CME 216, ME 343 - Winter 2021

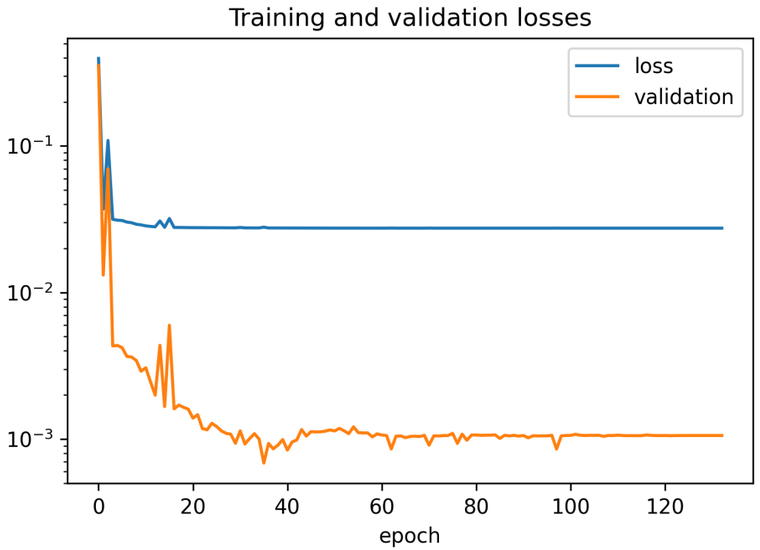
Eric Darve, ICME

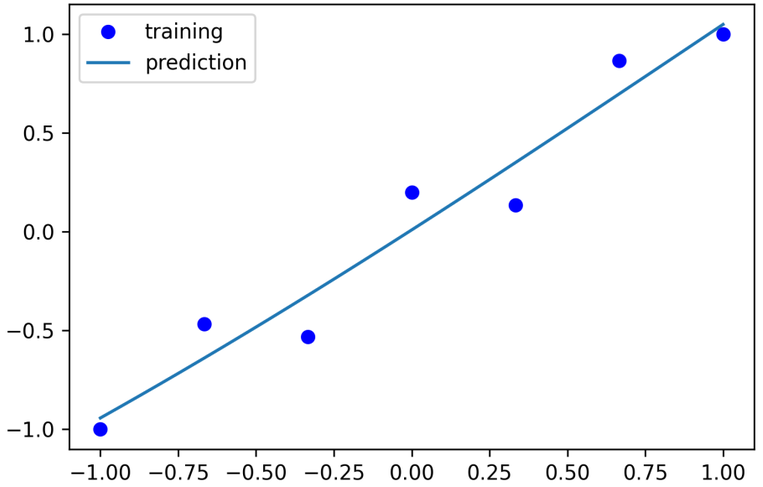
One simple method to avoid overfitting is to reduce the complexity of the model.

We chose a very simple example to illustrate these concepts.

To avoid overfitting in this case, we need to use a single `tanh` function (one output node in the hidden layer).

Then we get:


Training and validation losses
loss
validation
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
120
epoch


training
prediction
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

Simplifying the model may reduce the issue of overfitting but this does not imply that we will necessarily get the right solutions.

By restricting the structure of the DNN we are optimizing, we may oversimplify and end up with a DNN that cannot accurately reproduce the solution.

Another approach consists in keeping the same DNN but using regularization.

This means adding a new term to the loss function that penalizes large weights and biases.

Regularization is usually done using either l1 or l2 regularization.

Consider some training data (x_i, y_i) and a mean squared error loss function.

With l2 regularization, we define the loss as

$$\text{Loss} = \sum_i (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{ijl} (w_{ij}^{(l)})^2 + \lambda_2 \sum_{il} (b_i^{(l)})^2$$

λ_1 and λ_2 are the regularization factors

$w_{ij}^{(l)}$ is the weight (i, j) in layer l

$b_i^{(l)}$ is the bias i in layer l.

