04.2022	4	Datenreduktionstechniken
26.04.2022	5	Nichtlineare Regression mit Splines und lokaler Regression
03.05.2022	6	KNN Feed Forward Regression
10.05.2022	7	Hyperparameterauswahl
17.05.2022	8	Rekurrente Netze:KNN LSTM Regression
24.05.2022	9	Rekurrente Netze:KNN LSTM CNN Regression
31.05.2022	10	Einführung in Clusterverfahren
07.06.2022	11	Arbeiten an Case Study Teil B
14.06.2022	12	Einführung in Klassifizierung mit KNN
21.06.2022	13	Anwendungen KNN: Non Intrusive Load Management
28.06.2022	14	Anwendungen: Predictive Maintenance
05.07.2022	15	Rückfragen Case Study Teil B

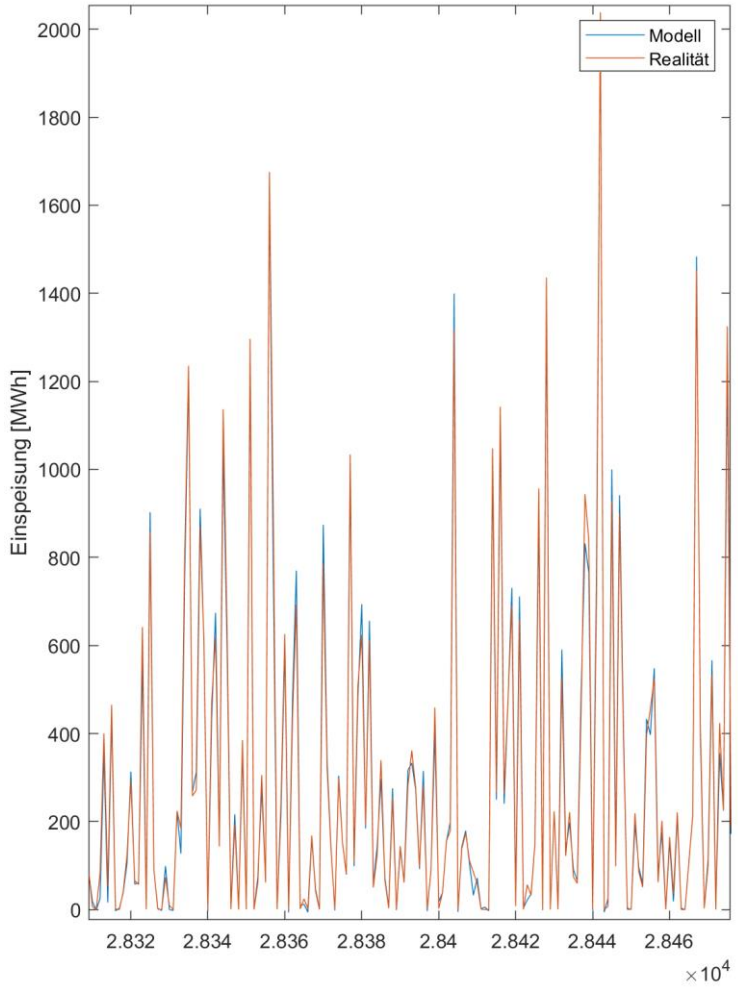
Benchmark-modell

Vertiefendes Modell

Zeitslot der Synchronveranstaltung 14:15 – 17:30

Rücksprache Übung Windkraftanlage



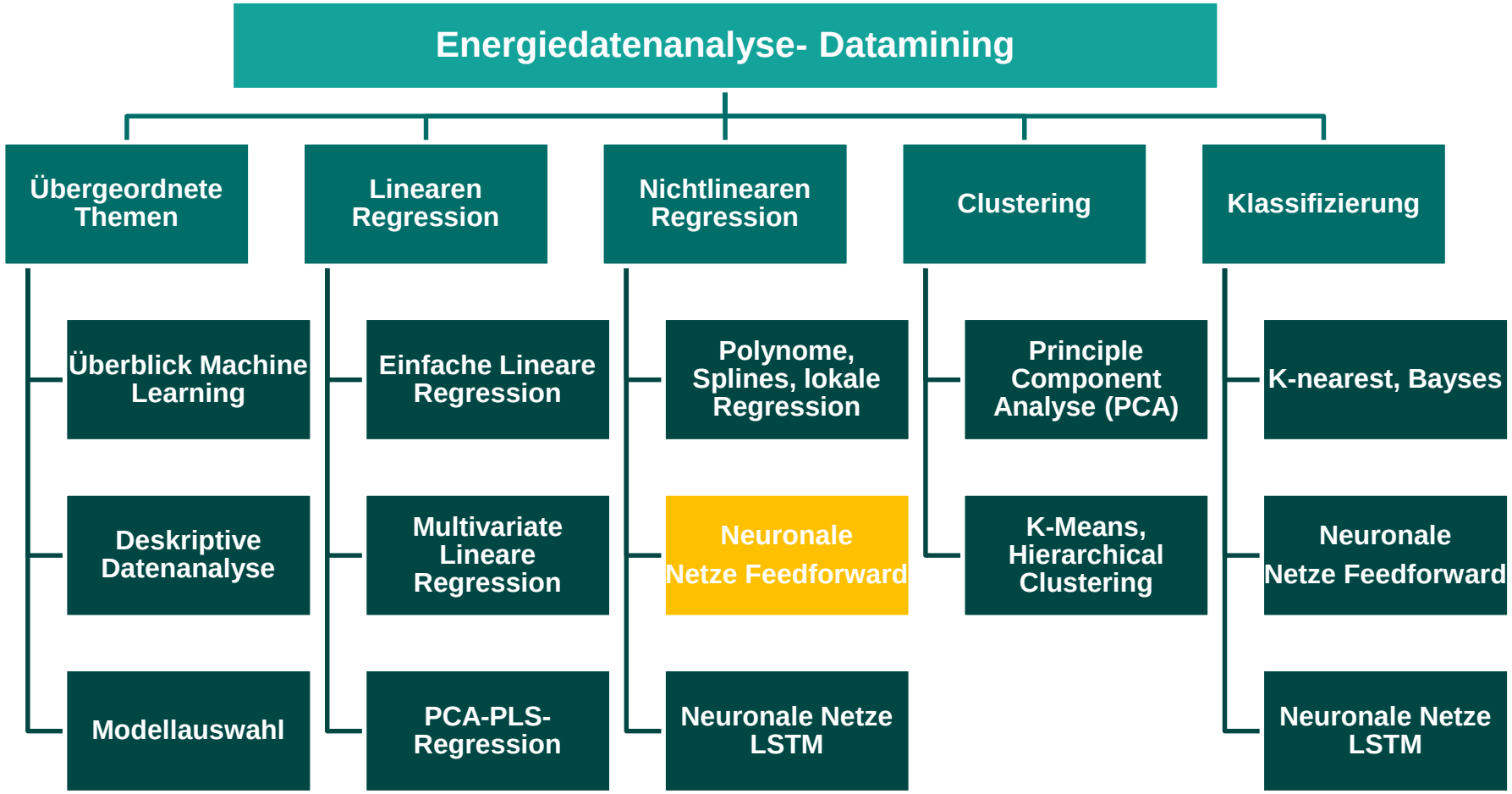


Zielsetzung der heutigen Vorlesung:
Einführung in Hyperparameteroptimierung und Quantilsregression

Thema	Anwendung der Hyperparameteroptimierung zum Auffinden der optimalen Konfiguration eines neuronalen Netzes

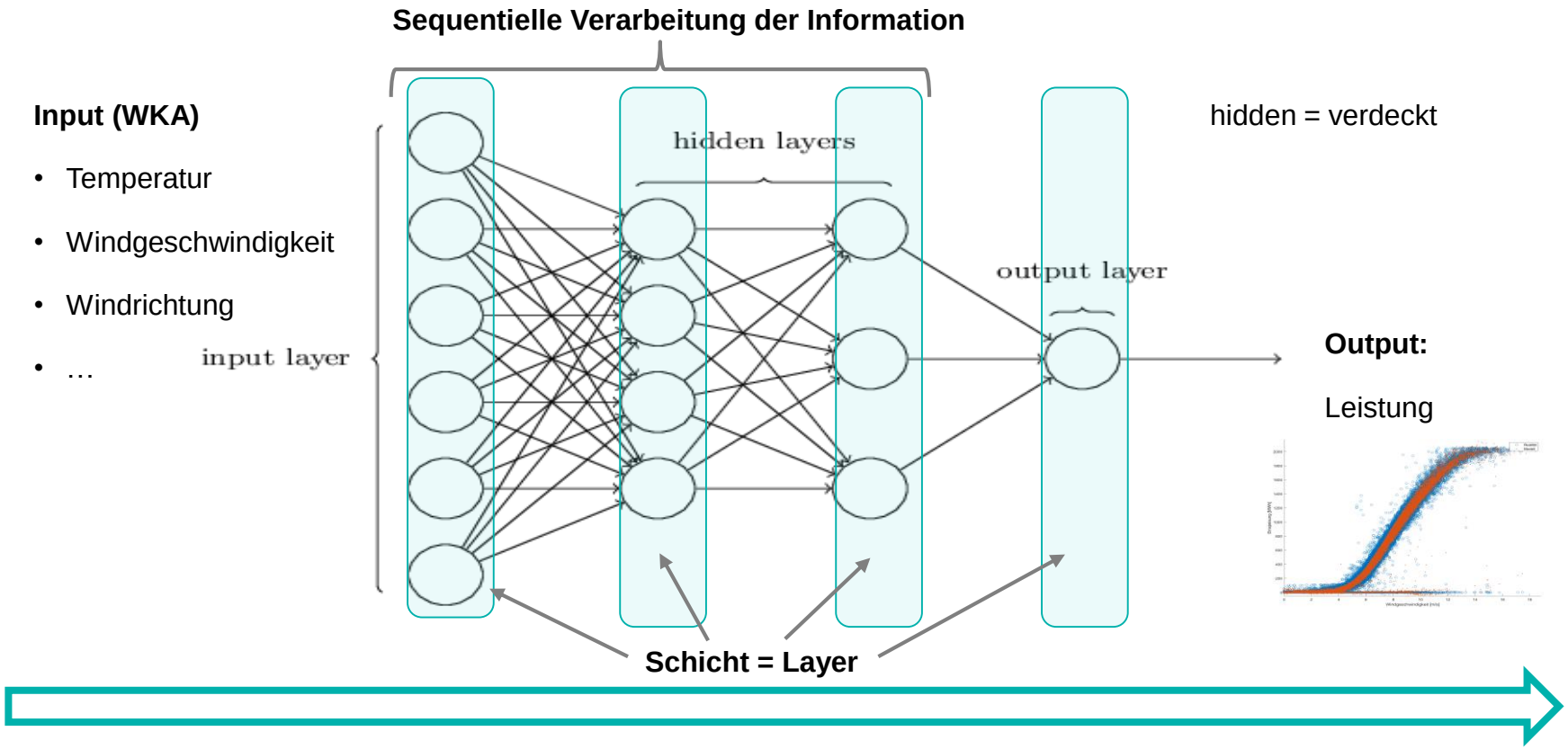
	Aufbau der heutigen Vorlesung:	**Lernziel:**
1	Hyperparameter im Optimierungsalgorithmus	Begriffe **Adam, SGDM, RMSProp** verstehen
2	Experimente-Manager in Matlab	Wie wird der Experimente–Manager aufgesetzt
3	Quantilsregression	Einsatzzweck und Abbildung verstehen

Die Themengebiete der Veranstaltung verknüpfen Modelle des „Machine Learning“ mit energiewirtschaftlichen Fragestellungen



Wiederholung: Mit Hilfe eines neuronalen Netzes können Input-Informationen anhand definierter inbs. nichtlinearer Funktionen verarbeitet (aktiviert) und in ein Output ausgegeben werden.

