Generalization

Hamidreza Baradaran Kashani

Deep Neural Networks

Objectives

- Understanding the concept of generalization and overfitting
- Discussing some methods to avoid overfitting and improve generalization such as Early stopping, Regularization, Dropout and Data augmentation.

List of Contents

The Idea of Generalization

- One of the key issues in designing a multilayer network is determining the number of neurons to use.

- The more complexity you have in your model, the greater the possibility for errors.

- If the number of neurons is too large, after training the network will perform well on the training data, but the error on the test data will be too much. This behavior is called **overfitting** which is representative of poor generalization capability of the neural network.

The Idea of Generalization

- A network that generalizes well will perform as well on new data as it does on the training data.

- The complexity of a neural network is determined by the number of free parameters that it has (weights and biases), which in turn is determined by the number of neurons.

- If a network is too complex for a given data set, then it is likely to overfit and to have poor generalization.

The Idea of Generalization

- **Ockham's Razor:** If there are two explanations for a problem such that one of them is simple and the other is complicated, the correct explanation is the simple one.

- In terms of neural networks, the simplest model is the one that contains the smallest number of free parameters (weights and biases), or equivalently the smallest number of neurons.

- Inspired by Ockham's razor principle, to find a network that generalizes well, we need to find the simplest network that fits the data.

The Idea of Generalization

- There are at least **5** different approaches that people have used to produce simple networks: growing, pruning, global searches, parameter magnitude control, and parameter number control.

- **Growing methods** start with no neurons in the network and then add neurons until the performance is adequate.

- **Pruning methods** start with large networks, which likely overfit, and then remove neurons (or weights) one at a time until the performance degrades significantly.

The Idea of Generalization

- **Global searches**, such as evolutionary algorithms, search the space of all possible network architectures to locate the simplest model that explains the data. **NAS (neural architecture search)** is a popular method in this category.

- **Parameter magnitude control** methods like *regularization* and *early stopping*, keep the network small by constraining the magnitude of the network weights.

- **Parameter number control** methods like dropout and dropconnect, keep the network small by constraining the number of the network weights in a probabilistic way.

Overfitting

- We start with a training set of example network inputs and corresponding target outputs:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_q, \mathbf{t}_q\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$\mathbf{p}_q$: input vector $\mathbf{t}_q$: target (ground truth)

- we will assume that the target outputs are generated by:

$$\mathbf{t}_q = \mathrm{g}(\mathbf{p}_q) + \varepsilon_q$$

  - ✓ g: an unknown function
  - ✓ ε_q: a random, independent and zero mean noise source

Overfitting

- Our training objective will be to produce a neural network that approximates g, while ignoring the noise.

- The loss function for neural network training is the sum squared error on the training set:

$$F(\mathbf{x}) = E_D = \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q)$$

$\mathbf{a}_q$: the network output for the input $\mathbf{p}_q$

✓ We use the variable E_D to represent the sum squared error on the training data, because later we will modify the loss function to include an additional term.

Overfitting

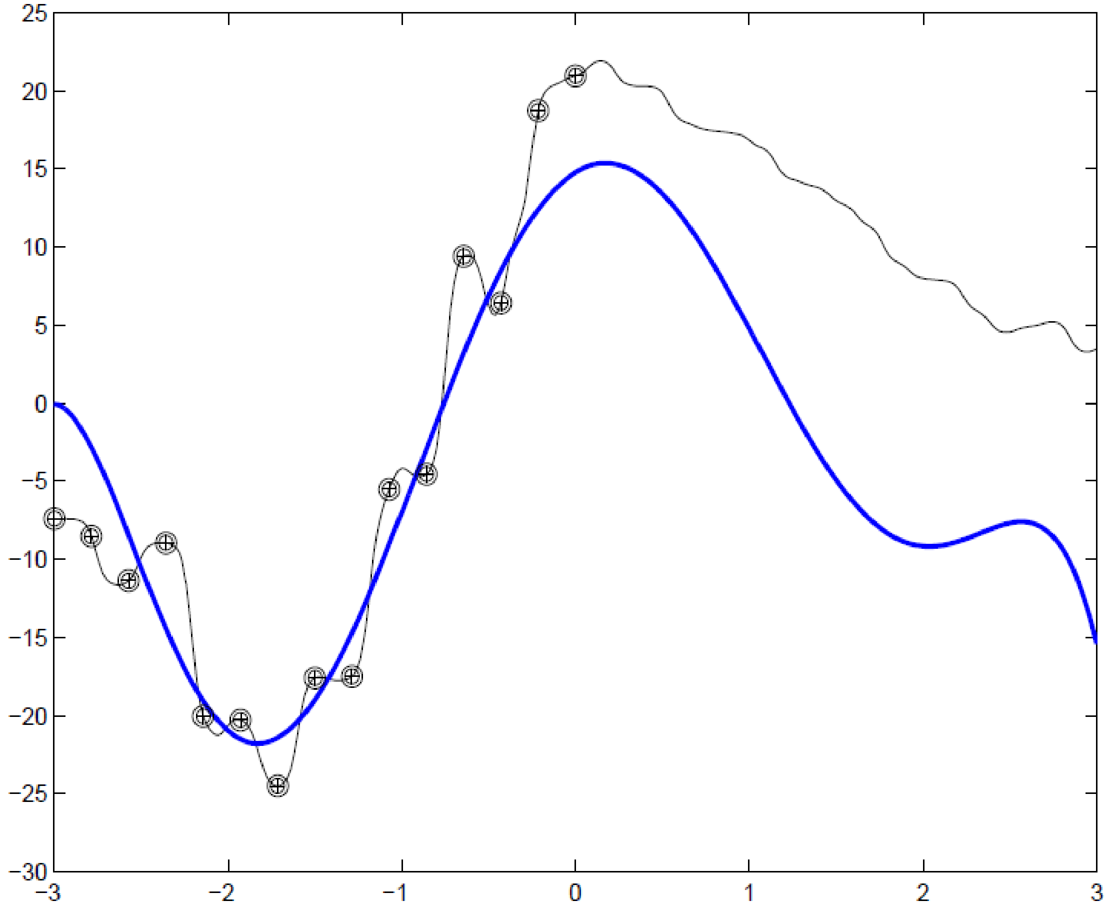
❑ **Case1:** Poor Generalization

✓ The blue curve represents the function g.

✓ The large open circles represent the noisy target points.

✓ The black curve represents the trained network response.

✓ The smaller circles filled with crosses represent the network response at the training points.



Overfitting

- We can see that the network response exactly matches the training points. But it does a very poor job of matching the underlying function. We call this situation, **overfitting**.

- Two kinds of errors occur in Case 1: **Interpolation** and **Extrapolation**

- Interpolation: This type of error occurs for input values between -3 and 0. This is the region where all of the training data points occur. The network response in this region overfits the training data.

  - ✓ Actually interpolation is the situation where the network fails to accurately approximate the function **near** the training points.

