8 Simple Techniques to Prevent Overfitting

David Chuan-En Lin · Follow
Published in Towards Data Science · 5 min read · Jun 7, 2020

Overfitting occurs when the model performs well on training data but generalizes poorly to unseen data. Overfitting is a very common problem in Machine Learning and there has been an extensive range of literature dedicated to studying methods for preventing overfitting. In the following, I'll describe eight simple approaches to alleviate overfitting by introducing only one change to the data, model, or learning algorithm in each approach.

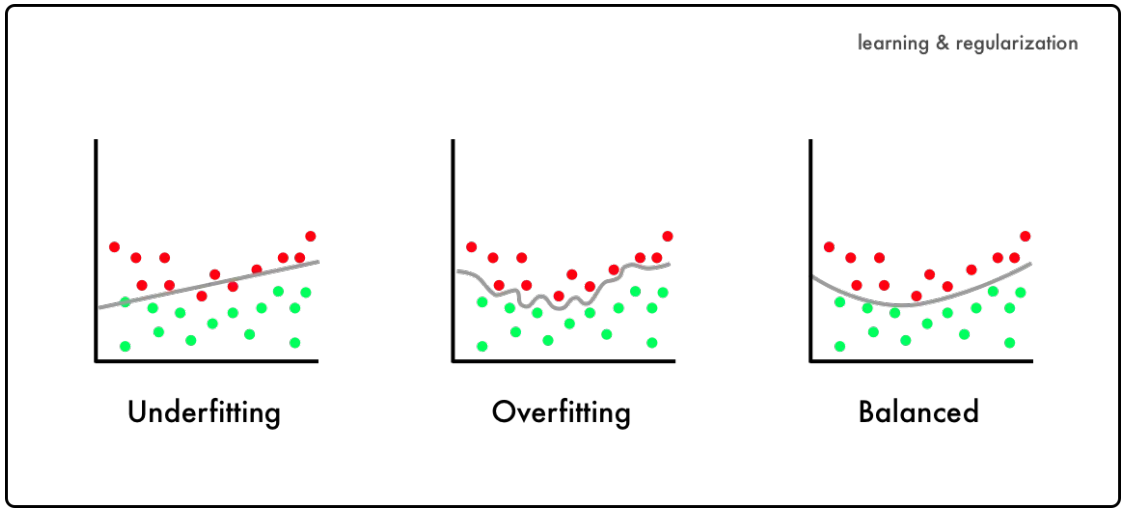


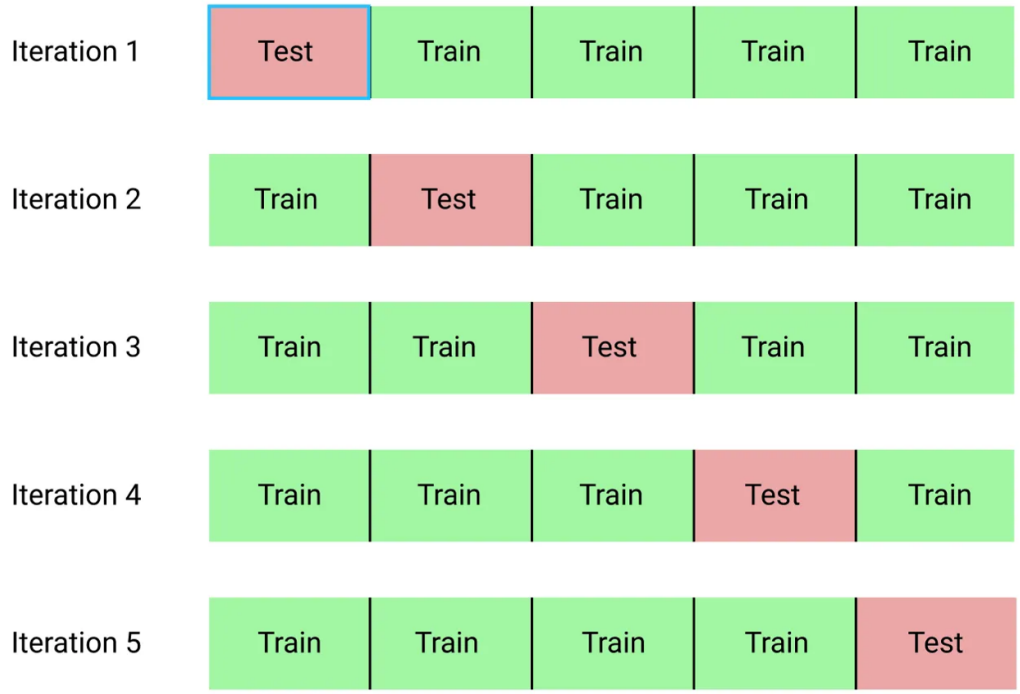
Table of Contents

1. Hold-out (data)

Rather than using all of our data for training, we can simply split our dataset into two sets: training and testing. A common split ratio is 80% for training