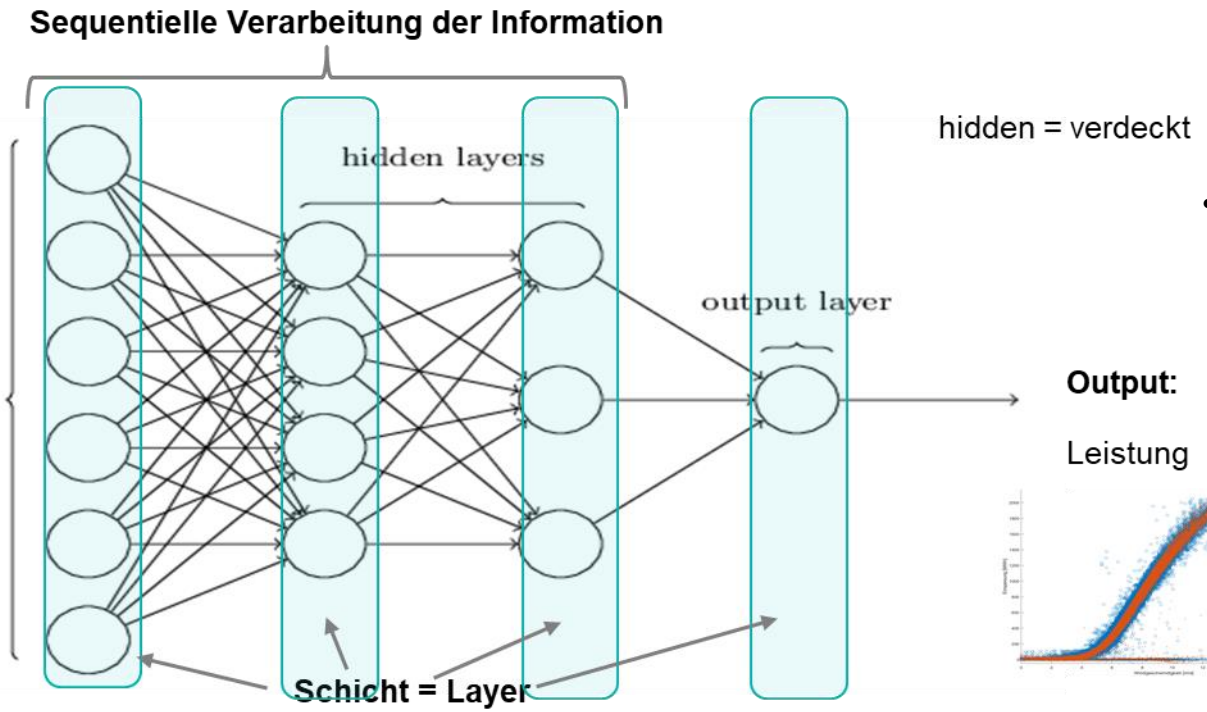
- Feed-Forward Networks: Information fließt vom **Input** Layer nur in einer Richtung zum **Output** Layer.



Bei einem neuronalen Netz müssen eine Vielzahl von Parametern in der Topologie festgelegt werden

Input (WKA)

- Temperatur
- Windgeschwindigkeit
- Windrichtung
- …



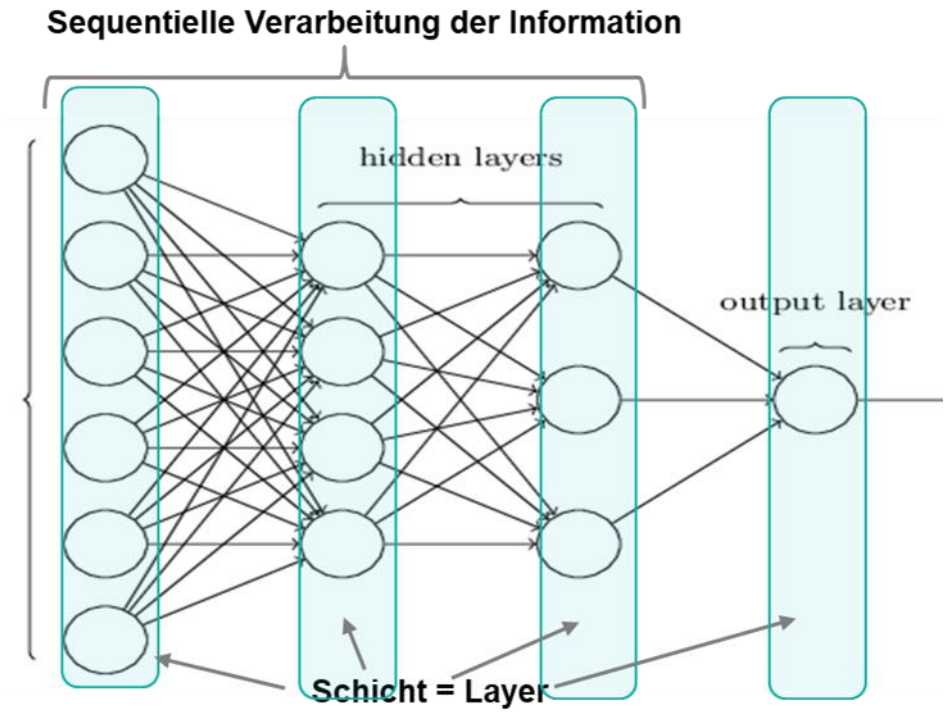
Output:

Leistung

- **Bestimmung von**
  - Anzahl Input-Knoten
  - Anzahl der verdeckten Schichten
  - Art der Aktivierungsfunktion
  - Anzahl der Output-Knoten
- **Einfluss der Komplexität des Netzes**
  - Gering –> hoher Bias
  - Hoch -> hohe Varianz (Overfitting)

Diese Festlegung erfolgt ex ante durch den Anwender und wird nicht durch den Algorithmus optimiert!

Architektur



- 1) Inputs:
  - Es sollten nur Informationen verwendet werden, welche auch einen Zusammenhang vermuten lassen!
  - Die Übergabe ist i.d.R. normiert
  - Unterteilung in Training, Validierung und Testdaten
- 2) Netztopologie:
  - Anzahl der Schichten
  - Anzahl der Neuronen pro Schicht
  - Form der Aktivierung je Schicht
  - Jedes Gewicht stellt einen Freiheitsgrad dar
  - Gefahr des „Overfitten“

Schritte zum Aufbau eines neuronalen Netzes

Formulierung des Zusammenhangs

- Auswahl und **Normierung** Input- und Outputgrößen

Vorbereitung der Daten

- Trennung Trainings-, Validierungs-, und Testdaten
- Zuschnitt von X und Y aus der Datensequenz

Aufbau des neuronalen Netzes

- Anzahl der Schichten
- Typ der Schichten
- Art der Aktivierung
- Anzahl der Neuronen pro Schicht

Festlegung der weiteren Hyperparameter

- Wahl der Verlustfunktion
- Wahl der Epochen
- Wahl des Optimierungsmechanismus
- Wahl der Parameter des Optimierungsmechanismus

Was ist der Weg durch den Dschungel?

- Eine unwissenschaftliche Leitregel:
  - Schritt 1, keep it simple, um mit Hyperparametern effizient zu „experimentieren“:
    - Normierung der Daten
    - wenige Variablen
    - einfache Netzwerke mit niedriger Komplexität
    - Verringerung des Datensatzes
    - Gewinnung von Erfahrung und einer Basislösung
    - Erste Festlegung der Hyperparameter
  - Schritt 2, Erhöhung der Komplexität für das Feintuning der Parameter
    - Hinzunahme von Variablen
    - Erhöhung der Komplexität des Netzwerks
    - Erhöhung des Datensatzes

Mögliche Anpassungen bei Overfitting:
eine nicht abschließende Aufzählung von Gegenmaßnahmen I: Regularisierung

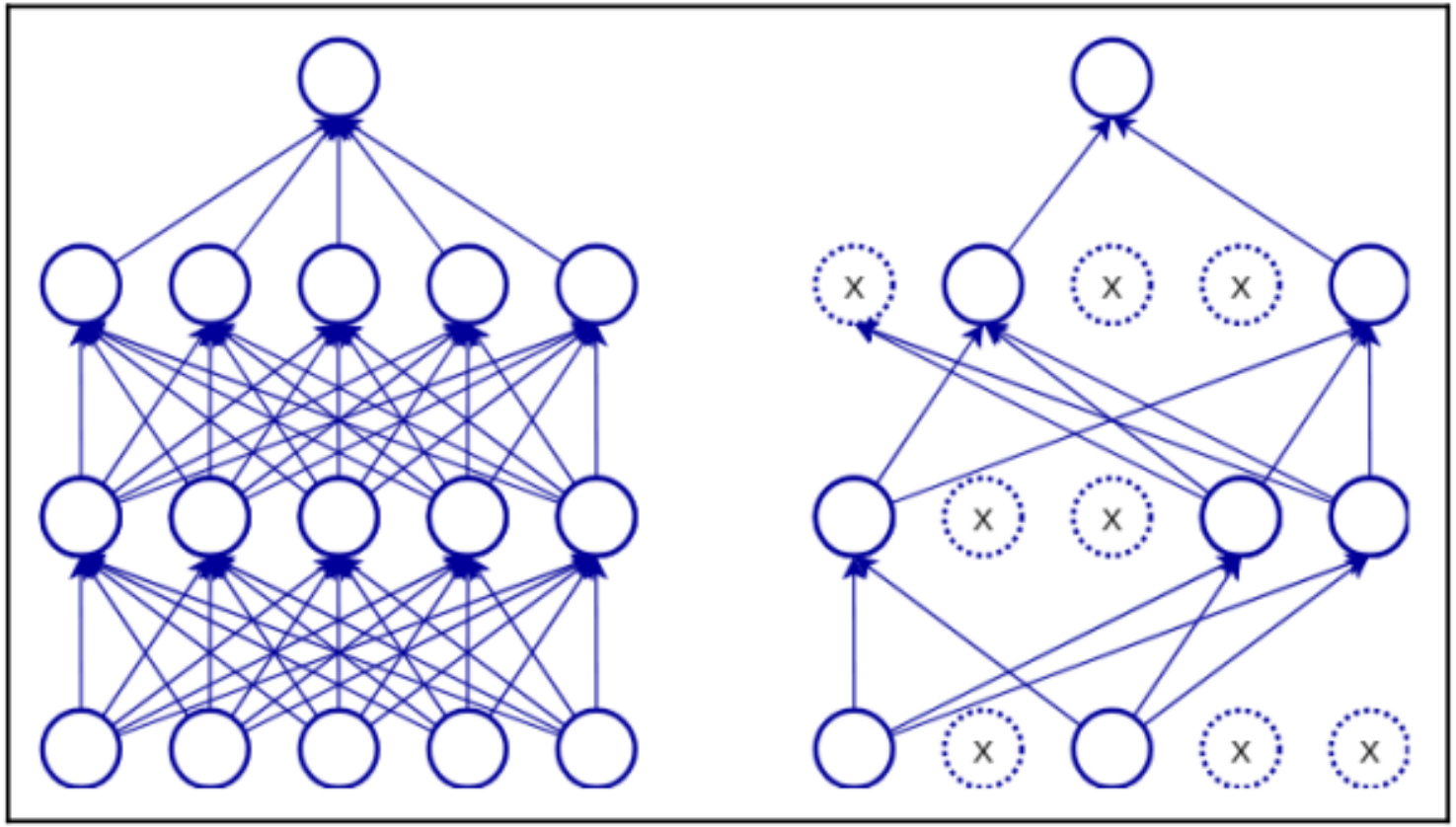
Regularisierung

$$\underbrace{C = C_0 + \frac{\lambda}{n}\sum_w |w|}_{L1-Regularisierung} \qquad \underbrace{C = C_0 + \frac{\lambda}{2n}\sum_w w^2}_{L2-Regularisierung}$$