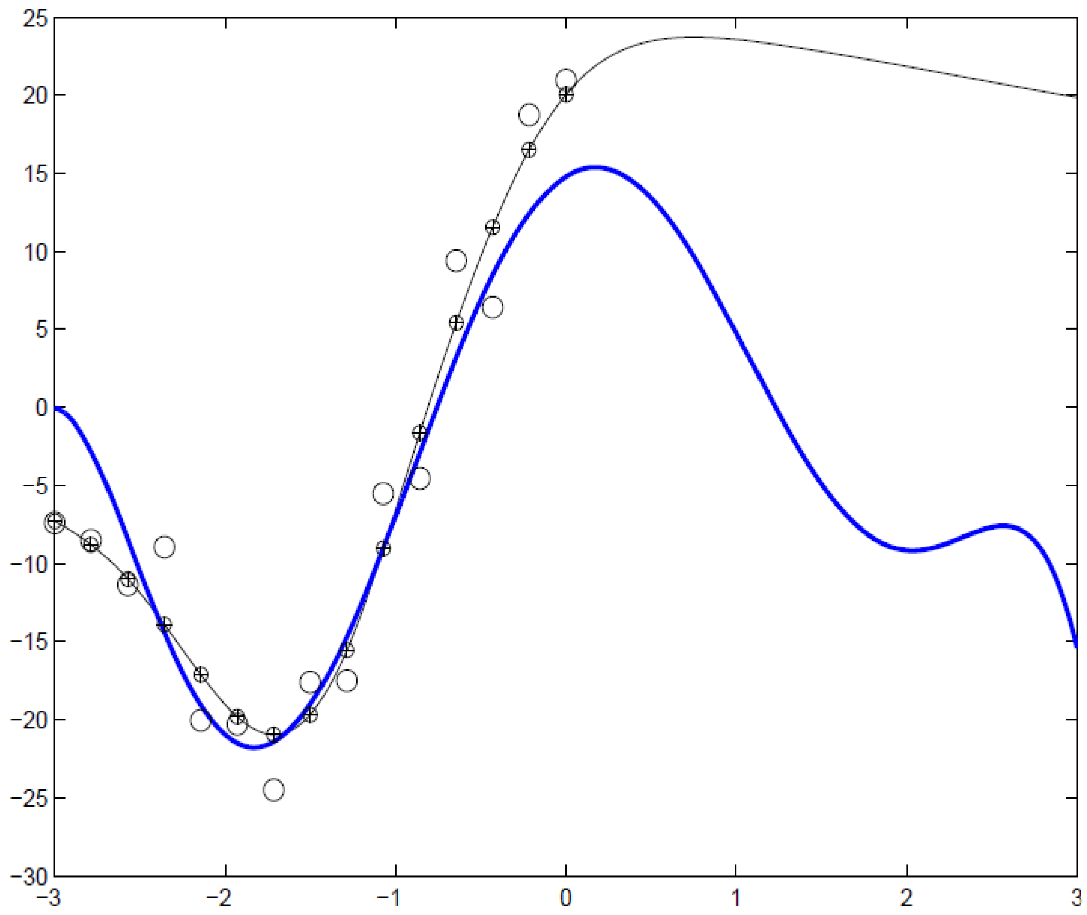
Overfitting

- Extrapolation: The second type of error occurs for inputs in the region between 0 and 3. The network fails to perform well in this region, because there is no training data there.

  ✓ Actually extrapolation is the situation where the network fails to accurately approximate the function **beyond** the range of training data.

Overfitting

- **Case2:** Good Generalization
- ✓ The **blue** curve represents the function g.
- ✓ The large open circles represent the noisy target points.
- ✓ The **black** curve represents the trained network response.
- ✓ The smaller circles filled with crosses represent the network response at the training points.



Overfitting

- In this case, although the network has the same number of weights as the network of case 1 and it was trained using the same data set, but we can see that it generalizes well.

- It has been trained in such a way that it does not fully use all of the weights that are available. It only uses as many weights as necessary to fit the data.

- The network response does not fit the function perfectly, but it does the best job it can, based on limited and noisy data.

Overfitting

- In both case 1 and case 2 we can see that the network fails to extrapolate accurately (although the case 2 has not overfitting).
- The network has been provided with no information about the characteristics of the function out-side of the range -3 to 0, so the response outside this range will be unpredictable.
- There is no way to prevent errors of extrapolation, unless the data that is used to train the network covers all regions of the input space where the network will be used.

Overfitting

- **Generalizability Estimation:** When we are given a limited amount of data, it is important to hold aside a certain subset of this dataset called test set.

- After the network has been trained on training set, we will compute the errors that the trained network makes on this test set.

- The test set errors will then give us an indication of how the network will perform in the future; they are a measure of the generalization capability of the network.

Overfitting

- In order for the test set to be a valid indicator of generalization capability, there are two important conditions required:

  ✓ First, the test set must never be used in any way to train the neural network, or even to select one network from a group of candidate networks. The test set should only be used after all training and selection is complete.

  ✓ Second, the test set must be representative of all situations for which the network will be used. This can sometimes be difficult to guarantee, especially when the input space is high-dimensional or has a complex shape.

Overfitting

- Overfitting is a big issue in neural networks. Therefore we will discuss methods for improving the generalization capability of neural networks.

- There are a number of approaches for improving the generalizability. Most of them try to find the simplest network that will fit the data. We will discuss several methods in this category including Early stopping, Regularization and Dropout.

- Methods like Data augmentation are also used to improve generalizability without changing the network structure. We will discuss it at the end of this chapter.

Early Stopping

- The first method for improving generalization is early stopping. The idea behind this method is that as training progresses, the network uses more and more of its weights, until *all weights are fully used when training reaches a minimum of the loss surface*.

Training iterations ⬆ $\Rightarrow$ **Network complexity** ⬆

- If training is stopped before the minimum is reached, then the network will effectively be using fewer parameters and will be less likely to overfit.

Early Stopping

- In order to use early stopping effectively, we need to know when to stop the training. We will describe a method, called cross-validation, that uses a validation set to decide when to stop. The available data (after removing the test set) is divided into two parts: a training set and a validation set.

  ✓ The training set is used to compute gradients or Jacobians and to determine the weight update at each iteration.

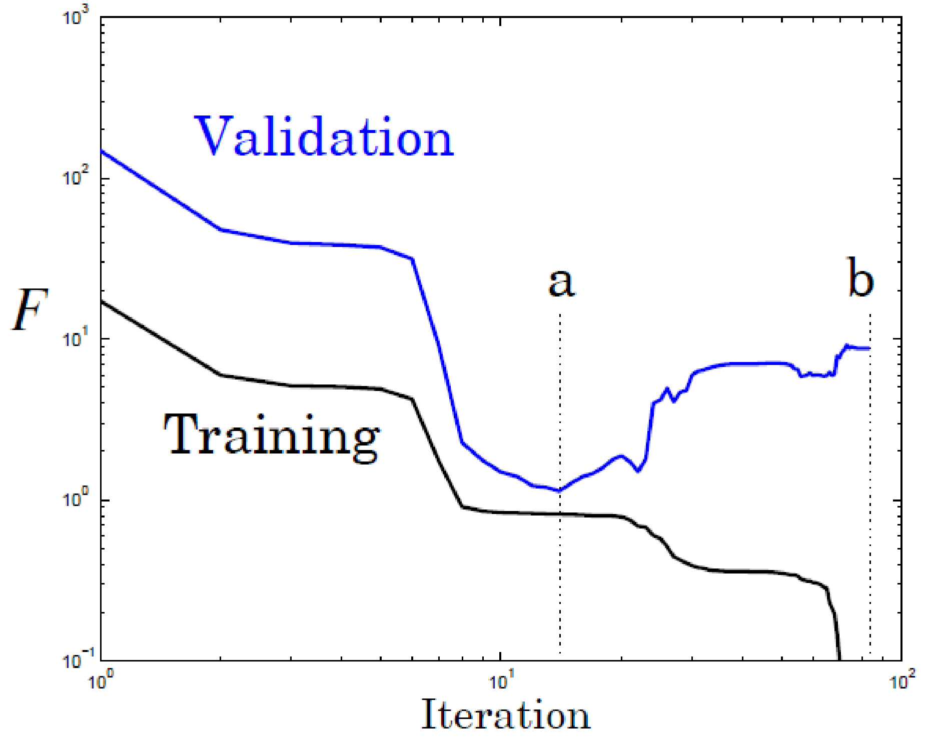
  ✓ The validation set is an indicator of what is happening to the network function “in between” the training points, and its error is monitored during the training process.

