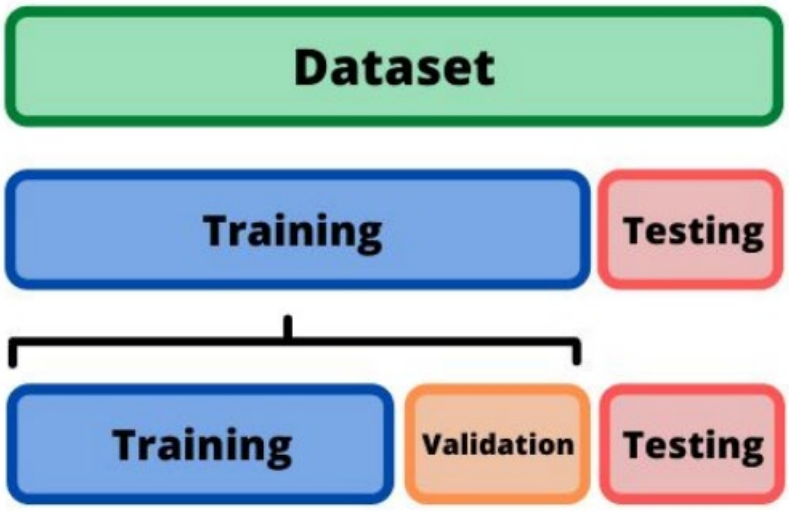
Chapter 05 - Model Training | Measuring Performance and Regularization

Author: Gianmarco Scarano

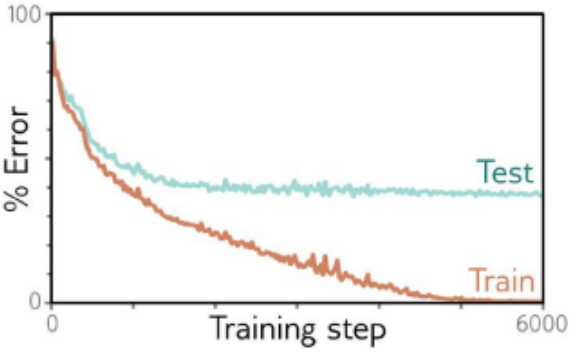
gianmarcoscarano@gmail.com

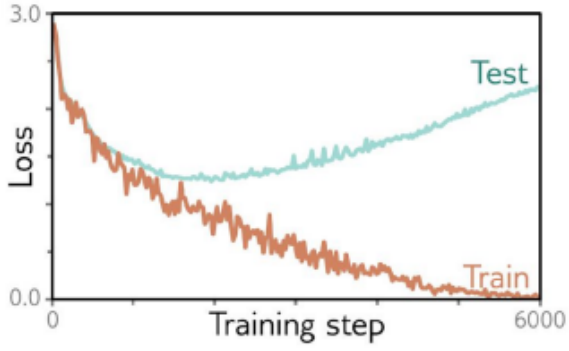
1. Measuring Performance

For measuring the performances of our model, we want to take into consideration the following scheme:



In the case of MNIST-1D dataset, these on the right are the results. They show that as for the Error rate, the training set decreases to zero, but test errors are still up to 40%. This means that the model does not generalize well to new test data. In the second image (bottom one), we are plotting the Loss as a function of the training step. The training loss decreases towards zero. Test loss decreases at first but then it keeps on increasing as the model becomes confident about its (wrong) predictions.

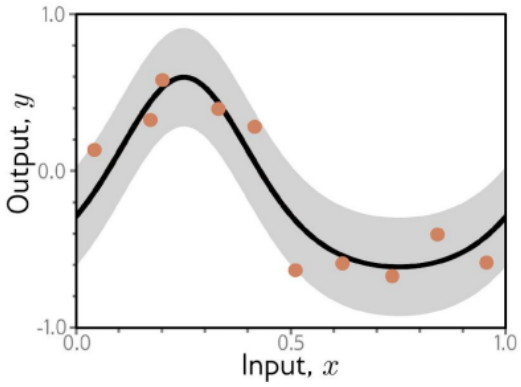




But how exactly these errors raise up?

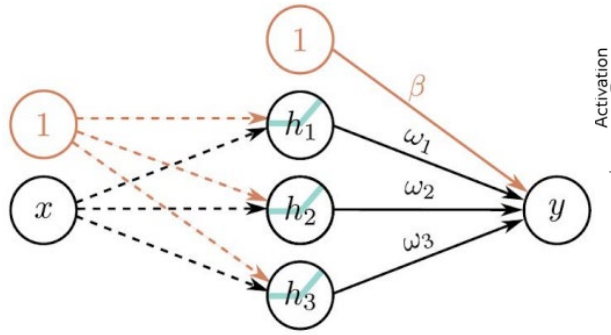
Considering a regression function, we have a X/Y plane divided in I equal segment. We plot a black line (Ground Truth for our function).