1. Reduktion der Komplexität des Modells
    1. Anzahl der Schichten
    2. Anzahl der Neuronen pro Schicht
2. Reduktion der Anzahl der Epochen zum Lernen
3. Regularisierung
    1. Mit Hilfe von Straftermen entsprechend der L1 bzw. L2 –Regularisierung

Mögliche Anpassungen bei Overfitting:
eine nicht abschließende Aufzählung von Gegenmaßnahmen I: Drop Out -Layer

- Im Rahmen der Drop-out-Regularisierung werden pro Umlauf zufällig Neuronen ausgeschaltet, um Overfitting zu vermeiden

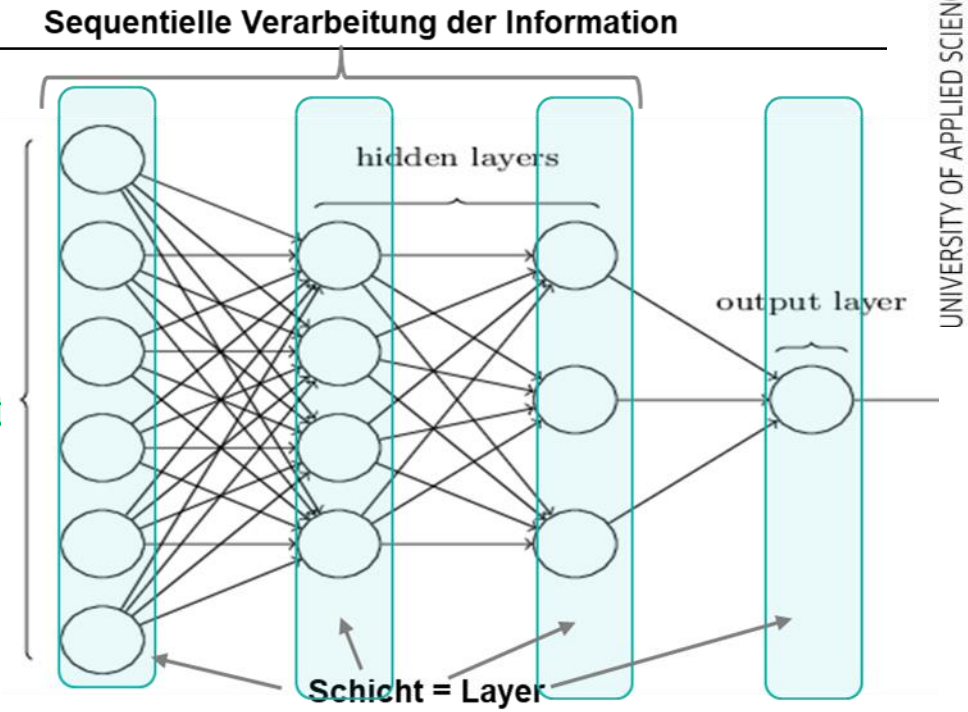


Lösungsstrategien bei Problemen

Problem	Possible Solution
NaNs or large spikes in the loss	Decrease the initial learning rate using the `'InitialLearnRate'` option of `trainingOptions`. If decreasing the learning rate does not help, then try using gradient clipping. To set the gradient threshold, use the `'GradientThreshold'` option in `trainingOptions`.
Loss is still decreasing at the end of training	Train for longer by increasing the number of epochs using the `'MaxEpochs'` option in `trainingOptions`.
Loss plateaus	If the loss plateaus at an unexpectedly high value, then drop the learning rate at the plateau. To change the learning rate schedule, use the `'LearnRateSchedule'` option in `trainingOptions`. If dropping the learning rate does not help, then the model might be underfitting. Try increasing the number of parameters or layers. You can check if the model is underfitting by monitoring the validation loss.
Validation loss is much higher than the training loss	To prevent overfitting, try one or more of the following: • Use data augmentation. For more information, see Train Network with Augmented Images. • Use dropout layers. For more information, see `dropoutLayer`. • Increase the global L2 regularization factor using the `'L2Regularization'` option in `trainingOptions`.
Loss decreases very slowly	Increase the initial learning rate using the `'InitialLearnRate'` option of `trainingOptions`. For image data, try including batch normalization layers in your network. For more information, see `batchNormalizationLayer`.

1. Optimierungskernels
2. Experimente-Manager
3. Quantilsregression

Umsetzung / Aufbau in Matlab -> Aufbau der Layer




```
numFeatures = size(X,2);                    %-> Anzahl der Inputs
numResponses = 1;                                %-> Anzahl der Input

numHiddenUnits1 = 400;                      %-> Definition beliebig vieler
…                                                Schichten
numHiddenUnitsn = 400;

%Zusammensetzung im Netz
layers = [ ...                              % Zusammenfassung der einzelnen Layer
sequenceInputLayer(numFeatures)             % Inputlayer
fullyConnectedLayer(numHiddenUnits1)        % erster Hidden Layer
tanhLayer                                   % Aktivierungsfunktion des ersten Hidden Layer
…
fullyConnectedLayer(numResponses)           % Ausgabelayer
regressionLayer];                           % Ausgabefunktion
```

Schritte zum Aufbau eines neuronalen Netzes

Formulierung des Zusammenhangs

- Auswahl und **Normierung** Input- und Outputgrößen

Vorbereitung der Daten

- Trennung Trainings-, Validierungs-, und Testdaten
- Zuschnitt von X und Y aus der Datensequenz

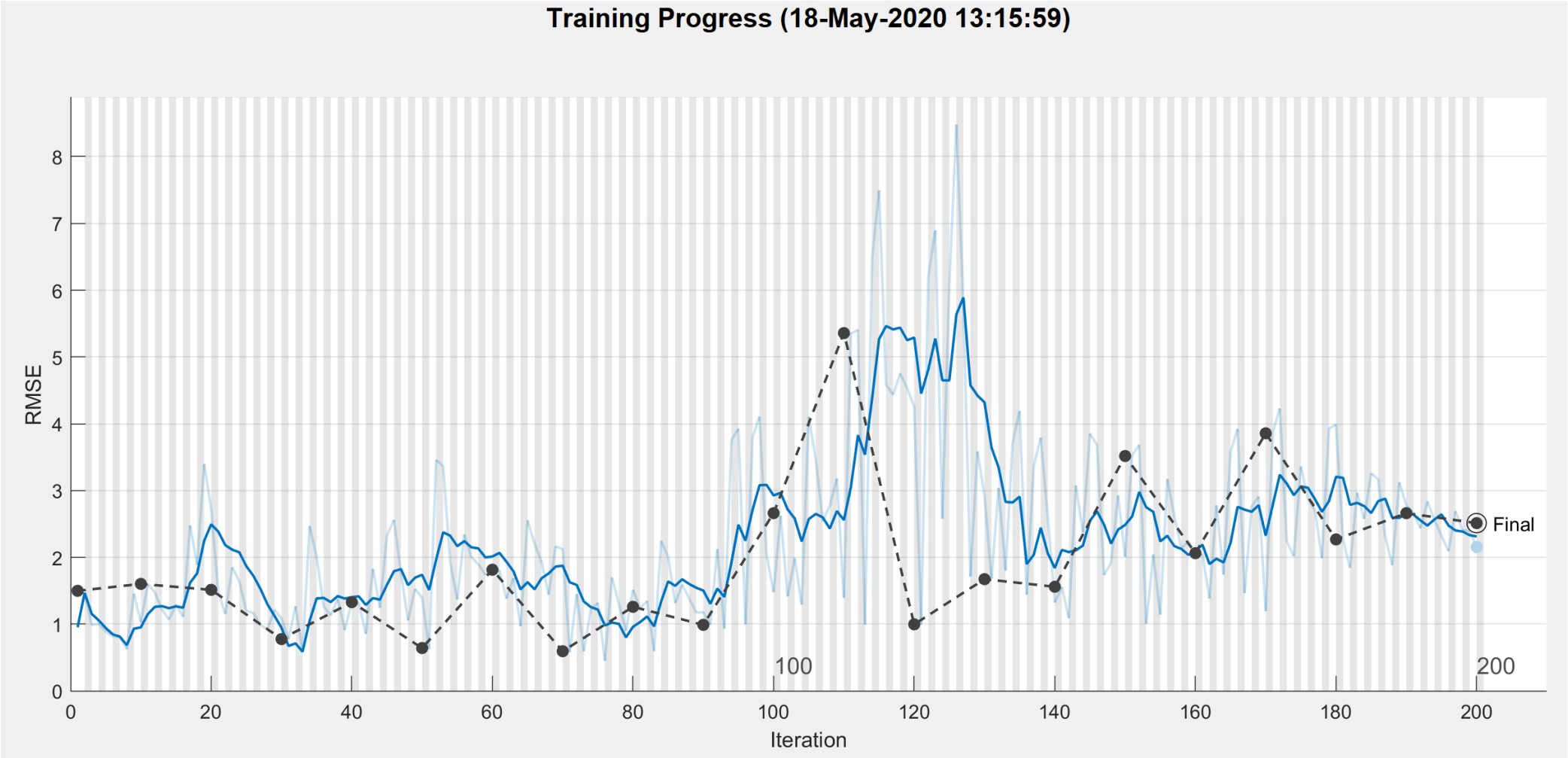
Aufbau des neuronalen Netzes

- Anzahl der Schichten
- Typ der Schichten
- Art der Aktivierung
- Anzahl der Neuronen pro Schicht

Festlegung der Hyperparameter

- Wahl der Verlustfunktion
- Wahl der Epochen
- Wahl des Optimierungsmechanismus
- Wahl der Parameter des Optimierungsmechanismus