Early Stopping

- When the error on the validation set goes up for several iterations, the training is stopped, and the weights that produced the minimum loss on the validation set are used as the final trained network weights.

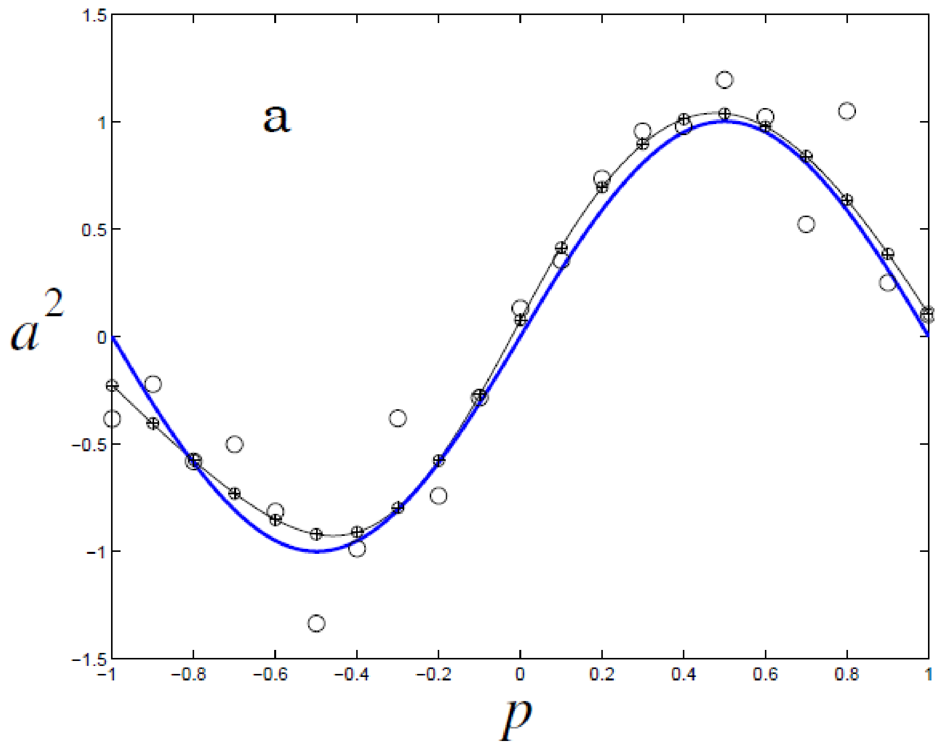
- The figure shows the progress of the training and validation loss (the sum squared error), during training.

- Although the training error continues to go down, minimum of the validation loss occurs at the point labeled "a" corresponds to training iteration 14.



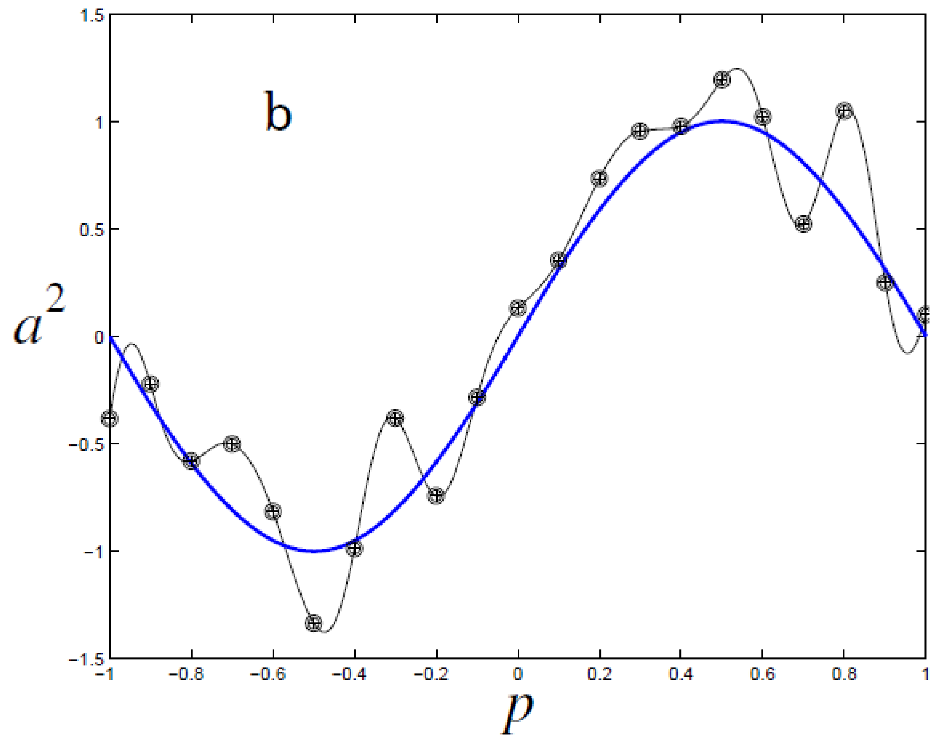
Early Stopping

- The following figure shows the network response at the early stopping point (iteration 14). The resulting network provides a **good** fit to the true function.



Early Stopping

- The following figure demonstrates the network response if we continue to train to point “b” (iteration 82) where the validation loss has increased and the network is **overfitting**.



Early Stopping

- The validation set must be chosen so that it is representative of all situations for which the network will be used. This is also true for the test and training sets, as we mentioned earlier

- When we divide the data, approximately 70% is typically used for training, with 15% for validation and 15% for testing.

- We should use a relatively slow training method. During training, the network will use more and more of the available network parameters.

  ✓ If the training method is too fast, it will likely jump past the point at which the validation loss is minimized.

Regularization

- ❑ The other method for improving generalization is called regularization. For this method, we modify the sum squared error loss function to include a term that penalizes network complexity.

- We add a penalty or regularization term that involved the derivatives of the approximating function (neural network in our case), which forced the resulting function to be smooth.

Regularization

- Under certain conditions, this regularization term can be written as the sum of squares of the network weights, as:

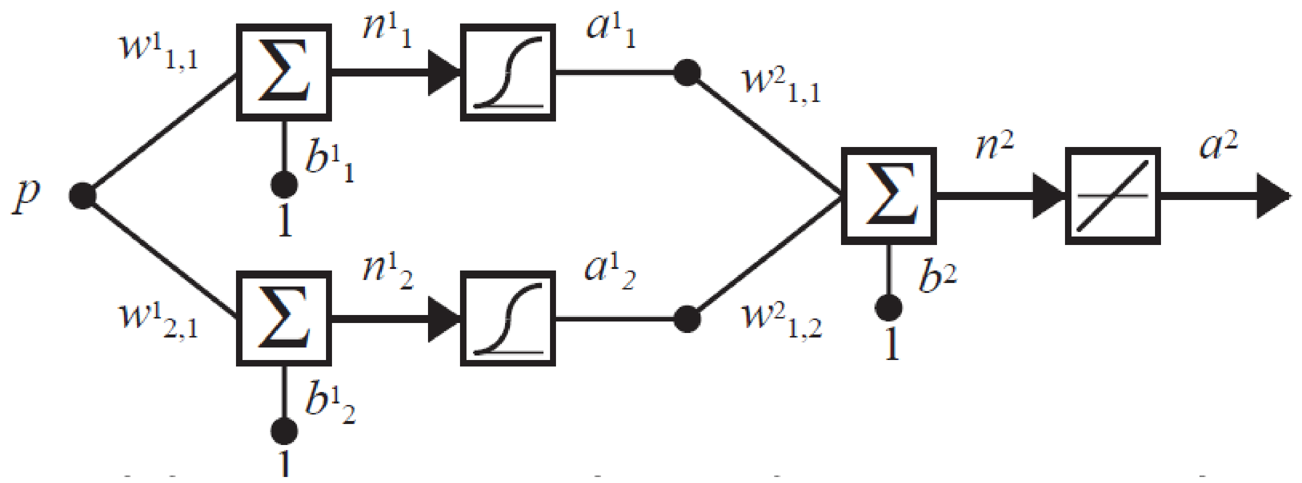
$$F(\mathbf{x}) = \beta E_D + \alpha E_W = \beta \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q) + \alpha \sum_{i=1}^{n} x_i^2$$

- the ratio $\frac{\alpha}{\beta}$ controls the **effective complexity** of the network.

- The larger ratio $\frac{\alpha}{\beta}$ is, the smoother the network response.

Regularization

- Why do we want to **penalize the sum squared weights**, and how is this similar to **reducing the number of neurons**? Consider the example multilayer network shown in the following figure.

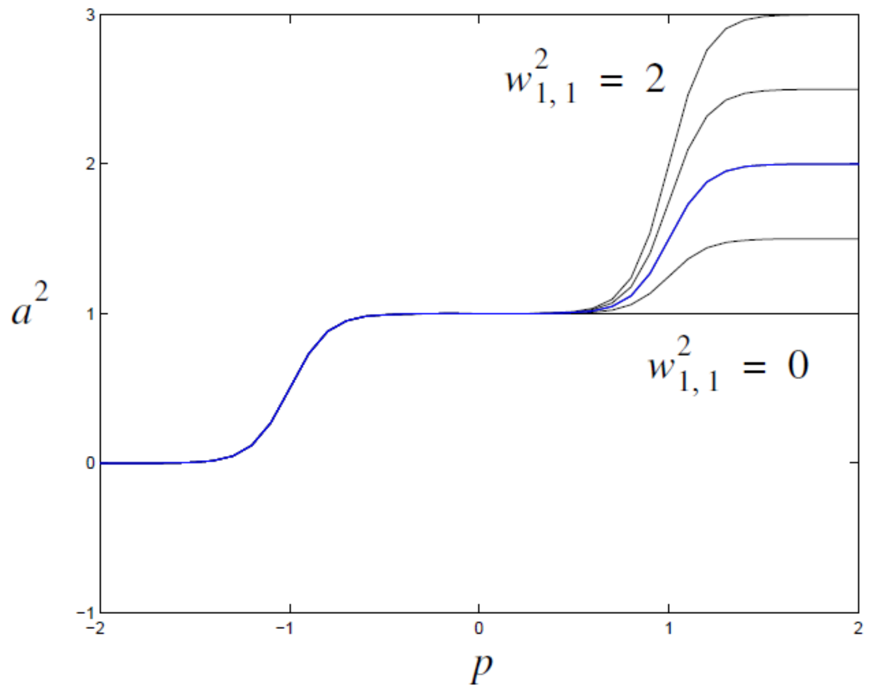


Regularization

- Increasing a weight will increase the slope of the network function. We change a weight from 0 to 2 as shown in this figure.

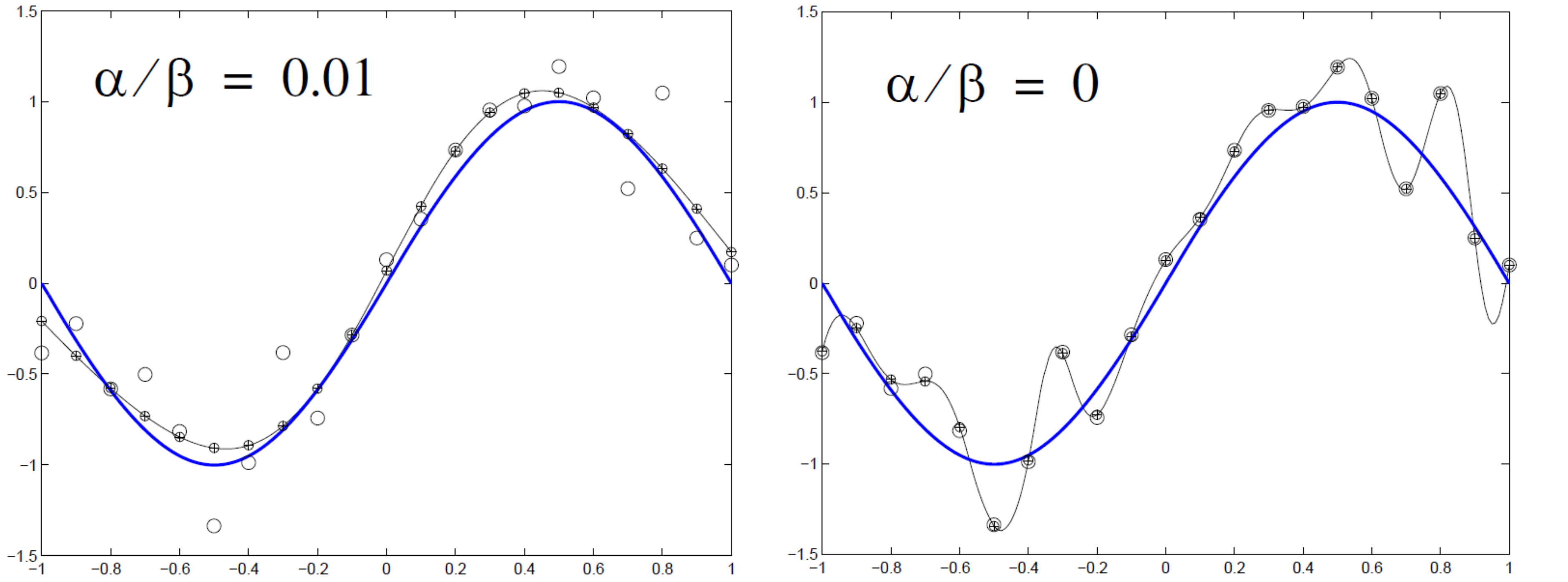
✓ When the weights are large, the function created by the network can have large slopes, and is therefore more likely to overfit the training data.

✓ If we restrict the weights to be small, then the network function will create a smooth interpolation through the training data just as if the network had a small number of neurons.



