Deep Learning - COSC2779/2972

Regularization

Dr. Ruwan Tennakoon

RMIT UNIVERSITY

Semester 2, 2022

Reference: *Chapter 7: Ian Goodfellow et. al., "Deep Learning", MIT Press, 2016.*

Generalization Gap

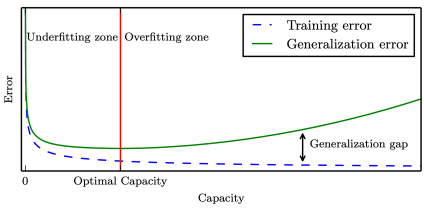


Image: Goodfellow, 2016.

Simpler functions are more likely to generalize, but a sufficiently complex hypothesis is needed to achieve low training error.

Challenge in machine learning is how to make an algorithm that will perform well not just on the training data, but also on unseen data.

$*$ It is possible for the model to have optimal capacity and yet still have a large gap between training and generalization errors. In this situation, we may be able to reduce this gap by gathering more training examples.

If you discover your model is over-fitting, you can: gather more, change the model or do regularization.

Regularization: any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

1. Designed to encode specific kinds of prior knowledge.
2. Express a generic preference for a simpler model class.
3. Make an under-determined problem determined.
4. Combine multiple hypotheses that explain the training data

Regularization of an estimator works by trading increased **bias** for **reduced variance**.

Parameter Norm Penalties

Many regularization approaches are based on limiting the capacity of models.

How can we reduce the capacity of a neural network model?

- Change the structure to have less depth, and width.
- Drive some of the weights towards zero during training (has the same affect as point 1).

Many regularization approaches are based on limiting the capacity of models.

Can achieve this by adding a penalty term on model weights to cost function.

$$\mathcal{L}(\mathbf{w}) = \underbrace{\mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} \mathrm{L}(y, h(\mathbf{x};\mathbf{w}))}_{\text{Empirical Risk}} + \lambda \mathrm{R}(\mathbf{w})$$

Larger values of λ correspond to more regularization.

$\mathrm{R}(\mathbf{w})$ is usually applied only to the weight matrix of the affine transformation at each layer and leaves the biases unregularized.

$$\mathbf{z}^{(i)} = \mathbf{W}^{(l)}\mathbf{x}^{(i)} + \mathbf{b}^{(l)} \quad \triangleright \text{Affine transform}$$

$$h^{(i)} = g\left(\mathbf{z}^{(i)}\right) \quad \triangleright \text{Activation}$$

- **L2 regularization:** Cost added is proportional to the square of the value of the weights coefficients.

$$\mathrm{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_{j=1}^{d} {w_j}^2 = \mathbf{w}^\top \mathbf{w}$$

Called weight decay and is the most common. when W is a matrix we use "Frobenius Norm".

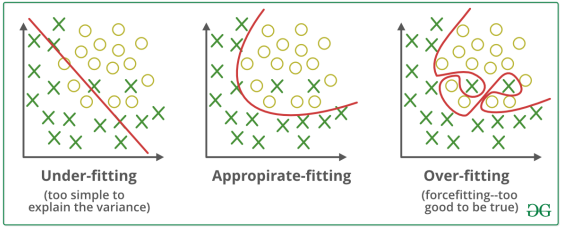
- **L1 regularization:** Cost added is proportional to the absolute value of the weights coefficients.

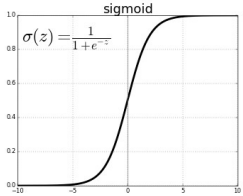
$$\mathrm{R}(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

Tend to make the weights sparse.

Why Does Regularization Work

$$\mathcal{L}(\mathbf{w}) = \underbrace{\mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} \mathrm{L}(y, h(\mathbf{x};\mathbf{w}))}_{\text{Empirical Risk}} + \lambda \mathrm{R}(\mathbf{w})$$





Parameter Norm Penalties

- L1 regularization: Cost added is proportional to the absolute value of the weights coefficients.
- L2 regularization: Cost added is proportional to the square of the value of the weights coefficients.
- The regularization hyper-parameter λ should be tuned using validation set.