In TensorFlow/Keras the syntax is:

- `kernel_regularizer`: regularizer to apply a penalty on the layer's weights (kernel)
- `bias_regularizer`: penalty on the layer's bias
- `activity_regularizer`: penalty on the layer's output

L2

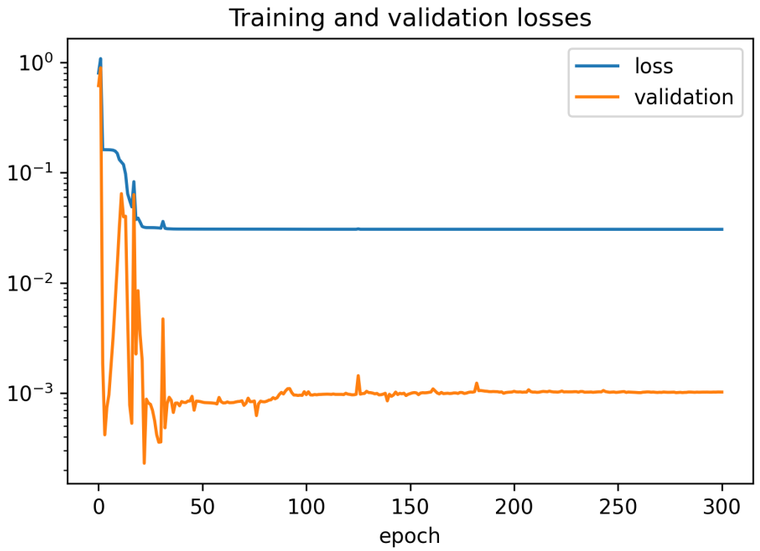
```
kernel_regularizer=tf.keras.regularizers.L2(l=1e-3)
bias_regularizer=tf.keras.regularizers.L2(l=1e-3)
```

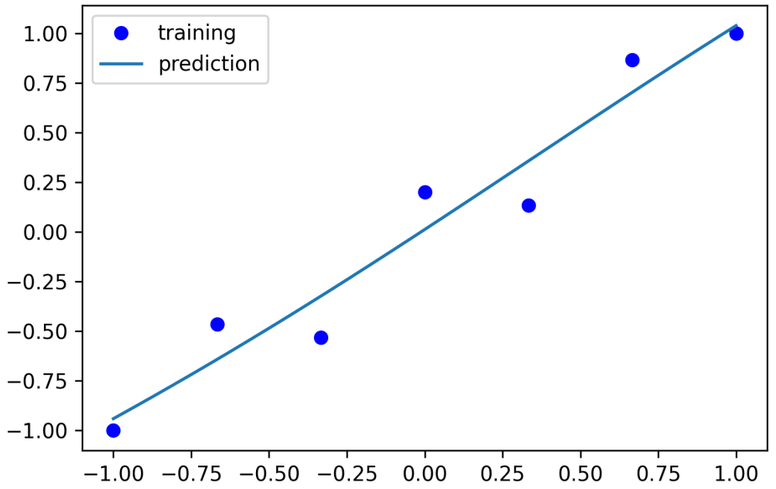
With regularization we ensure that weights are kept small and cannot diverge.

The accuracy is then much improved.

2 layers of size 128

`tf.keras.regularizers.L2(l=1e-3)`





Physically we can interpret the l2 regularization as adding a spring to the weights and biases so that they are always pulled back towards 0.

Indeed consider

$$\lambda_1 \sum_{ijl} (w_{ij}^{(l)})^2$$

Minus the gradient with respect to $w_{ij}^{(l)}$ is equal to

$$-2\lambda_1 w_{ij}^{(l)}$$

This is a spring force where the "displacement" of the spring is $w_{ij}^{(l)}$, and the stiffness $2\lambda_1$.

It prevents $w_{ij}^{(l)}$ from getting too big.

The l1 regularization uses the following formula (with the l1 norm):

$$\text{Loss} = \sum_i (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{ijl} |w_{ij}^{(l)}| + \lambda_2 \sum_{il} |b_i^{(l)}|$$

The l1 regularization has a different effect.

It makes the solution (weights and biases) sparser.

That is, it tries to reproduce the training data with as few non-zero weights and biases as possible.

L1

```
kernel_regularizer=tf.keras.regularizers.L1(l=1e-3)
bias_regularizer=tf.keras.regularizers.L1(l=1e-3)
```

L1 + L2

```
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)
bias_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)
```