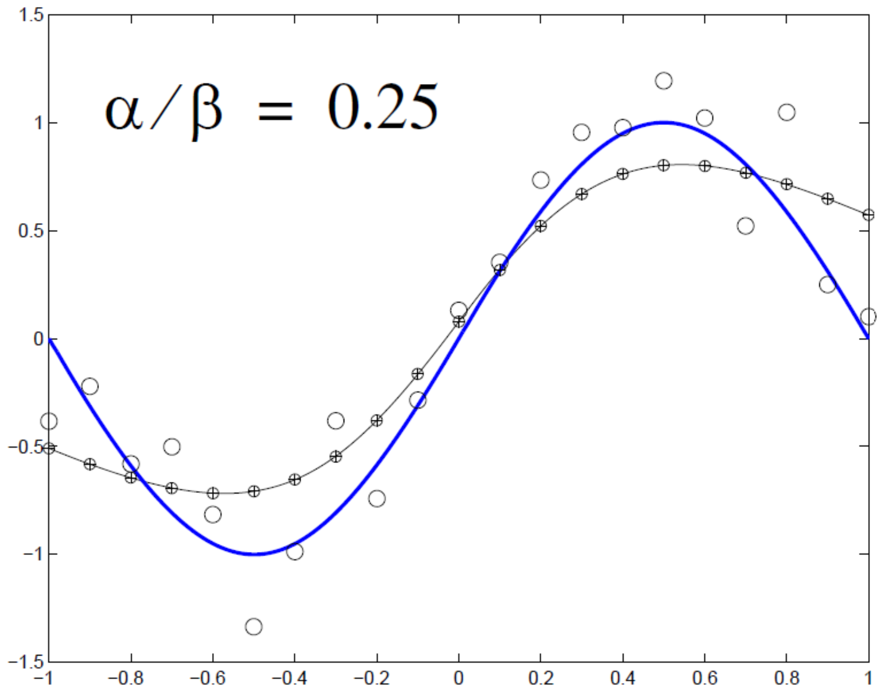
Regularization

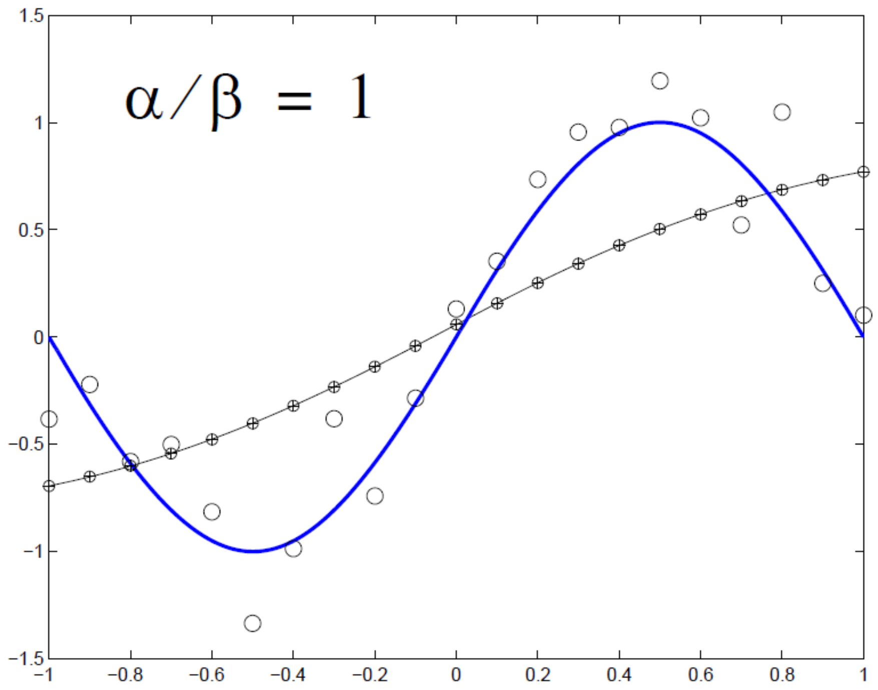
- The key to the success of the regularization method in producing a network that generalizes well, is the correct choice of the regularization ratio $\frac{\alpha}{\beta}$.



Regularization

- The key to the success of the regularization method in producing a network that generalizes well, is the correct choice of the regularization ratio $\frac{\alpha}{\beta}$.





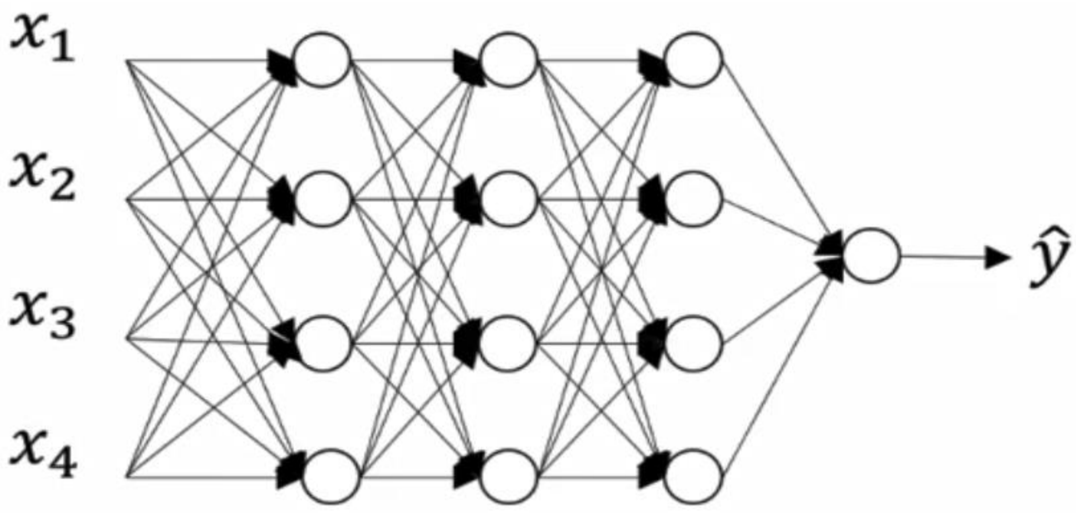
Regularization

- We can see that the ratio $\frac{\alpha}{\beta} = 0.01$ produces the best fit to the true function. For ratios larger than this, the network response is too smooth, and for ratios smaller than this, the network overfits.

- There are several techniques for setting the regularization parameters (α and β). One approach is to use a validation set, such as we described in the section on early stopping; the regularization parameter is set to minimize the validation loss.

Dropout

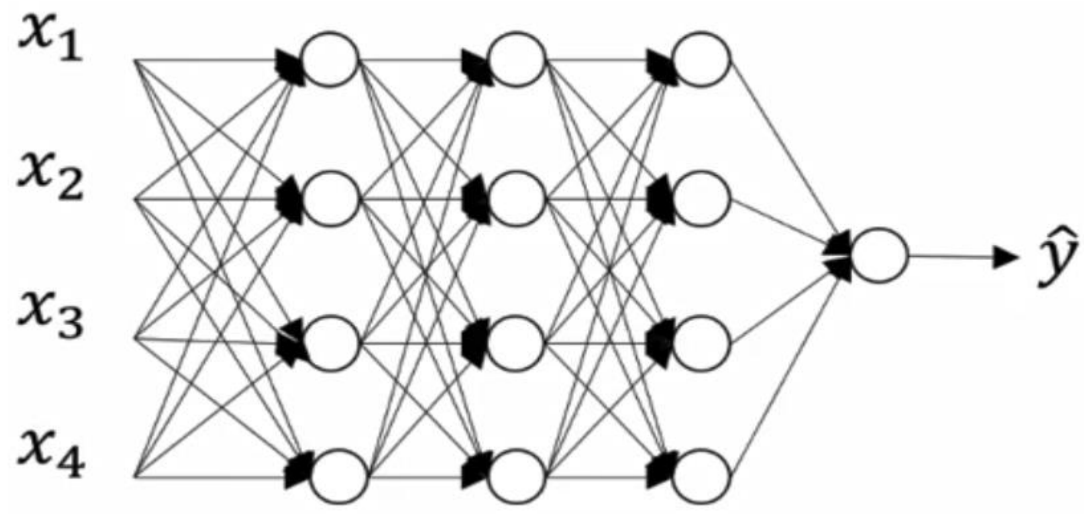
- Next method for improving generalization is called dropout. This method randomly eliminates some neurons in the training process.
- Suppose that we want to train the following network in order to have a good generalization capability.