Der Lösungsprozess wird über sogenannte Hyperparameter optimiert



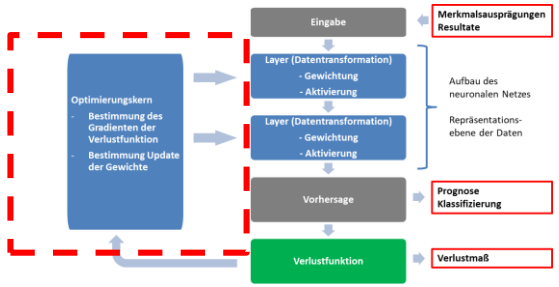
Schritt 4: Festlegung der Hyperparameter

options = trainingOptions(solverName,Name,Value) returns training options with additional options specified by one or more name-value pair arguments

```
                     Momentum: 0.9000
             InitialLearnRate: 0.0100
    LearnRateScheduleSettings: [1x1 struct]
            L2Regularization: 1.0000e-04
      GradientThresholdMethod: 'l2norm'
            GradientThreshold: Inf
                    MaxEpochs: 20
                MiniBatchSize: 64
                      Verbose: 1
              VerboseFrequency: 50
                ValidationData: []
```

```
          ValidationFrequency: 50
           ValidationPatience: Inf
                      Shuffle: 'once'
               CheckpointPath: ''
          ExecutionEnvironment: 'auto'
                   WorkerLoad: []
                    OutputFcn: []
                        Plots: 'training-progress'
               SequenceLength: 'longest'
          SequencePaddingValue: 0
      SequencePaddingDirection: 'right'
          DispatchInBackground: 0
       ResetInputNormalization: 1
```

Einsetzbare Optimierungsverfahren I



- Stochastic Gradient Descent
  Anpassung der Gewichte in Richtung des negativen Gradienten der Verlustfunktion

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \alpha \nabla L(\boldsymbol{w}_l)$$

mit: $\boldsymbol{w}$ = Gewichte; ∇L= Gradient der Zielfunktion; $\alpha > 0 \; = \; Lernrate$

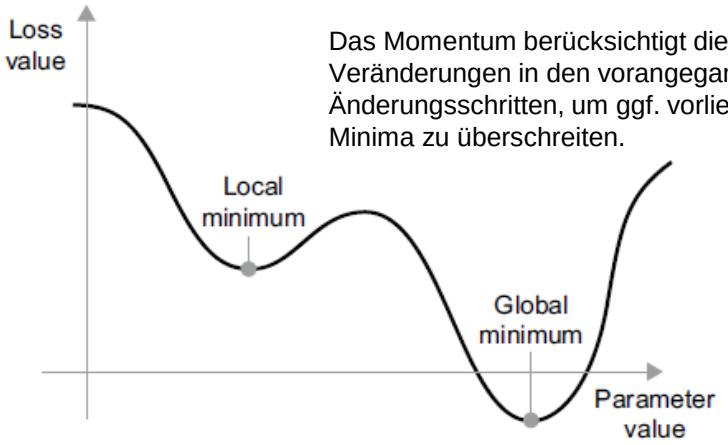
Das Momentum adressiert zwei Aspekte des “stochastic gradient” -Verfahrens: **Konvergenz** und **lokale Minima**

- Stochastic Gradient Descent with Momentum (**SGDM**)
  Anpassung unter Berücksichtigung der Anpassung im letzten Zeitschritt (Korrektiv)

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \alpha \nabla L(\boldsymbol{w}_l) + \underbrace{\gamma(\boldsymbol{w}_l - \boldsymbol{w}_{l-1})}$$

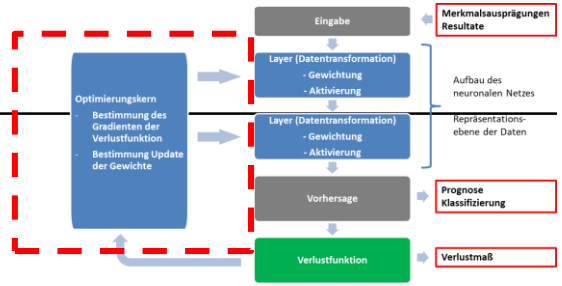
mit $\gamma = \; Momentum \; < 1$

Berücksichtigung des Momentum



Das Momentum berücksichtigt die Stärke der Veränderungen in den vorangegangenen Änderungsschritten, um ggf. vorliegende lokale Minima zu überschreiten.

Einsetzbare Optimierungsverfahren II



- root mean square propagation (**RMSProp**)
  individuelle Lernrate für jeden Parameter passt sich dem Gradienten an

  mit β = Abklingrate <1; ν =Lernrate

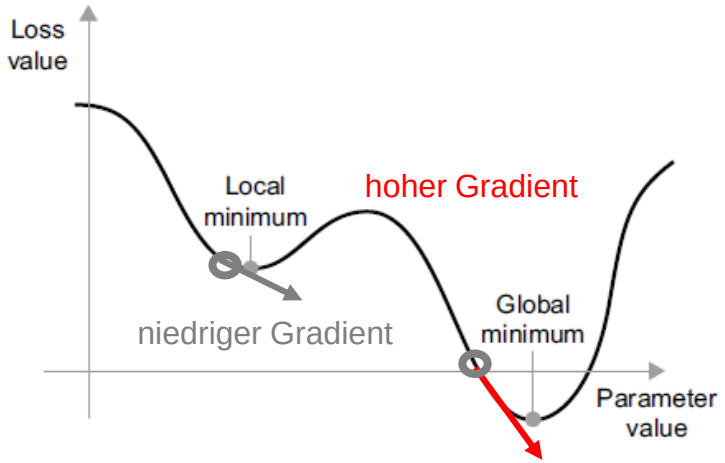
„InitialLearnRate“ wird angepasst

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \frac{\alpha}{\sqrt{v_l} + \varepsilon} \nabla L(\boldsymbol{w}_l), \underbrace{v_l = \beta_2 v_{l-1}}_{v_l \rightarrow \mathbf{AR-Prozess}} + (1 - \beta_2)[\nabla L(\boldsymbol{w}_l)]^2$$

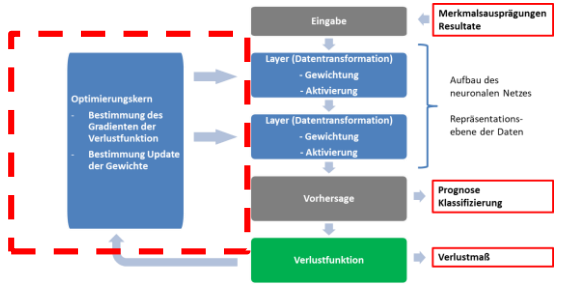
„SquaredGradientDecayFactor“

- <u>Auswirkungen</u>:
  - Bei einem hohen Gradienten folgt:

    $[\nabla L(\boldsymbol{w}_l)]^2 \uparrow \rightarrow v_l \uparrow \rightarrow \frac{\alpha}{\sqrt{v_l}+\varepsilon} \downarrow$
  - Hierdurch wird verhindert, dass Minima „übersprungen“ werden
  - Bei niedrigem Gradienten erhöht sich $\frac{\alpha}{\sqrt{v_l}+\varepsilon}$
  - Dies vergrößert die Schrittweite in Bereichen mit geringer Steigung der Zielfunktion



Einsetzbare Optimierungsverfahren III



- **Adam** (derived from adaptive moment estimation)
  ähnelt RMSProp inkl. elementweise gleitender Mittelwert aus den Parametergradienten
  - Es wird statt dem Gradienten $\nabla J(\boldsymbol{w}_l)$ ein komplexerer Term m_l

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \frac{\alpha m_l}{\sqrt{v_l} + \varepsilon}, v_l = \beta_2 v_{l-1} + (1 - \beta_2)[\nabla L(\boldsymbol{w}_l)]^2$$

„SquaredGradientDecayFactor“

mit β = Abklingrate <1; ν =Lernrate

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1)\nabla JL(\boldsymbol{w}_l)$$

„GradientDecayFactor“

- Wirkungsweise: m_l ist ähnlich zu einem ARX-Modell aufgebaut → Wirkung von $\nabla L(\boldsymbol{w}_l)$ wird **geglättet.**
- Verlauf von m_l ist glatter als Verlauf von $\nabla L(\boldsymbol{w}_l)$

Anhang: Optionen zur Festlegung der Hyperparameter

solverName — Solver for training network
'sgdm' | 'rmsprop' | 'adam'

'Plots' — Plots to display during network training
'none' (default) | 'training-progress'

'Verbose' — Indicator to display training progress information
1 (true) (default) | 0 (false)

'ValidationFrequency' — Frequency of network validation
50 (default) | positive integer

'InitialLearnRate' — Initial learning rate
positive scalar

'L2Regularization' — Factor for L_2 regularization
0.0001 (default) | nonnegative scalar

'Momentum' — Contribution of previous step
0.9 (default) | scalar from 0 to 1

'GradientDecayFactor' — Decay rate of gradient moving average
0.9 (default) | scalar from 0 to 1

'SquaredGradientDecayFactor' — Decay rate of squared gradient moving average
nonnegative scalar less than 1

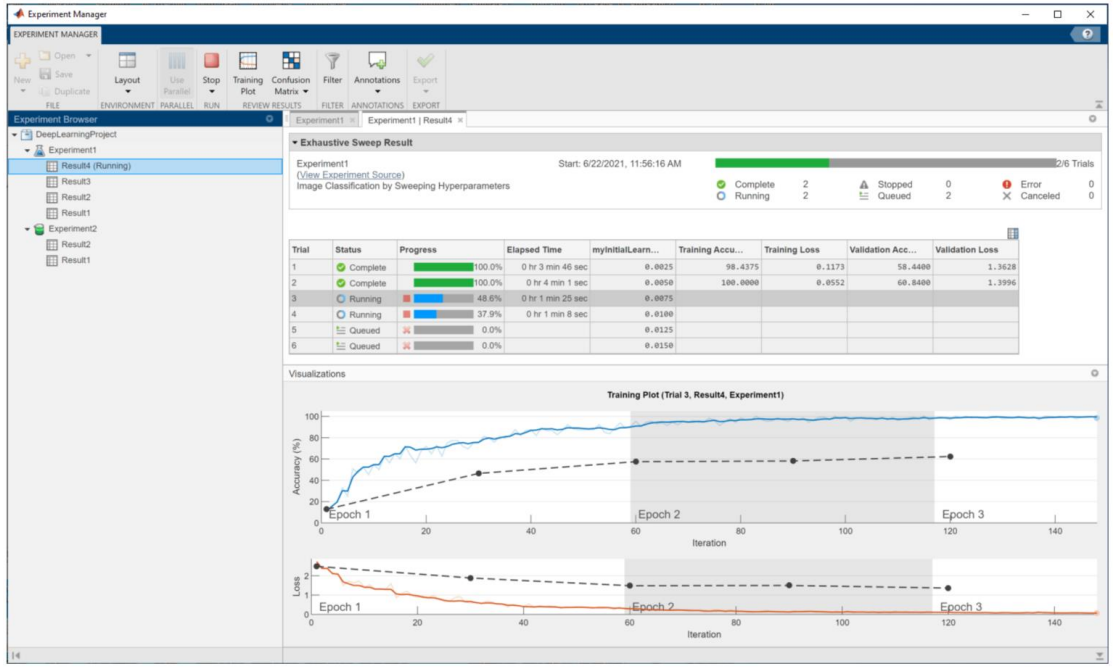
'ExecutionEnvironment' — Hardware resource for training network
'auto' (default) | 'cpu' | 'gpu' | 'multi-gpu' | 'parallel'

Field	Description
Epoch	Epoch number. An epoch corresponds to a full pass of the data.
Iteration	Iteration number. An iteration corresponds to a mini-batch.
Time Elapsed	Time elapsed in hours, minutes, and seconds.
Mini-batch RMSE	Root-mean-squared-error (RMSE) on the mini-batch.
Validation RMSE	RMSE on the validation data. If you do not specify validation data, then the software does not display this field.
Mini-batch Loss	Loss on the mini-batch. If the output layer is a RegressionOutputLayer object, then the loss is the half-mean-squared-error.
Validation Loss	Loss on the validation data. If the output layer is a RegressionOutputLayer object, then the loss is the half-mean-squared-error. If you do not specify validation data, then the software does not display this field.
Base Learning Rate	Base learning rate. The software multiplies the learn rate factors of the layers by this value.