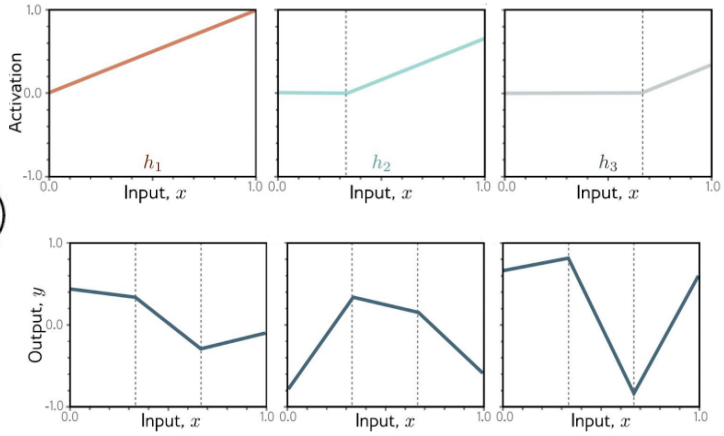
For each segment I, one sample x_i is drawn from a uniform distribution. The evaluation on the Y axis (y_i) is done by evaluating our approximated function at x_i and adding Gaussian Noise (gray region ± 2).



That's where errors raise up. From the difference between the ground truth and the evaluated y_i.

Talking about errors, thinking about a general 1 hidden layer architecture with 3 hidden units, we have weight and biases chosen such that the hidden unit activations have slope one and joints are equally spaced across the interval ($x = 0; x = 1/3; x = 2/3$)





In this representation on the right, we can observe three different representations, each with different parameters initialization.

1.1 Noise, Variance & Bias

Noise can arise from the fact that there are multiple possible valid outputs y for each input, due to stochastic elements to the data generation process or mislabeled data, etc.

Also, a second potential source of error may occur because the model is not flexible enough to fit the true function perfectly, not getting the closest possible approximation to the ground truth function. In fact, for different training datasets, we might have different results. Plus, the stochastic learning algorithm not necessarily converges to the same solution each time, adding additional variance.

Along with that, decreasing the variance corresponds to an increase in the bias and vice-versa.

Noise it's not depending on the training data but it's intrinsic to the data generation process.

In order to reduce variance, we can augment the training data (note that the prediction will be the same).

In order to reduce bias, we can increase the number of hidden units such that the model can fit the true function closely. Doing this, though, will increase the variance term. This is known as bias-variance trade-off.

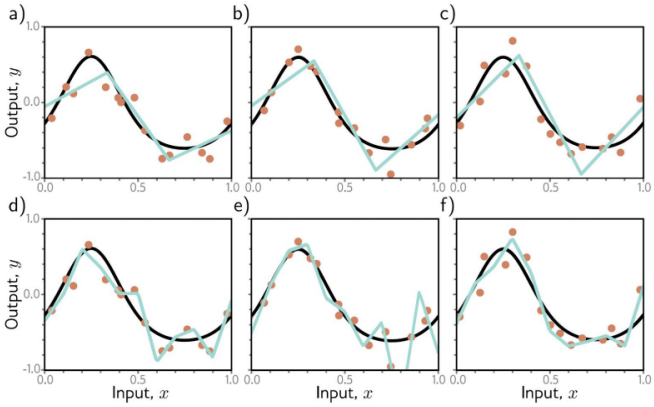


Figure 8.8 Overfitting. a–c) A model with three regions is fit to three different datasets of fifteen points each. The result is similar in all three cases (i.e., the variance is low). d–f) A model with ten regions is fit to the same datasets. The additional flexibility does not necessarily produce better predictions. While these three models each describe the training data better, they are not necessarily closer to the true underlying function (black curve). Instead, they overfit the data and describe the noise, and the variance (difference between fitted curves) is larger.