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import regularizers

l2_model = tf.keras.Sequential([
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001),
                 input_shape=(FEATURES,)),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l1(0.001)),
    layers.Dense(1)
])
```

Set some probability of eliminating a node. For each batch.

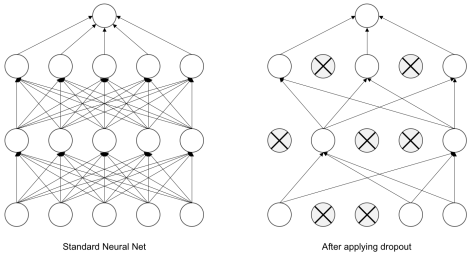


Image: Deep Learning for Computer Vision by Rajalingappaa Shanmugamani

Now a node upstream cannot rely on just one feature downstream. So have to spread out weights or shrink weights (similar affect to L2 regularization).

Can be seen as an extreme form of ensemble methods.

Dropout

- `rate` is the probability that a given unit will be dropped.
- Only applied in the training phase. No dropout at test time.
- A normalization is done to the activations at train time to make the expected value of activations are same to that with no dropout.

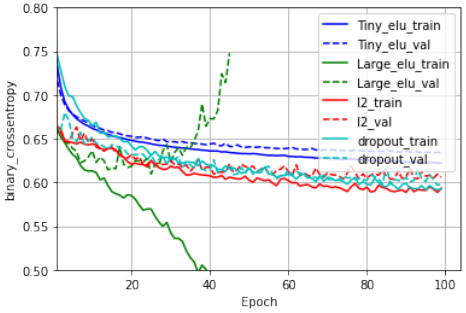
```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import regularizers

l2_model = tf.keras.Sequential([
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l2(0.001),
                 input_shape=(FEATURES,)),
    layers.Dropout(0.5),
    layers.Dense(512, activation='elu',
                 kernel_regularizer=regularizers.l1(0.001)),
    layers.Dropout(0.5),
    layers.Dense(1)
])
```

There are some debates about the dropout effects in convolutional neural network layers.

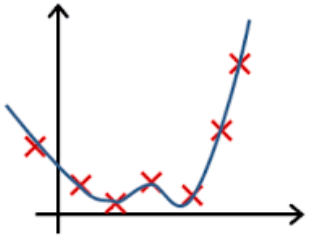
We will test different regularization techniques in lab 4.

Sneak peek at some of the results:



Data Augmentation

Over fitting can be addressed by adding more data.

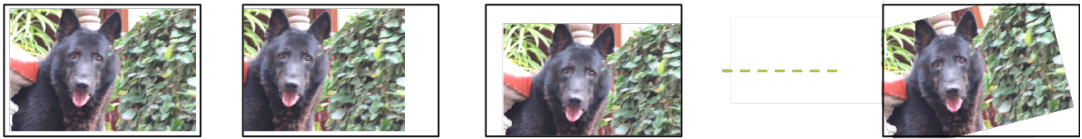


Data Augmentation

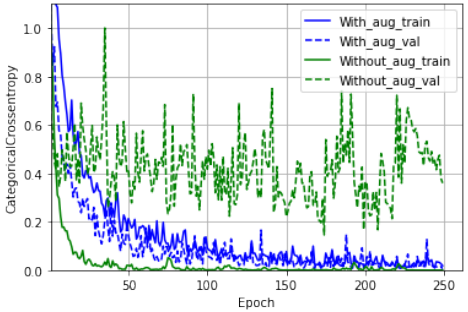
Over fitting can be addressed by adding more data.

Image Augmentation is the process of generating new images for training our deep learning model. These new images are generated using the existing training images and hence we don't have to collect them manually.

Images can be augmented with random image manipulation operations such as translations, rotations, zooming, change brightness, etc. See Keras Documentation for more.



We will cover Augmentation in Lab 5.



Other Techniques for Regularization

- Noise robustness (input and output)
- Early stopping
- Multi-task learning
- ...

We will discuss some of them later.