TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät für Elektro- und Informationstechnik, Professur für Grundlagen der Elektrotechnik und Elektronik

FliK Modul 2020

Regularization

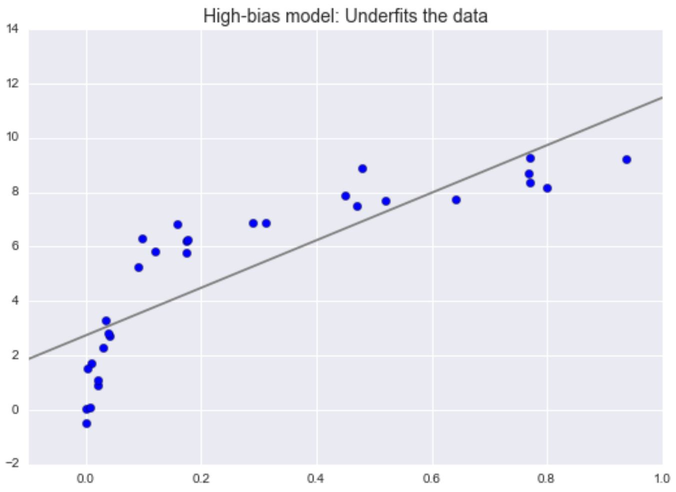
Steffen Seitz & Marvin Arnold

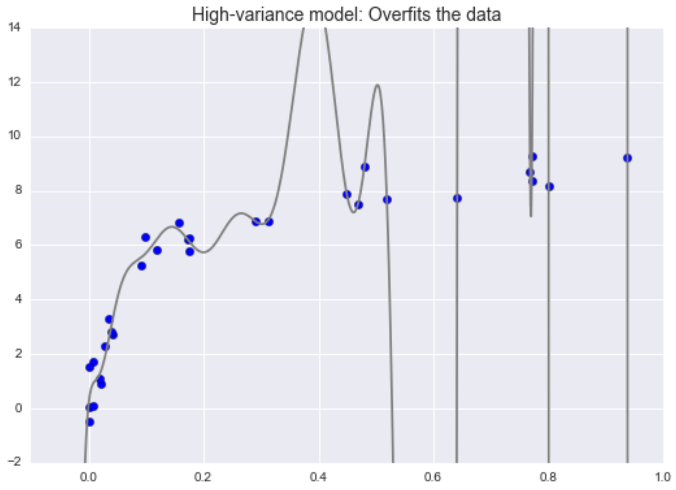
Prof. Ronald Tetzlaff

Dresden, 19-23.10.

Over and Underfitting

Neural Networks tend to over- or underfit the data.

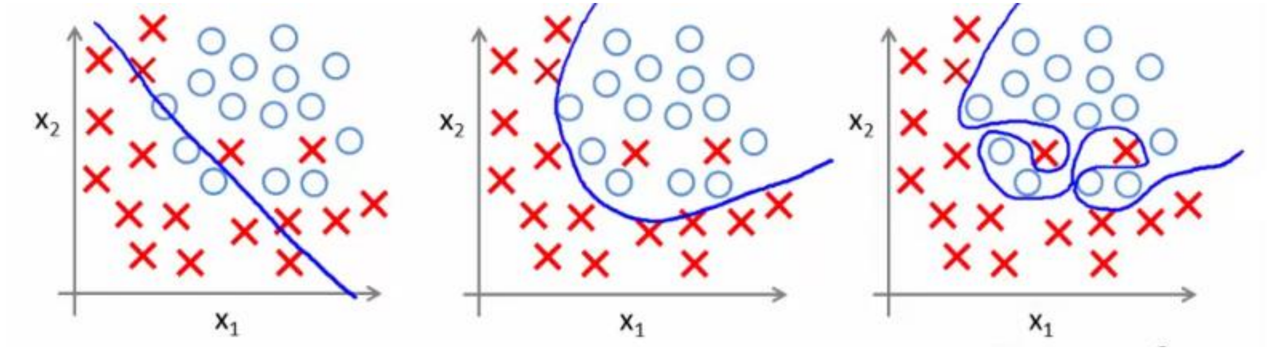




Model is missing flexability
→ The model is **not** able to **model the data** no matter how much training sampels exist.
→ The result of the model will be straight up bad

Model fit has to much flexibility
→ Seems to pay to much attention **random effects rather than** the **intrinsic properties** of whatever process generated that data
→ The results lack of generalizsation and are therefore **misleading**

Over and Underfitting



High Bias
(Underfitting)

High Variance
(Overfitting)

The question of "the best model" is about **finding** a **sweet spot** in the tradeoff between bias and variance.

Weight Decay
L1 & L2 Regularization

Overfitting is reflected as **very high weights** between particular neurons. A simple trick to counteract this phenomenon is to **penalize unregulated weight** growth in the loss/cost function directly. This is usually done by adding a regularization term. That is either linear (L1 reg.) or quadratic (L2 reg.)

Remember that while training we want to **minimize** this **cost** as much as we can!