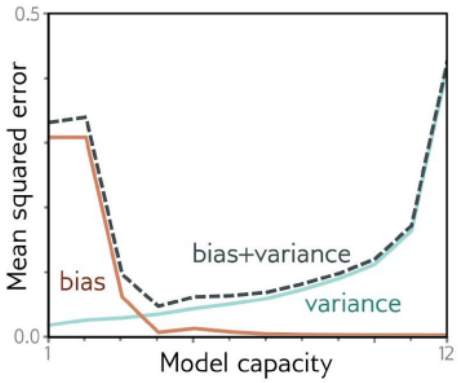
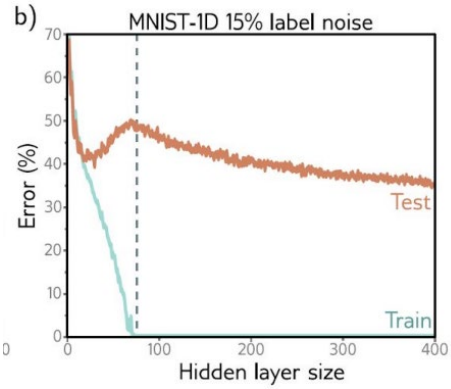
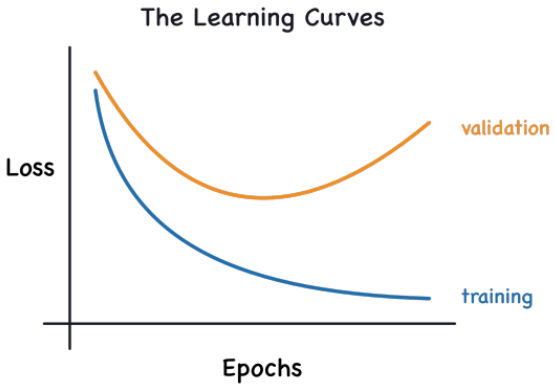


Figure 8.9 Bias-variance trade-off. The bias and variance terms from equation 8.7 are plotted as a function of the model capacity (number of hidden units / linear regions) in the simplified model using training data from figure 8.8. As the capacity increases, the bias (solid orange line) decreases, but the variance (solid cyan line) increases. The sum of these two terms (dashed gray line) is minimized when the capacity is four.

1.2 Double Descent

Double Descent problems arise when, during training, our error percentage decreases as the capacity of our model increases, but then increases again as we near the point where the training data is exactly memorized (dotted line).





In overfitting, instead, models might describe the training data better (by increasing number of regions), but they aren't able to get closer to the true underlying function (ground truth function). Instead, they overfit the data, enlarging the variance.

2. Techniques for reducing overfitting

2.1 Regularization

A regularization term λ is used to penalize larger weights in order to, hopefully, reduce errors in our model.

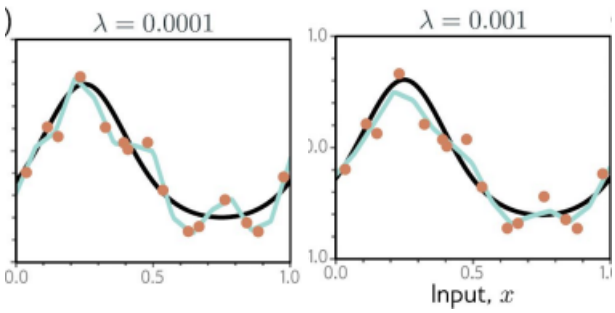
$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot g[\phi] \right]$$

g[ϕ] is a function that returns a scalar that takes a larger value when the parameters are less preferred

We have different kinds of regularization techniques. One of them is called L_2.

Weights and not biases

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \sum_{j} \phi_j^2 \right]$$



As we can see, the more we penalize (increasing the lambda value), the more fit our curve will be. Of course if we penalize a lot, our cyan line would get messy.

Regularization is also necessary when our network is overfitting or overparametrized. It will favor functions that smoothly interpolate between the nearby points.

2.2 Early Stopping

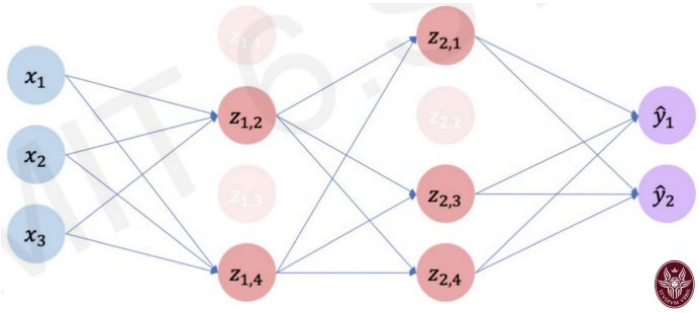
Early stopping technique is used to stop the training procedure before it has fully converged. This can reduce overfitting if the model has already captured the shape of the underlying function, but has not started yet to overfit to the noise. Performances on the validation set is monitored every T iterations, such that when we detect overfitting, we stop and load the latest useful model.

- Stop training as we detect overfitting

2.3 Dropout

During training phase, we set some weights to zero with a certain probability p (usually 0.5).

At each iteration, a random subset of hidden units is clamped to zero (deactivated). The result is that the incoming and outgoing weights from these units have no effect, namely training the network differently each time.



There is one down-side, which takes place when we are dealing with extremely few labeled training data. In this case, dropout is less effective.

2.4 Data augmentation