Dropout

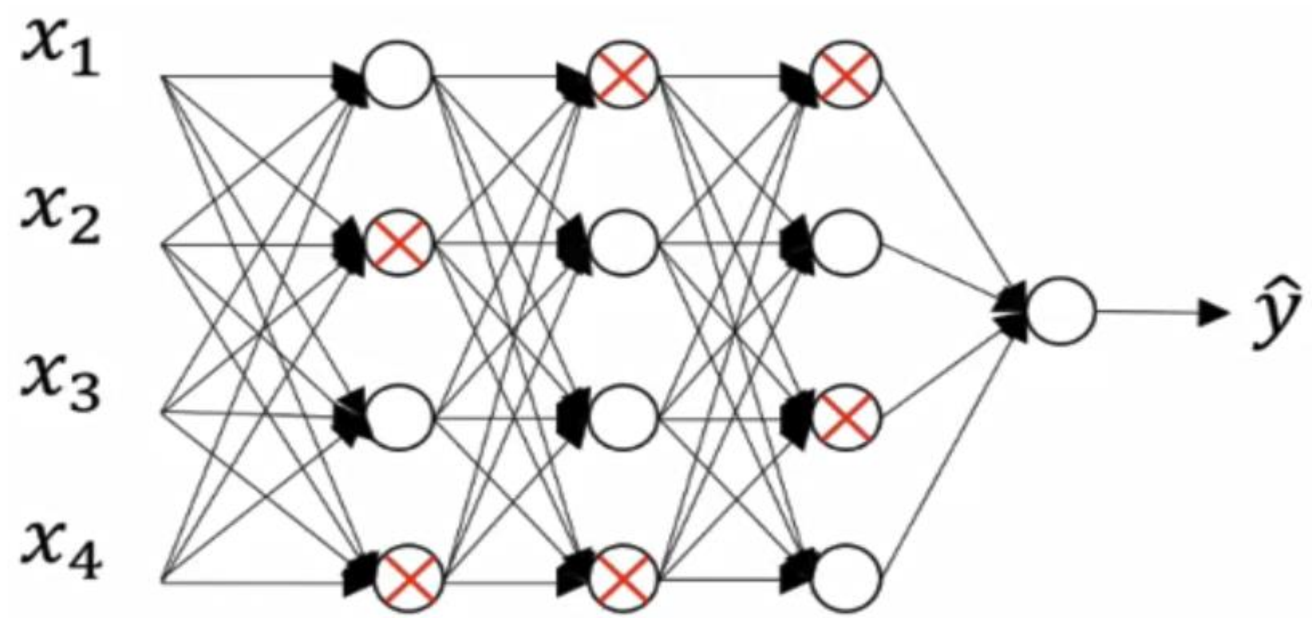
- With dropout, we go through each layer of the network and set some probability of eliminating a neuron in it, called dropout probability.

- For example for each neuron, we toss a coin and have a 0.5 chance of keeping that neuron and a 0.5 chance of removing it.



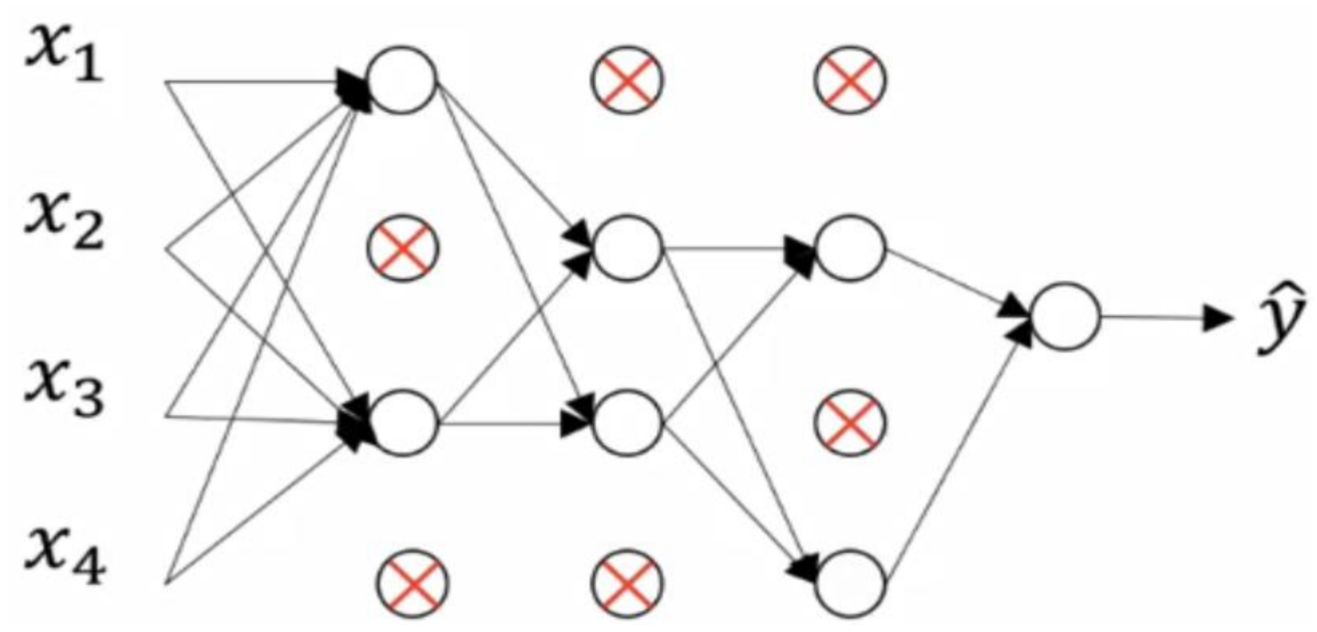
Dropout

- In the following figure the neurons with red signs will be removed for **a batch of training**. Note that for the next batch the signed neurons may differ.



Dropout

- We should remove all the ingoing and outgoing connections from that neuron as well. Then we should train this much diminished network.



Dropout

- Actually in **each batch of training**, we train a different network that is the same as the original network unless some hidden neurons are eliminated.

- Since we train a much smaller network (in comparison with the original network) on each batch of samples, it uses fewer parameters and will be less likely to overfit.

- The other reason that dropout can avoid overfitting is that a neuron cannot rely on one or two inputs only, **it has to spread out its weights and pay attention to all inputs**. As a result, it becomes less sensitive to input changes which results in the model generalizing better.

Dropout

- Spreading out the weights will tend to have an effect of shrinking the weights (like regularization).