L1 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{MSE}} + \lambda \boxed{\sum_{j=0}^{M}|W_j|}$$

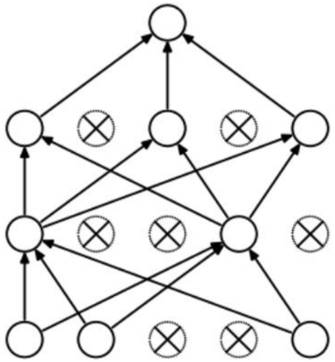
L2 Regularization

$$\text{Cost} = \underbrace{\underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{MSE}}}_{\text{Loss function}} + \lambda \underbrace{\boxed{\sum_{j=0}^{M}W_j^2}}_{\text{Regularization Term}}$$

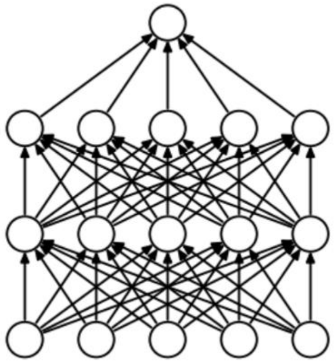
These terms will blow up if the sum of all weights gets to high.

Dropout

Another way to prevent unregulated weight growth is the **Dropout** method. Dropout is a technique where **randomly** selected **neurons** are **ignored** during **training**. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any **weight updates are not applied** to the neuron on the backward pass.



while Training



while testing

```
keras.layers.Dropout(rate, noise_shape=None, seed=None)
```

```
model=keras.models.Sequential()
model.add(keras.layers.Dense(150, activation="relu"))
model.add(keras.layers.Dropout(0.5))
```

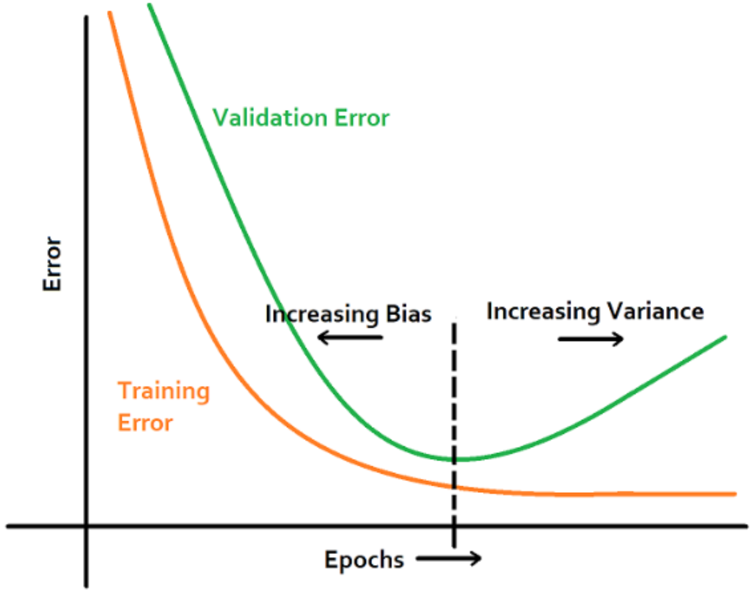
This technique can be viewed as **adding noise** to the network. Despite even the **authors** of the paper **don’t know** why this actually works. Most importantly Dropout is only used during the training of a model only and is not used when evaluating the model.

Early Stopping

Another sign of overfitting is the **rise** of the **validation error while training** the network. It is not an suprising behavior because the definition of overfitting is that overfitted models achieve worse.

With this plot it is possible to very effectively **monitor the whole training process** and determine the moment, timestamp the model stopped learning useful features.

If this point is moinitored, **Early Stopping** is defined as an **automatic** early **end** of the **training,** if the **validation error keep increasing** for a specific time period.



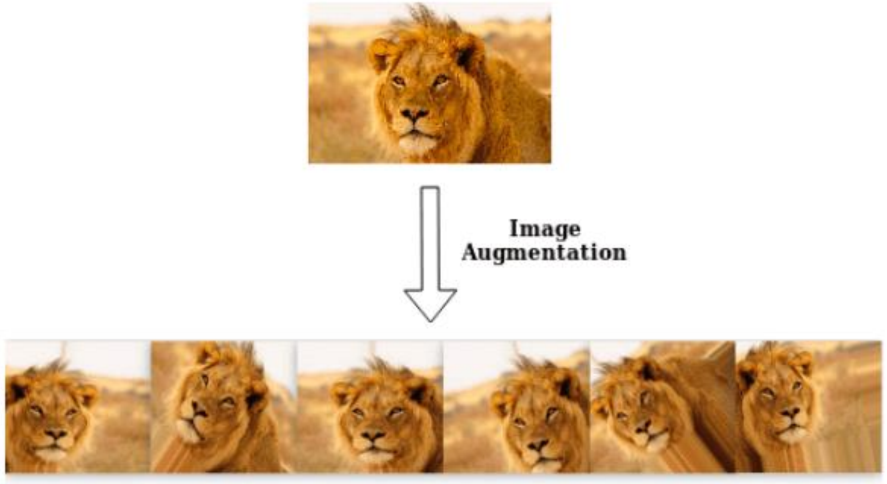

```
EarlyStop = EarlyStopping(monitor='val_loss', mode='min')
model = model.fit(trainX, trainy, validation_data=(testX, testy), epochs=4000, callbacks=[EarlyStop])
```

Keras Callbacks: Once we started the training in Keras, we are not able to access the process. Callbacks are the only a way to tell Keras to run other predefined functions while doing the training loops.

Data Argumentation

A very „cheap" way to address overfitting is to **create argumented data** based on real sampels.
This technique can be used for both NLP (time series) and CV (images).

In CV we can use the techniques like **Jitter**, **PCA** and **Flipping**. Similarly in NLP we can use the techniques like Synonym Replacement, Random Insertion, Random Deletion and Word Embeddings.



5. Exercise

Let‘s regularize our first classifier with Keras!