1. Optimierungskernels
2. Experimente-Manager
3. Quantilsregression

Mit Hilfe des Experimente-Managers kann eine optimale Einstellung der Hyperparameter „gesucht“ werden

Darstellung

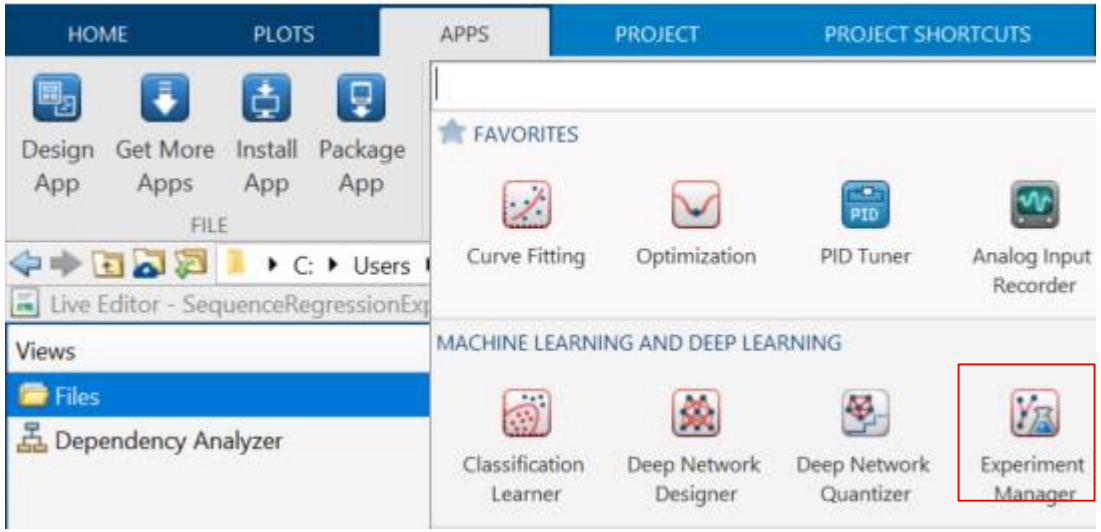


Erläuterung

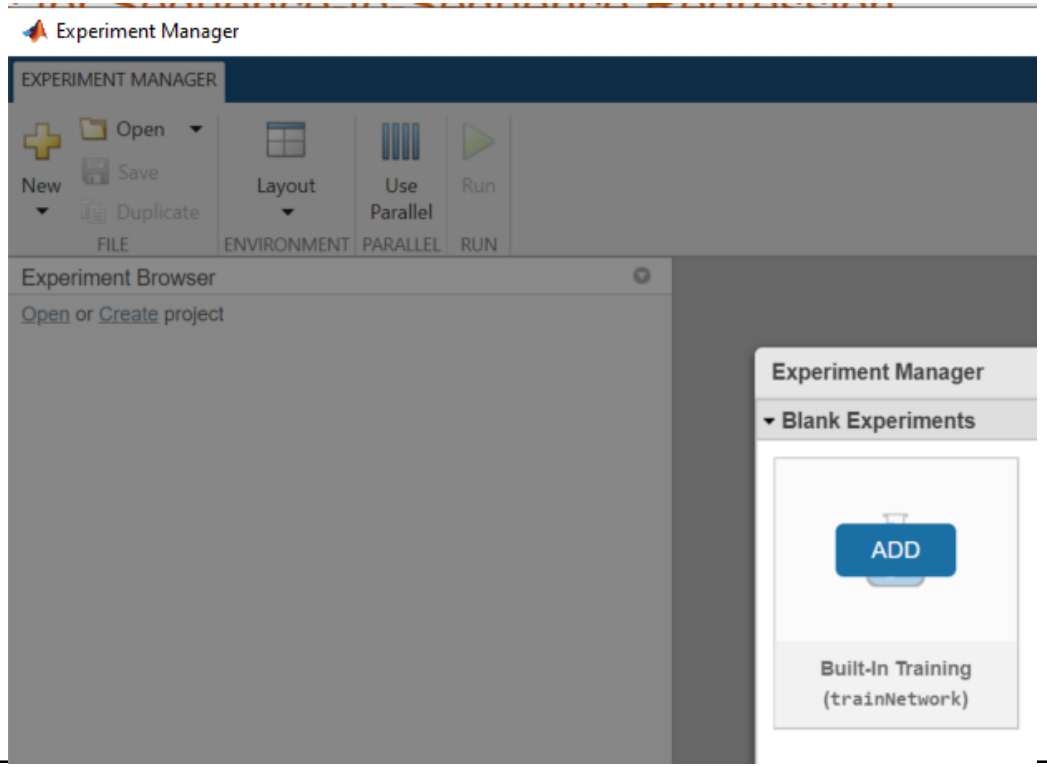
- Der Experimente Manager führt vereinfacht gesprochen eine Vielzahl von Optimierungsläufen mit abweichenden Hyperparametern durch und speichert Ergebnisse und Setting.
- Das Netz mit dem geringsten Validierungs-Loss wird final ausgewählt.
- Der Suchraum der Hyperparameter muss hierfür ex ante definiert warden.

Vorgehensweise Schritt 1/2: Anlegen eines neuen Projektes/ Experimentes

Schritt 1: App→ Experiment Manager

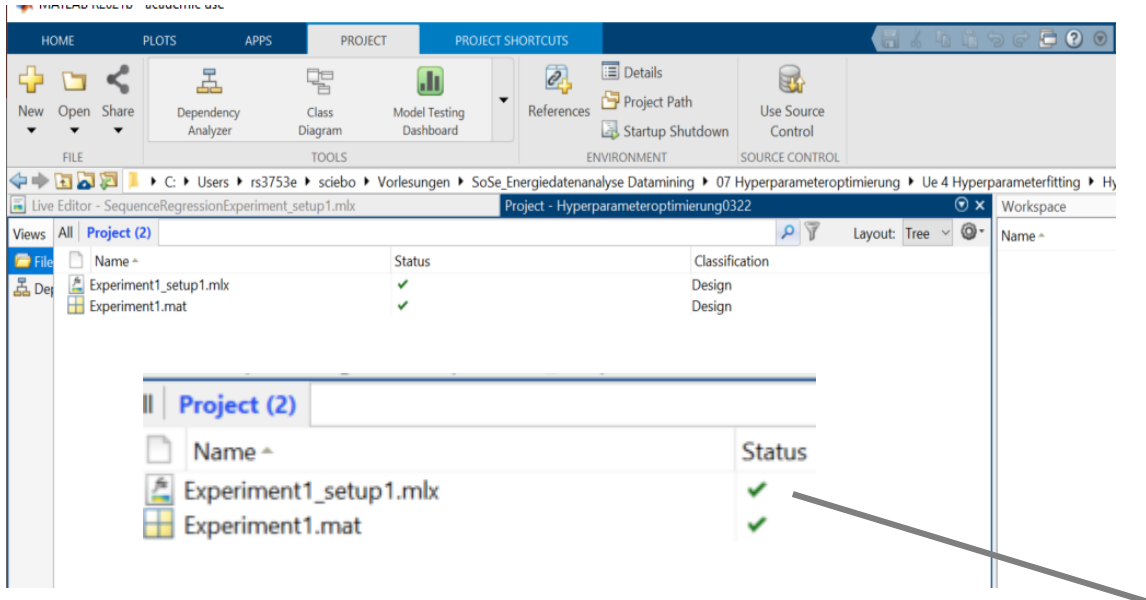


Schritt 2: New → Add Built-in Training



Im Projektordner ist ein leeres mlx-file als function eingebunden mit Übergabeparameter params

Sicht des Projektordners



Zuordnung von Files zum Projekt

Inhalt des Experiment Files

Experiment1_setup1.mlx

Built-In Training Experiment Using `trainNetwork`

Use this setup function to define the training data, network architecture, and training options for an experiment. Experiment Manager uses the outputs of this function to call the `trainNetwork` function. For more information, see Configure Built-In Training Experiment.

Input

- `params` is a structure with fields from the Experiment Manager hyperparameter table.

Output

- `trainingData` is a datastore, numeric array, cell array of numeric arrays, or table used to store the training data.
- `layers` is a layer graph that defines the neural network architecture.
- `options` is a `trainingOptions` object.

```
1  function [trainingData,layers,options] = Experiment1_setup1(params)
2  end
```

Im Setup-file werden die übergebenen Hyperparameter des jeweiligen Laufes in ein layers-objekt sowie Trainingsoptions zusammengefasst

```
function [x_train,ytrain,layers,options] = Experiment1_setup1(params)
load winddaten.mat
x_train= …
Ytrain =…;
…
…
for i=1:params.myLayerAnz
                layers = [ layers
                fullyConnectedLayer(params.myNumHiddenUnits)
                tanhLayer]
end
…
…
```

Beispiel für variable Netzarchitektur

```
options = trainingOptions(params.myoptimizer,...
                'InitialLearnRate',params.myInitialLearnRate,...
                "GradientThreshold",params.myThreshold,...
                "GradientDecayFactor",params.myGradientDecayFactor,...
                "SquaredGradientDecayFactor",params.mySquaredGradientDecayFactor,...
end
```