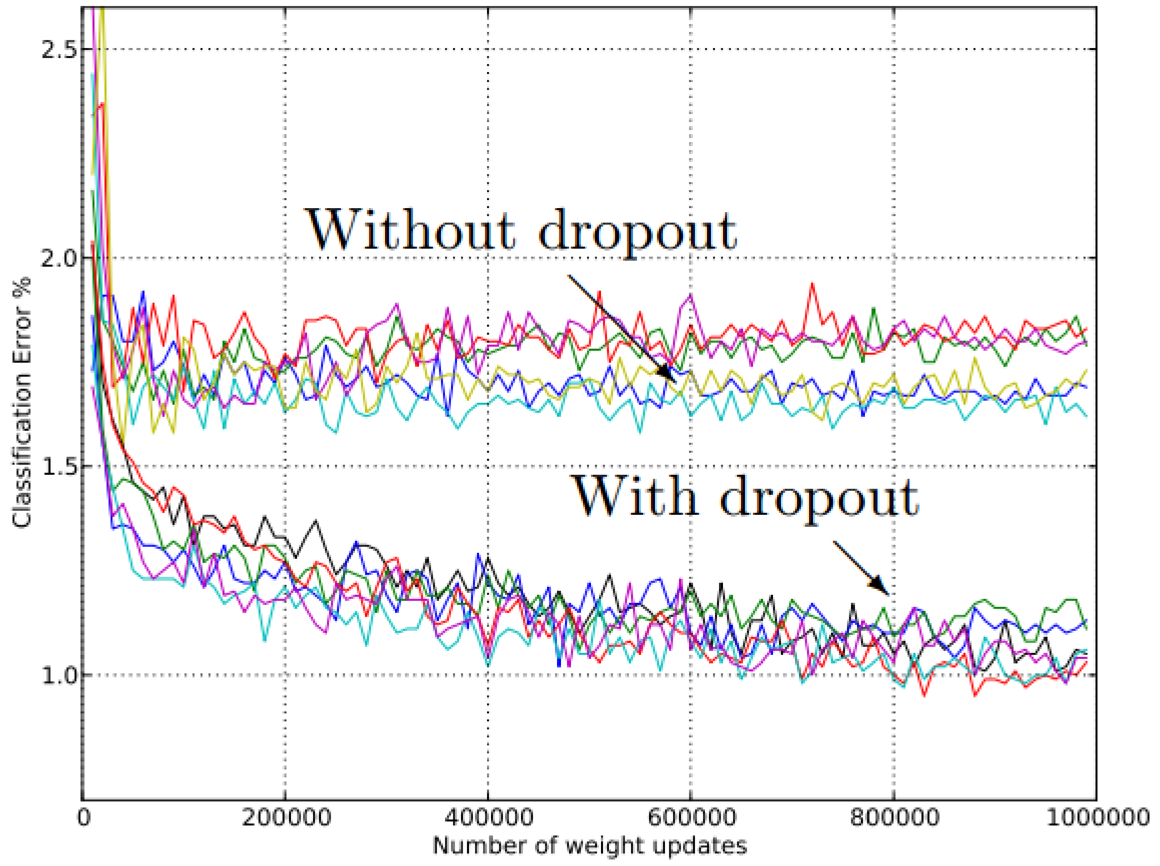
- With dropout, a neuron may be active (not eliminated) in some training iterations but inactive (eliminated) in other iterations based on the dropout probability.

- Each layer of the neural network usually has its own dropout probability.

- Note that the dropout method is only applied on the network during the training phase. at test time, we should use the full version of the original neural network.

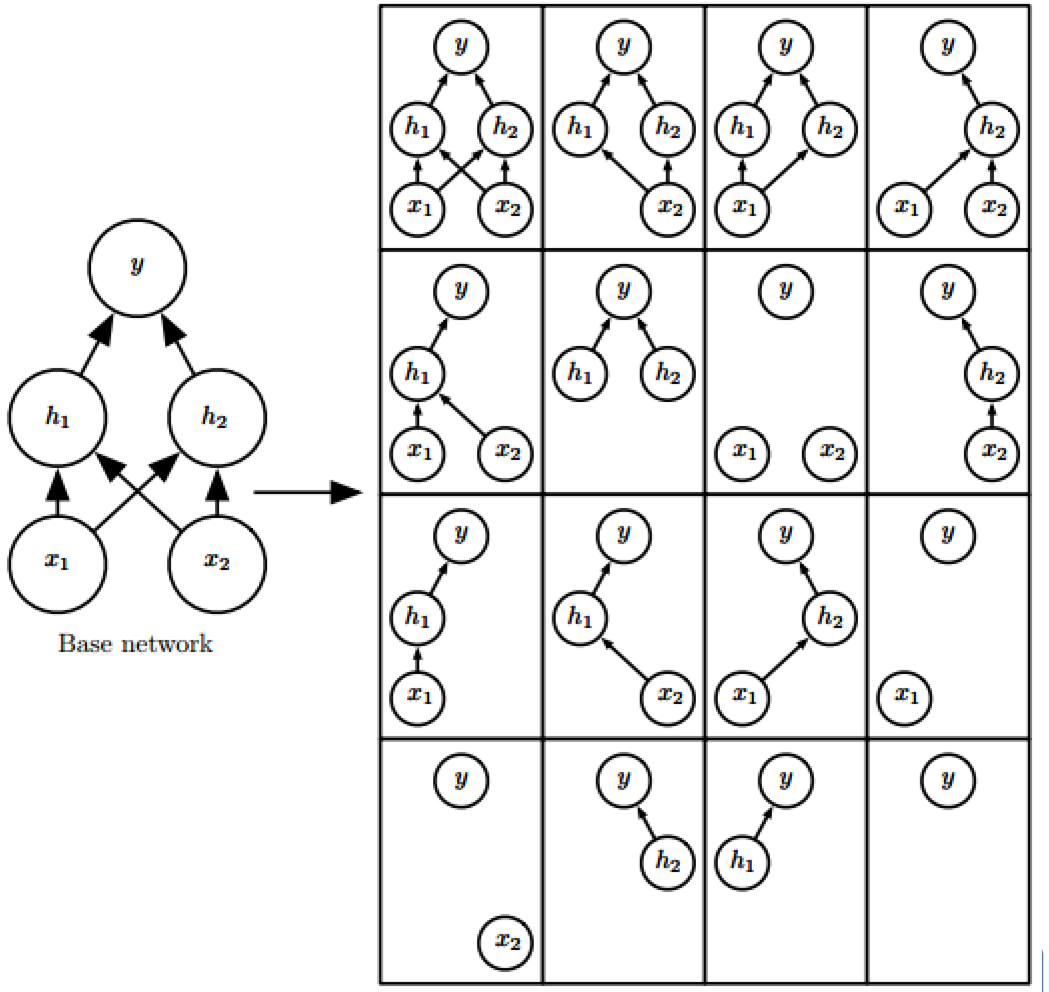
Dropout

Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.