and 20% for testing. We train our model until it performs well not only on the training set but also for the testing set. This indicates good generalization capability since the testing set represents unseen data that were not used for training. However, this approach would require a sufficiently large dataset to train on even after splitting.

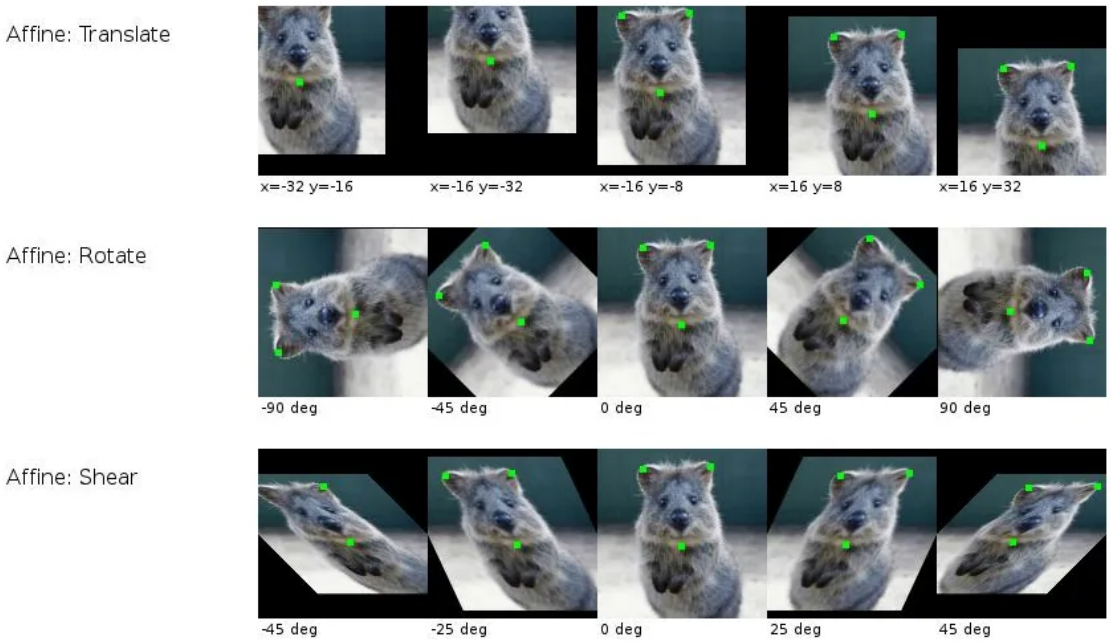
2. Cross-validation (data)

We can split our dataset into *k* groups (k-fold cross-validation). We let one of the groups to be the testing set (please see hold-out explanation) and the others as the training set, and repeat this process until each individual group has been used as the testing set (e.g., *k* repeats). Unlike hold-out, cross-validation allows all data to be eventually used for training but is also more computationally expensive than hold-out.