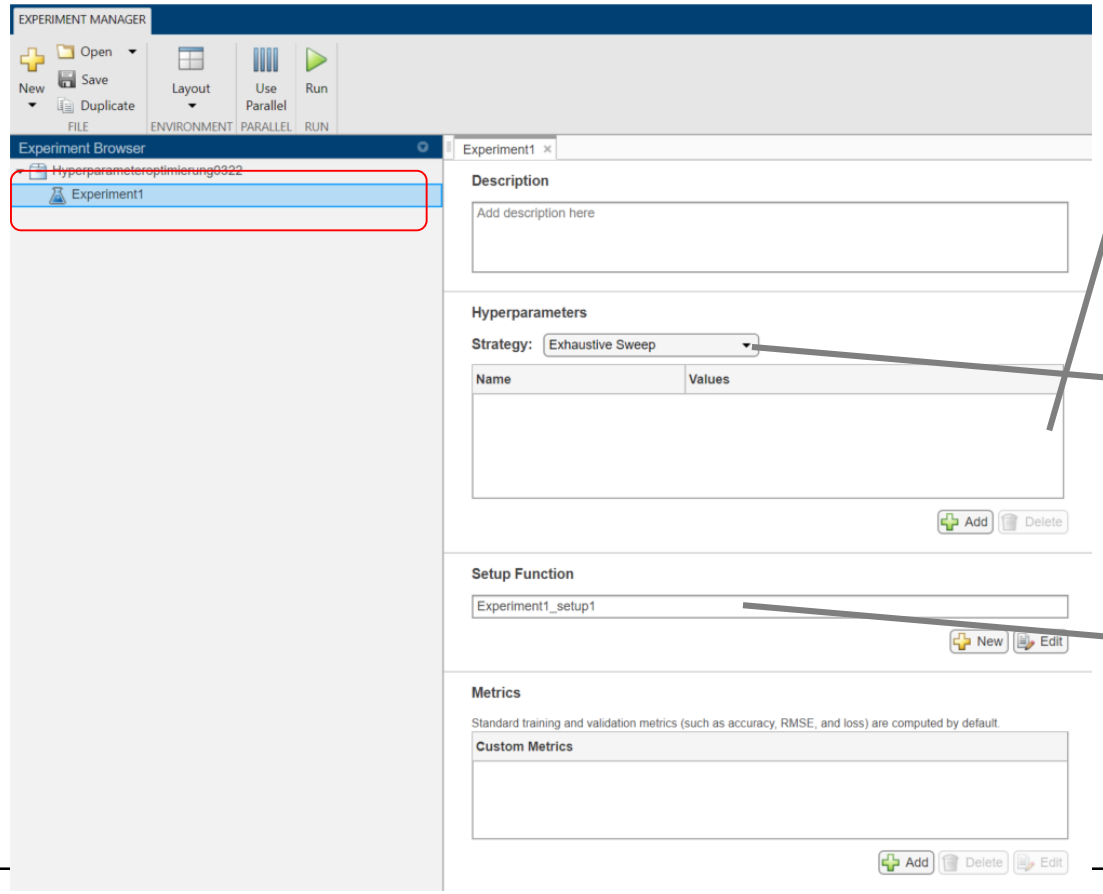
Beispiel für variable Hyperparameter des Optimierers

Inhalt

- **Input**
  - params is a structure with fields from the Experiment Manager hyperparameter table.
- **Output**
  - trainingData is a datastore, numeric array, cell array of numeric arrays, or table used to store the training data.
  - layers is a layer graph that defines the neural network architecture.
  - options is a trainingOptions object.

Im Experimente Manager können die globalen Festlegungen der Parametersuche definiert werden

Sicht Experiment-Managers



Inhalt des Experiment-Managers

Name	Values	
myThreshold	[1 1.5]	Numerical-Array
myoptimizer	["adam"]	String-Array
myLayerAnz	2:4	Sequence

Variablenbezeichnung

Add Delete

Exhaustive Sweep

Exhaustive Sweep
Bayesian Optimization

Rekombination aller Variablenbereiche
Globales Optimierungsverfahren

Verweis auf das file, welches mit dem jeweiligen Hyperparametersetting aufgerufen und das neuronale Netz erstellt wird.

Im Ergebnis werden die einzelnen Durchläufe gespeichert und die optimale Ausgestaltung der Hyperparameter ausgewählt werden kann

Experiment1 | Experiment1 | Result5 | Experiment1 | Result3 | Experiment1 | Result2 | Experiment1 | Result1 | Experiment1 | Result4

Exhaustive Sweep Result

Experiment1
(View Experiment Source)

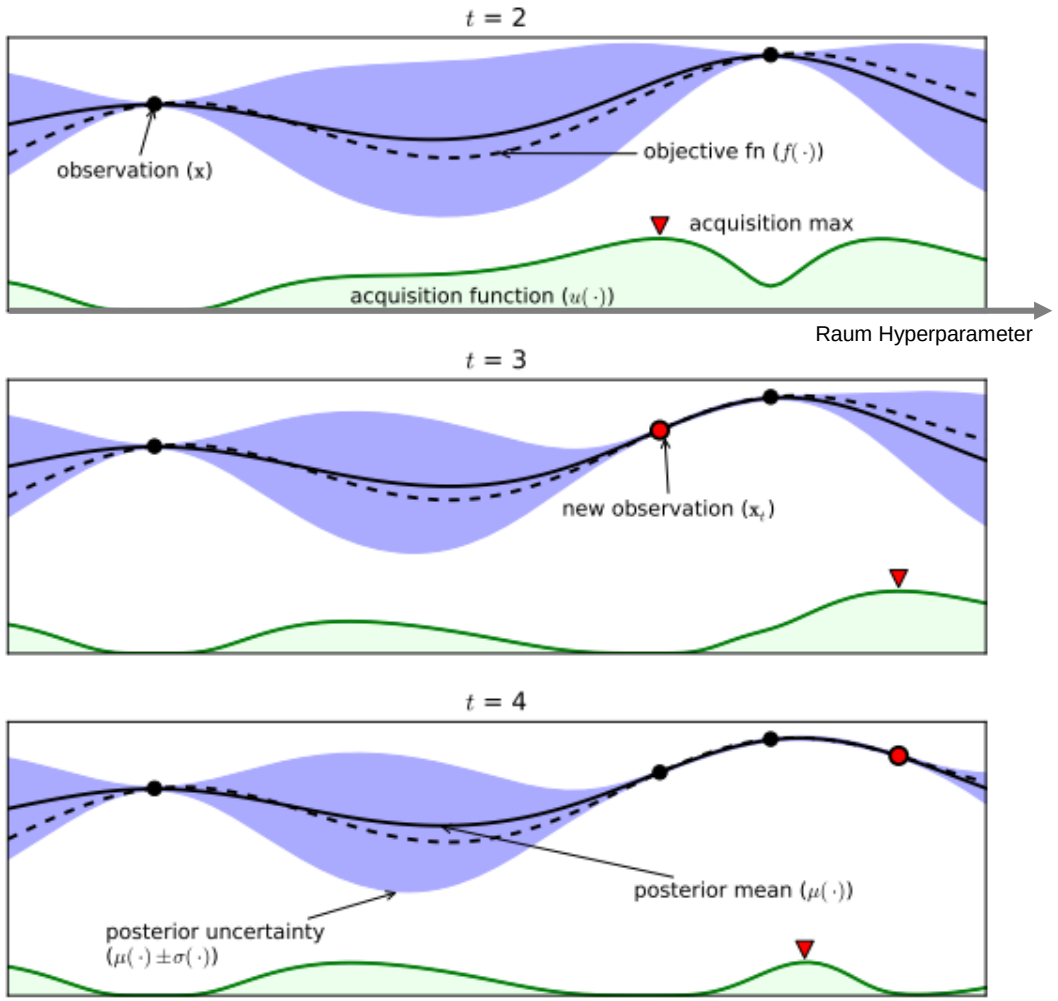
Start: 23.6.2020, 07:30:40

144/144 Trials

Complete 141 | Running 0 | Stopped 3 | Queued 0 | Error 0 | Canceled 0

Trial	Status	Progress	Elapsed Time	myInitialLearn...	myThreshold	myoptimizer	myLayerAnz	myNumHidde...	myGradientD...	mySquaredGr...	Training RMSE	Training Loss	Validation RM...	Validation Loss
1	Stopped	16.0%	0 hr 0 min 18 sec	0.0010	1.0000	adam	2.0000	50.0000	0.8000	0.9500	0.2937	0.0431	0.2864	0.0410
2	Complete	100.0%	0 hr 3 min 1 sec	0.0030	1.0000	adam	2.0000	50.0000	0.8000	0.9500	0.0963	0.0046	0.0984	0.0048
3	Complete	100.0%	0 hr 3 min 11 sec	0.0010	1.5000	adam	2.0000	50.0000	0.8000	0.9500	0.0964	0.0046	0.1006	0.0051
4	Complete	100.0%	0 hr 3 min 1 sec	0.0030	1.5000	adam	2.0000	50.0000	0.8000	0.9500	0.0961	0.0046	0.1011	0.0051
5	Complete	100.0%	0 hr 4 min 27 sec	0.0010	1.0000	adam	3.0000	50.0000	0.8000	0.9500	0.0982	0.0048	0.1027	0.0053
6	Complete	100.0%	0 hr 5 min 11 sec	0.0030	1.0000	adam	3.0000	50.0000	0.8000	0.9500	0.1010	0.0051	0.0967	0.0047
7	Complete	100.0%	0 hr 5 min 5 sec	0.0010	1.5000	adam	3.0000	50.0000	0.8000	0.9500	0.0982	0.0048	0.1030	0.0053
8	Complete	100.0%	0 hr 5 min 19 sec	0.0030	1.5000	adam	3.0000	50.0000	0.8000	0.9500	0.1000	0.0050	0.0991	0.0049
9	Complete	100.0%	0 hr 5 min 32 sec	0.0010	1.0000	adam	4.0000	50.0000	0.8000	0.9500	0.0968	0.0047	0.1020	0.0052
10	Complete	100.0%	0 hr 1 min 5 sec	0.0030	1.0000	adam	4.0000	50.0000	0.8000	0.9500	0.0980	0.0048	0.1051	0.0055
11	Stopped	1.3%	0 hr 0 min 7 sec	0.0010	1.5000	adam	4.0000	50.0000	0.8000	0.9500	0.7361	0.2709	0.6649	0.2211
12	Complete	100.0%	0 hr 0 min 19 sec	0.0030	1.5000	adam	4.0000	50.0000	0.8000	0.9500	0.1047	0.0055	0.1008	0.0051
13	Stopped	42.3%	0 hr 0 min 9 sec	0.0010	1.0000	adam	2.0000	100.0000	0.8000	0.9500	0.1083	0.0059	0.1103	0.0061
14	Complete	100.0%	0 hr 1 min 32 sec	0.0030	1.0000	adam	2.0000	100.0000	0.8000	0.9500	0.1168	0.0068	0.1148	0.0066
15	Complete	100.0%	0 hr 1 min 28 sec	0.0010	1.5000	adam	2.0000	100.0000	0.8000	0.9500	0.0992	0.0049	0.1039	0.0054
16	Complete	100.0%	0 hr 1 min 30 sec	0.0030	1.5000	adam	2.0000	100.0000	0.8000	0.9500	0.1190	0.0071	0.1169	0.0068
17	Complete	100.0%	0 hr 1 min 43 sec	0.0010	1.0000	adam	3.0000	100.0000	0.8000	0.9500	0.1024	0.0052	0.1071	0.0057
18	Complete	100.0%	0 hr 1 min 47 sec	0.0030	1.0000	adam	3.0000	100.0000	0.8000	0.9500	0.1317	0.0087	0.1248	0.0078
19	Complete	100.0%	0 hr 1 min 45 sec	0.0010	1.5000	adam	3.0000	100.0000	0.8000	0.9500	0.0994	0.0049	0.1062	0.0056
20	Complete	100.0%	0 hr 1 min 17 sec	0.0030	1.5000	adam	3.0000	100.0000	0.8000	0.9500	0.1010	0.0051	0.1151	0.0066
21	Complete	100.0%	0 hr 1 min 54 sec	0.0010	1.0000	adam	4.0000	100.0000	0.8000	0.9500	0.1104	0.0061	0.1090	0.0059
22	Complete	100.0%	0 hr 1 min 29 sec	0.0030	1.0000	adam	4.0000	100.0000	0.8000	0.9500	0.0953	0.0045	0.1499	0.0112
23	Complete	100.0%	0 hr 1 min 32 sec	0.0010	1.5000	adam	4.0000	100.0000	0.8000	0.9500	0.1048	0.0055	0.0960	0.0046
24	Complete	100.0%	0 hr 1 min 33 sec	0.0030	1.5000	adam	4.0000	100.0000	0.8000	0.9500	0.0984	0.0048	0.1303	0.0085
25	Complete	100.0%	0 hr 0 min 59 sec	0.0010	1.0000	adam	2.0000	50.0000	0.9000	0.9500	0.0939	0.0044	0.0978	0.0048
26	Complete	100.0%	0 hr 0 min 59 sec	0.0030	1.0000	adam	2.0000	50.0000	0.9000	0.9500	0.0943	0.0044	0.0974	0.0047
27	Complete	100.0%	0 hr 0 min 56 sec	0.0010	1.5000	adam	2.0000	50.0000	0.9000	0.9500	0.0943	0.0044	0.0981	0.0048
28	Complete	100.0%	0 hr 0 min 59 sec	0.0030	1.5000	adam	2.0000	50.0000	0.9000	0.9500	0.0928	0.0043	0.0970	0.0047
29	Complete	100.0%	0 hr 1 min 10 sec	0.0010	1.0000	adam	3.0000	50.0000	0.9000	0.9500	0.0949	0.0045	0.0993	0.0049
30	Complete	100.0%	0 hr 1 min 8 sec	0.0030	1.0000	adam	3.0000	50.0000	0.9000	0.9500	0.0929	0.0043	0.0978	0.0048
31	Complete	100.0%	0 hr 1 min 7 sec	0.0010	1.5000	adam	3.0000	50.0000	0.9000	0.9500	0.0948	0.0045	0.0987	0.0049
32	Complete	100.0%	0 hr 1 min 8 sec	0.0030	1.5000	adam	3.0000	50.0000	0.9000	0.9500	0.0930	0.0043	0.0976	0.0048
33	Complete	100.0%	0 hr 1 min 19 sec	0.0010	1.0000	adam	4.0000	50.0000	0.9000	0.9500	0.0924	0.0043	0.0968	0.0047
34	Complete	100.0%	0 hr 1 min 20 sec	0.0030	1.0000	adam	4.0000	50.0000	0.9000	0.9500	0.0927	0.0043	0.0970	0.0047