Dropout

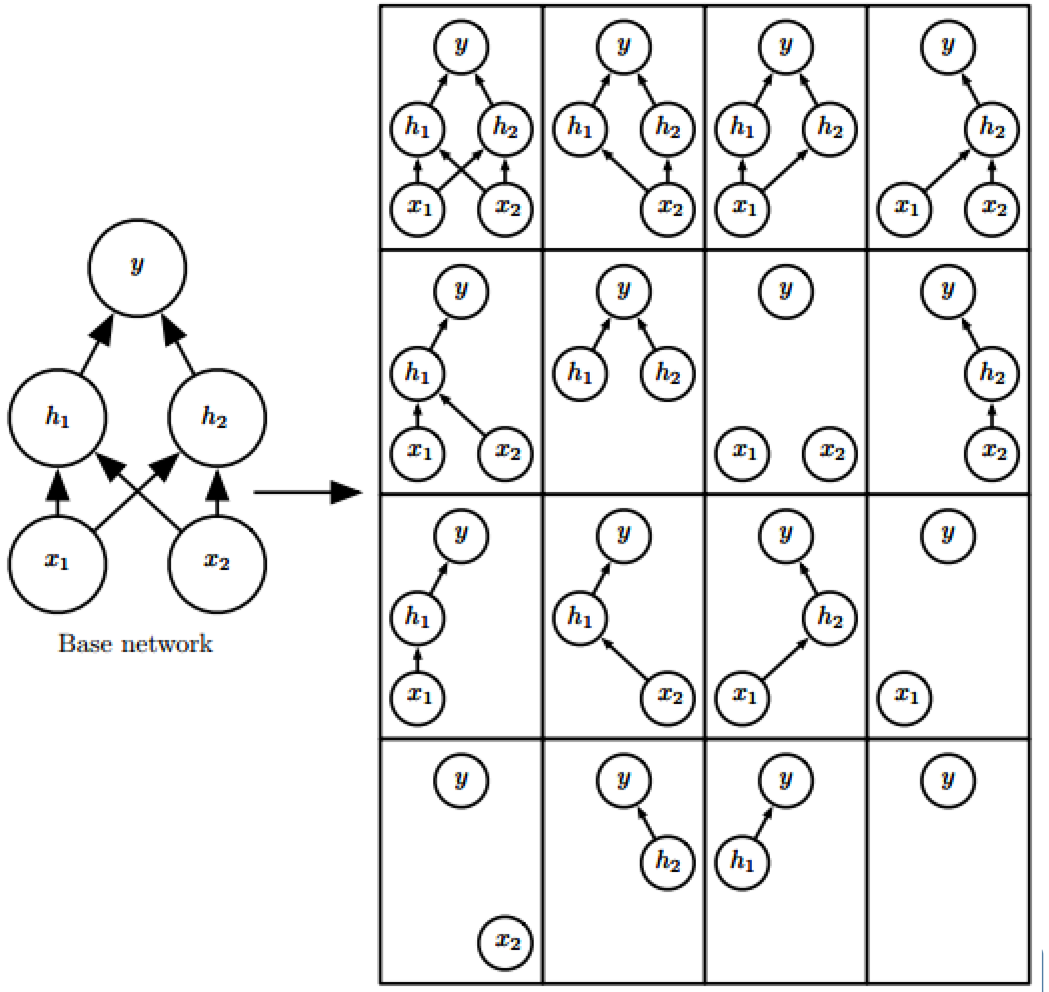
- Dropout trains an ensemble consisting of all subnetworks that can be constructed by removing nonoutput units from an underlying base network.
- We begin with a base network with two visible units and two hidden units.
- There are sixteen possible subsets of these four units. We show all sixteen subnetworks that may be formed by dropping out different subsets of units from the original network.



Dropout

- In this small example, a large proportion of the resulting networks have no input units or no path connecting the input to the output.

- This problem becomes insignificant for networks with wider layers, where the probability of dropping all possible paths from inputs to outputs becomes smaller.



Base network

Ensemble of subnetworks

Data Augmentation

- In all over this chapter, we are assuming that there is a limited amount of data with which to train the network. Limited data is one of the causes of poor generalization capability.

- If the amount of data is unlimited (which in practical terms means that the number of data points is significantly larger than the number of network parameters), then there will not be a problem of overfitting.

- For this reason we try to increase our training set size artificially and build fake examples. This method is called data augmentation.

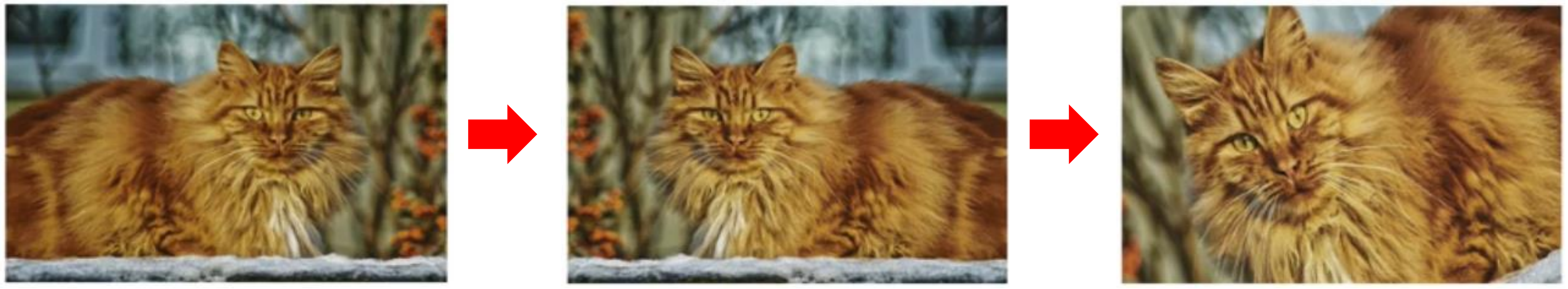
Data Augmentation

- ❑ Suppose we want to try a neural network to classify cats and dogs images:

- If our model overfits and we can solve this problem by giving more training data to the network but getting more real training data is expensive, one thing we can do is augment our training set.

- Because our data are images, we can simply flip horizontally, rotate or scale them and finally add them to the original training set to obtain larger training set.

Data Augmentation



original cat

flipped cat

flipped and scaled cat

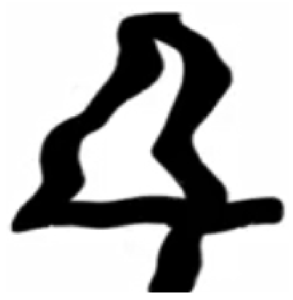
- These extra fake training examples don't add as much information as the original examples, But we can obtain them for free!

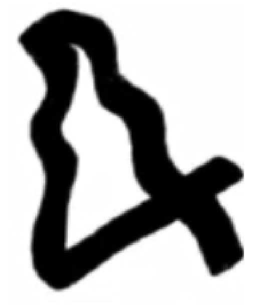
Data Augmentation

- Although we don’t obtain a brand new independent example of a cat, but we say to our model that if something is a cat, then flipping it horizontally is still a cat.

❑ In another example for the OCR task, we can also bring our dataset by taking a digit and imposing random rotations and distortions to it.

original character

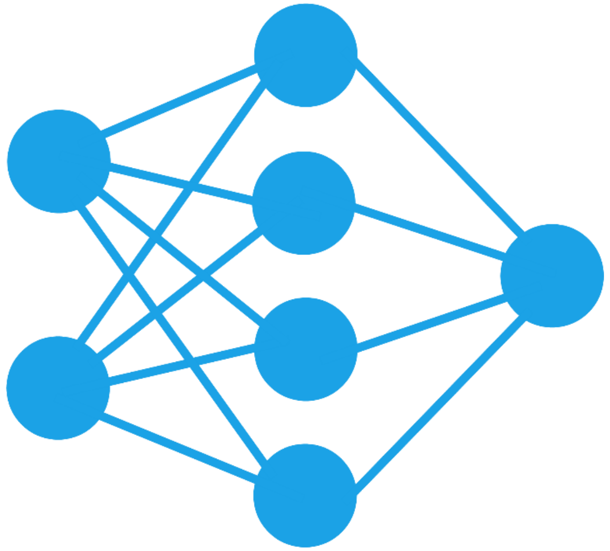

distorted character


rotated & distorted character

Data Augmentation

- Data augmentation can be used as a technique to improve generalization and avoid overfitting, without manipulating the structure of the network.

- This technique is not only used to avoid overfitting, but also for other purposes such as fixing imbalanced classes in classification tasks or applying self-supervised learning in the situations where we suffer from the lack of labeled data.



Thanks for your attention

End of chapter 7

Hamidreza Baradaran Kashani