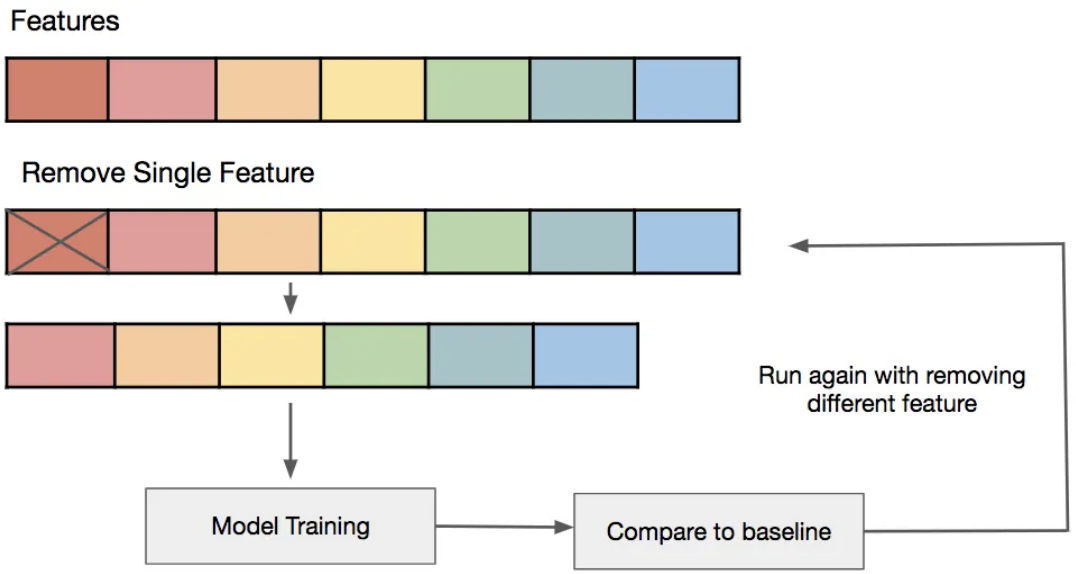
3. Data augmentation (data)

A larger dataset would reduce overfitting. If we cannot gather more data and are constrained to the data we have in our current dataset, we can apply data augmentation to artificially increase the size of our dataset. For example, if we are training for an image classification task, we can perform various image transformations to our image dataset (e.g., flipping, rotating, rescaling, shifting).



4. Feature selection (data)

If we have only a limited amount of training samples, each with a large number of features, we should only select the most important features for training so that our model doesn't need to learn for so many features and eventually overfit. We can simply test out different features, train individual models for these features, and evaluate generalization capabilities, or use one of the various widely used feature selection methods.



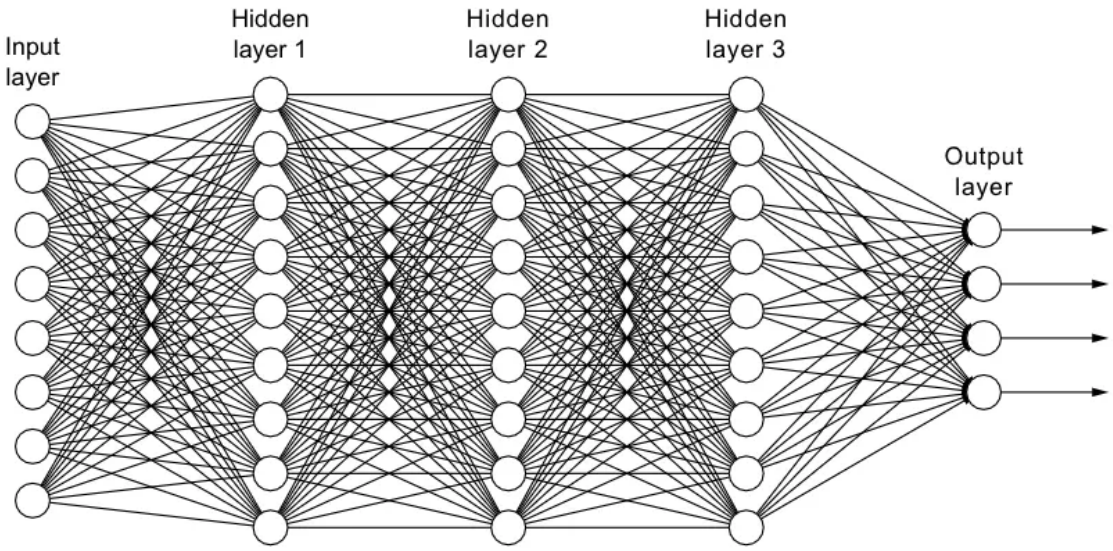
5. L1 / L2 regularization (learning algorithm)