Exkurs: „Bayesian Optimization“ ist ein globales Optimierungsverfahren, um das Optimum einer komplexen Funktion ohne Verwendung der Ableitung bzw. Kenntnis des exakten Funktionsverlaufs



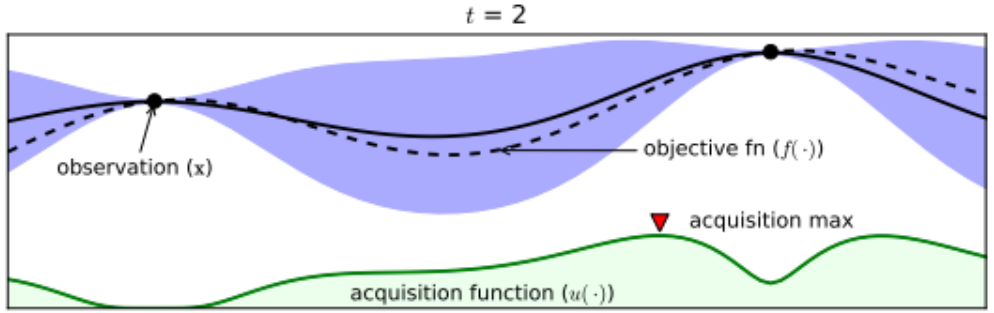
Erläuterung

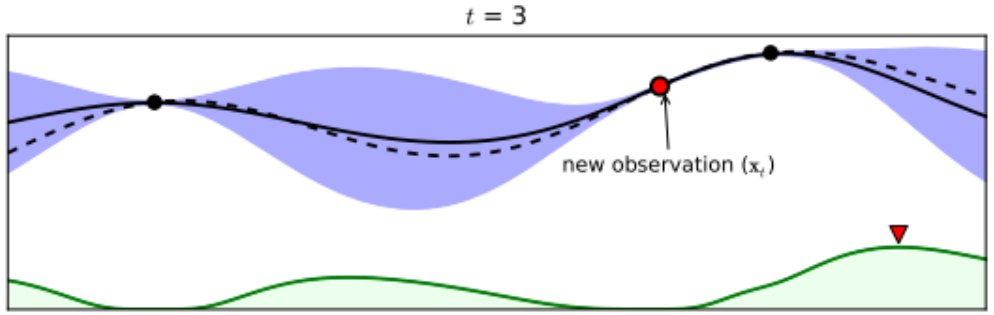
- Annahme: die Verlustfunktion f^* ist abhängig von der optimalen Einstellung der Hyperparameter x: $f(x)$
- Frage: wie erhalten wir die Hyperparametereinstellung x für welche die unbekannte $f(x)$ minimal wird?
- An bereits vorliegenden Beobachtungen von x kennen wir f(x) exakt (links schwarze Punkte)
- Auf Basis dieser Information kann unter der Annahme eines Gausprozesses GP die a-posteriori Wahrscheinlichkeit von f bestimmt werden (links blaue Fläche $\mu(x_i) +/- \sigma(x_i)$)
- Vorteil: GP lässt sich vollständig über Erwartungswert und Kovarianzmatrix an jeder Stelle x beschreiben

Quelle: Brochu et. al(2010): A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning

* Die Verlustfunktion ist i.d.R. der MSE, welcher im Rahmen des Machine Learning anhand der variablen Lerngewichte minimiert werden soll, um die Beobachtungen bestmöglich zu rekonstruieren. Wie wissen **ABER** nicht wie sich eine Veränderung der Hyperparameter auf die Verlustfunktion auswirkt, da diese aus Sicht des Algorithmus Parameter und nicht Lerngewichte darstellen. Die funktionale Form f in Abhängigkeit der Hyperparameter ist **unbekannt** aber stetig.

Exkurs: „Bayesian Optimization“ ist ein globales Optimierungsverfahren, um das Optimum einer komplexen Funktion ohne Verwendung der Ableitung bzw. Kenntnis des exakten Funktionsverlaufs





$t = 4$

Algorithm 1 Bayesian Optimization

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}}\, u(\mathbf{x}|\mathcal{D}_{1:t-1})$.
3: Sample the objective function: $y_t = f(\mathbf{x}_t) + \varepsilon_t$.
4: Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.
5: **end for**

Erläuterung

- Zielsetzung: Es sollen möglichst effizient weitere Punkte von x mit $f(x)$ ausgewertet werden
- Hierzu wird die a-posteriori Wahrscheinlichkeit von f betrachtet $p(f|x_{beobachtet})$, diese ist charakterisiert durch:
  a. Erwartungswert $E[f(x)]$ (schwarze Linie)
  b. Unsicherheit über den Verlauf von f (blaue Fläche bzw. Kovarianz)
- Für die weitere Untersuchung sind Punkte mit möglichst hohen Erwartungswert (Exploitation) und/oder hoher Unsicherheit (Exploration) für $f(x)$ interessant
- In einer speziellen Funktion („Aquisition“) werden beide Zielrichtungen gegeneinander abgewogen und in einer Zahl verdichtet (grüne Kurve)
- Im Maximum (rotes Dreieck) kann der nächste auszu-wertende Hyperparameterkonstellation abgeleitet werden.
- Daraus erfolgt eine Anpassung der a-posteriori Wahrscheinlichkeit und die Suche des nächsten Punktes (t=3)

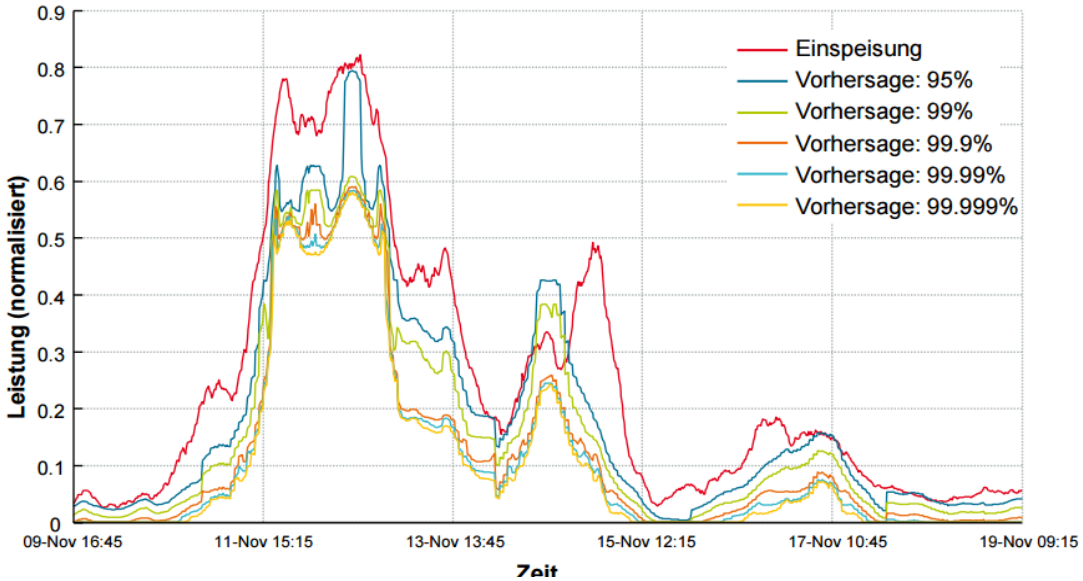
Quelle: Brochu et. al(2010): A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning

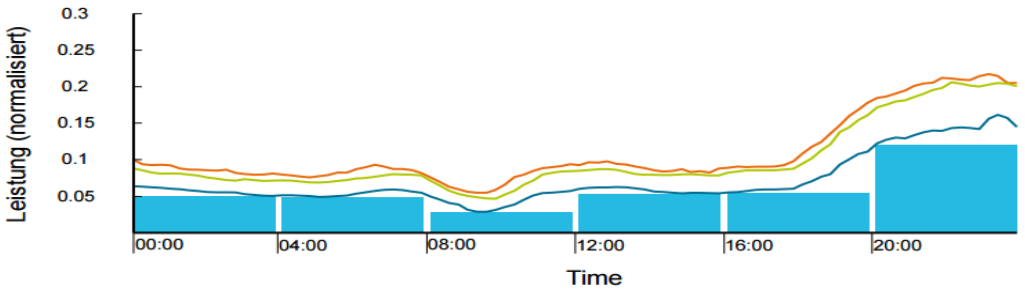
1 Optimierungskernels

2 Experimente-Manager

3 Quantilsregression

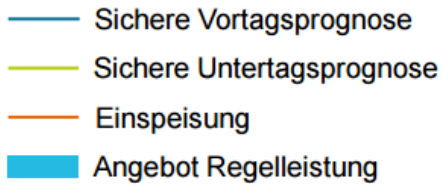
Probabilistische Prognosen eignen sich besonders für die Regelenergievermarktung





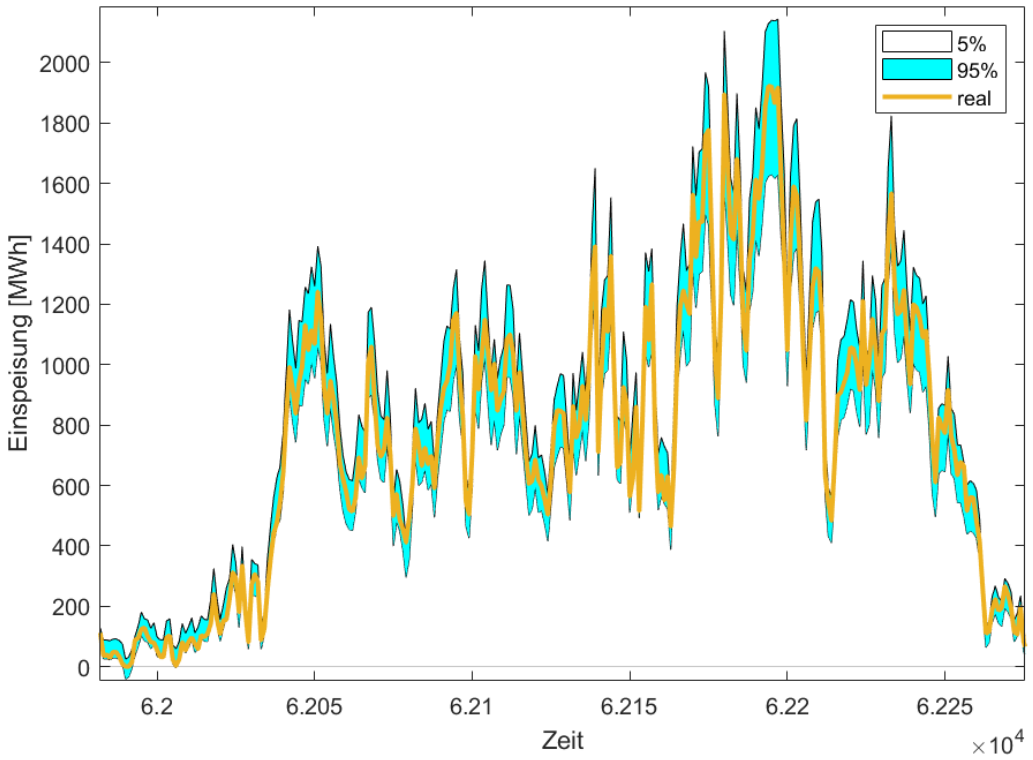
Regelenergievermarktung

- Grundlage: Einsatz probabilistischer Einspeiseprognosen
- Berücksichtigung einer Vorhersagegenauigkeit zu einem definierten Quantil
- Ableitung einer Mindesteinspeisung als Grundlage der RE-Vermarktung



Mit Hilfe von neuronalen Netzen können auch die Quantile prognostiziert werden

Prognose der Quantile

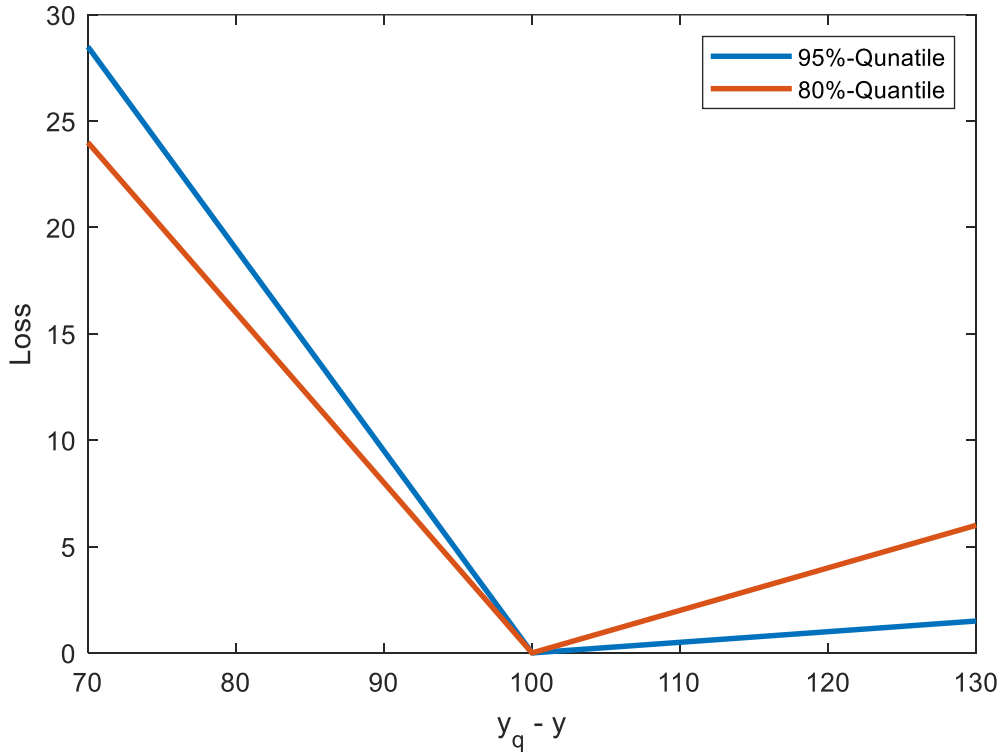


Erläuterung

- Neben der Punktprognose kann mit Hilfe eines neuronalen Netzes auch eine Intervallschätzung vorgenommen werden.
- Als Ziel soll nun ein Intervall prognostiziert werden, das X-% Sicherheit den realisierten Wert abdeckt (typischweise 90%)
- Output des Neuronalen Netzes sind nun zwei Werte (z.B.)
  - 05% Quantil
  - 95% Quantil
- Für die Berechnung des 05% bzw. 95% Quantils muss zuerst die Verlustfunktion geeignet angepasst werden.

Für die Prognose der Quantile muss die Zielfunktion angepasst werden

Pinball Loss



Erläuterung

- Hinweis: ein 90% Quantilsvorhersage bedeutet, dass der Vorhersagewert durch eine Beobachtung nur in 10% der Fälle überschritten wird.
- Die Verlustfunktion gewichtet in Abhängigkeit eines definierten Quantils [%] Über- und Unterschätzung asymmetrisch:
- $$L_{q,t}(y_t, y_{q,t}) = \begin{cases} (1-q) \cdot (y_{q,t} - y_t) & y_{q,t} > y_t \\ q(y_t - y_{q,t}) & y_{q,t} \leq y_t \end{cases}$$
  - y_t = realisierter Wert
  - $y_{q,t}$ = prognostizierter Quantilswert
- Je höher das Quantile gewählt wird, desto größer ist die Steigung im Bereich $y_{q,t} \leq y_t$ und niedriger ist die Steigung im Bereich $y_{q,t} > y_t$.
- Hierdurch werden Überschätzungen wenig und Unterschätzungen stark bestraft.

Für eine Differenzierbarkeit muss die Pinballfunktion leicht abgewandelt werden

Huber Norm

$$H(y_t, \hat{y}_t^q) = \begin{cases} \dfrac{(\hat{y}_t^q - y_t)^2}{2\varepsilon} & 0 \leq |\hat{y}_t^q - y_t| \leq \varepsilon \\ |\hat{y}_t^q - y_t| - \dfrac{\varepsilon}{2} & |\hat{y}_t^q - y_t| > \varepsilon, \end{cases}$$

$$L_{q,t}(y_t, \hat{y}_t^q) = \begin{cases} (1-q)h(y_t, \hat{y}_t^q) & \hat{y}_t^q \geq y_t \\ qh(y_t, \hat{y}_t^q) & \hat{y}_t^q < y_t. \end{cases}$$

Erläuterung

- An der Stelle $y_{q,t} = y_t$ ist die Funktion nicht stetig
- Zur Vermeidung kann der Pinball-Loss in die „Huber Norm“ transformiert werden

In Matlab muss eine neue Verlustfunktion und ein Outputlayer kreiert werden

Eigenständige Formulierung Verlustfunktion

```
classdef myqLayer < nnet.layer.RegressionLayer

methods

function loss = forwardLoss(layer, Y, T) % Layer forward loss function goes here.

% Calculate Pinball -Loss
N = size(Y,3);eps =0.01; q=[.05 0.95];

% Huber Norm
h=@(Y,T) (abs(Y-T)-eps/2).*(abs(Y-T)>eps) +
((Y-T).^2/(2*eps)).*(abs(Y-T)<=eps);
```

$$H(y_t, \hat{y}_t^q) = \begin{cases} \frac{(\hat{y}_t^q - y_t)^2}{2\varepsilon} & 0 \le |\hat{y}_t^q - y_t| \le \varepsilon \\ |\hat{y}_t^q - y_t| - \frac{\varepsilon}{2} & |\hat{y}_t^q - y_t| > \varepsilon, \end{cases}$$

```
% Verlustfunktion
```

$$L_{q,t}(y_t, \hat{y}_t^q) = \begin{cases} (1-q)h(y_t, \hat{y}_t^q) & \hat{y}_t^q \ge y_t \\ qh(y_t, \hat{y}_t^q) & \hat{y}_t^q < y_t. \end{cases}$$

```
loss=0;
for i=1:2
        loss = loss+ sum(...
                (1-q(1))*h(Y(i,1,:),T(i,1,:)).*(Y(i,1,:)>=T(i,1,:))+...
                 q(1)*h(Y(i,1,:),T(i,1,:)).*(Y(i,1,:)<T(i,1,:)))/N;
end
```

Einbindung in die Layerstruktur

```
KNN.layers = [ ...
sequenceInputLayer(KNN.numFeatures)
fullyConnectedLayer(KNN.numHiddenUnits)
tanhLayer % in die nächste Zeile könnte eine weitere Schicht integriert
werden
%batchNormalizationLayer
dropoutLayer(0.2)
fullyConnectedLayer(KNN.numHiddenUnits)
reluLayer
tanhLayer % in die nächste Zeile könnte eine weitere Schicht integriert
werden
dropoutLayer(.2)
fullyConnectedLayer(KNN.numHiddenUnits)
tanhLayer % in die nächste Zeile könnte eine weitere Schicht integriert
werden
fullyConnectedLayer(KNN.numResponses)
myqLayer('qoutput')];
```