Regularization is a technique to constrain our network from learning a model that is too complex, which may therefore overfit. In L1 or L2 regularization, we can add a penalty term on the cost function to push the estimated coefficients towards zero (and not take more extreme values). L2 regularization allows weights to decay towards zero but not to zero, while L1 regularization allows weights to decay to zero.

L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

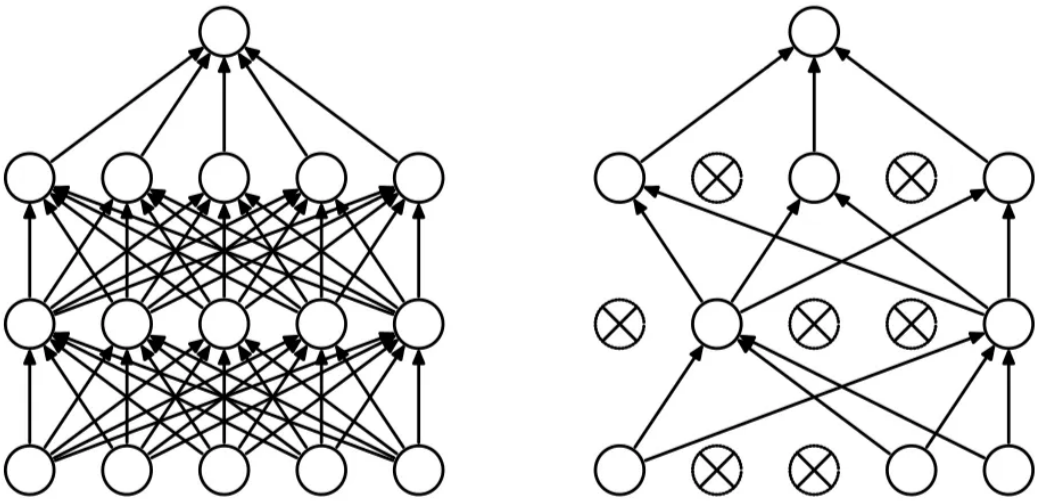
6. Remove layers / number of units per layer (model)

As mentioned in L1 or L2 regularization, an over-complex model may more likely overfit. Therefore, we can directly reduce the model's complexity by removing layers and reduce the size of our model. We may further reduce complexity by decreasing the number of neurons in the fully-connected layers. We should have a model with a complexity that sufficiently balances between underfitting and overfitting for our task.



7. Dropout (model)

By applying dropout, which is a form of regularization, to our layers, we ignore a subset of units of our network with a set probability. Using dropout, we can reduce interdependent learning among units, which may have led to overfitting. However, with dropout, we would need more epochs for our model to converge.

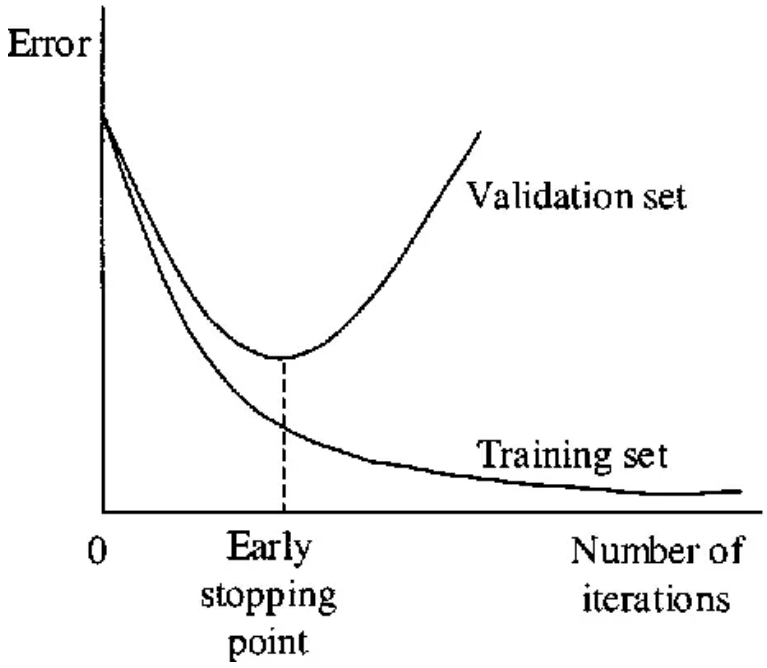


(a) Standard Neural Net (b) After applying dropout.

8. Early stopping (model)

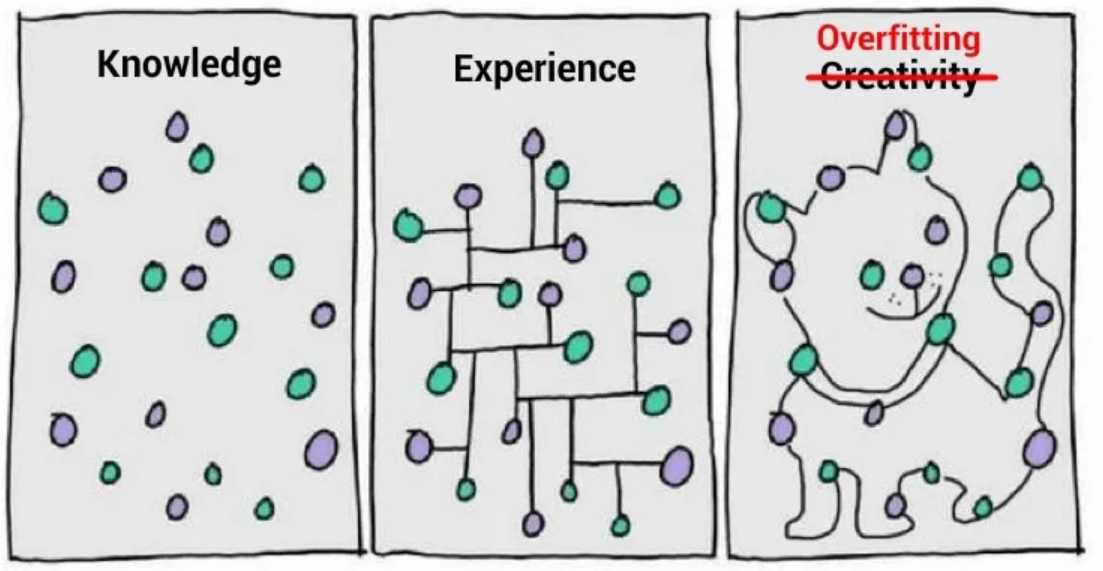
We can first train our model for an arbitrarily large number of epochs and plot the validation loss graph (e.g., using hold-out). Once the validation loss

begins to degrade (e.g., stops decreasing but rather begins increasing), we stop the training and save the current model. We can implement this either by monitoring the loss graph or set an early stopping trigger. The saved model would be the optimal model for generalization among different training epoch values.



. . .

You have reached the end of the article! Hopefully, you now have a toolbox of methods to battle overfitting ⚔️.



. . .

Hats off to you for completing this article and I hope you enjoyed it 🎩. If you're interested in more ML-related topics, also check out some interesting articles by Tim :).

Subscribe? 🙂

Machine Learning | Overfitting | Regularization | Artificial Intelligence | Data Science