Neural Networks: Generalization and verification of NN learning process

Andrzej Kordecki

Neural Networks (ML.ANK385 and ML.EM05): Lecture 07
Division of Theory of Machines and Robots
Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology

Table of Contents

Generalization

Empirical Risk

Neural network learning procedure need to assume many values:

- Assume sampled training data with unknown distribution $P(x, y)$,
- Assume the cost function E, and activation functions $f(x, w)$,
- Ideally, the goal is to minimize the expected loss:

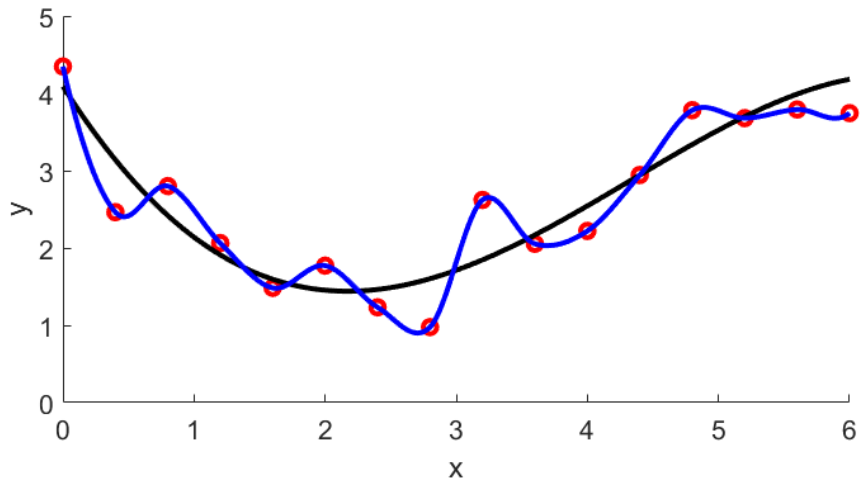
$$R(w) = E(f(w, x), t)$$

- The true distribution is unknown - measurable proxy, which describe empirical loss on the training set E

$$E(w) = \sum_i E(f(w, x_i), t_i)$$

Approximation

The learning process may be viewed as a "curve fitting" problem:



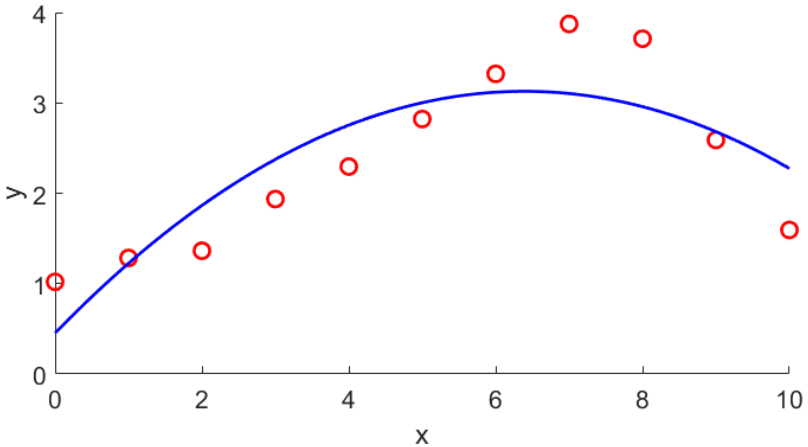
Correct approximation:

- output curve does not pass through all the data points,
- larger error on the training data can lead to better approximation.

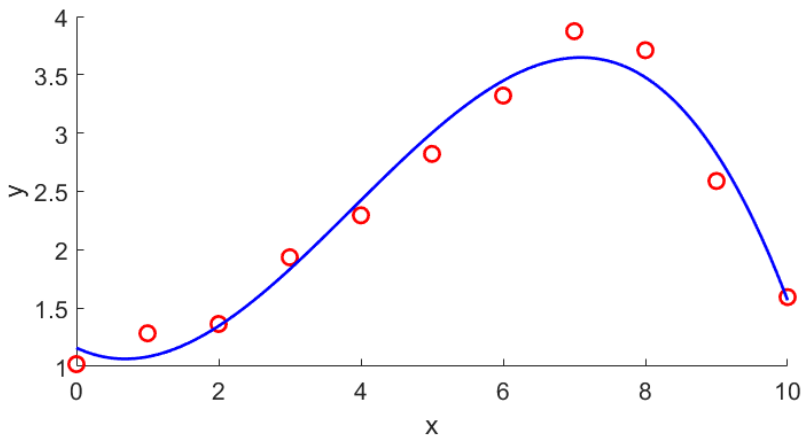
Approximation

The approximation of real 3rd order polynomial model

order $= 2$



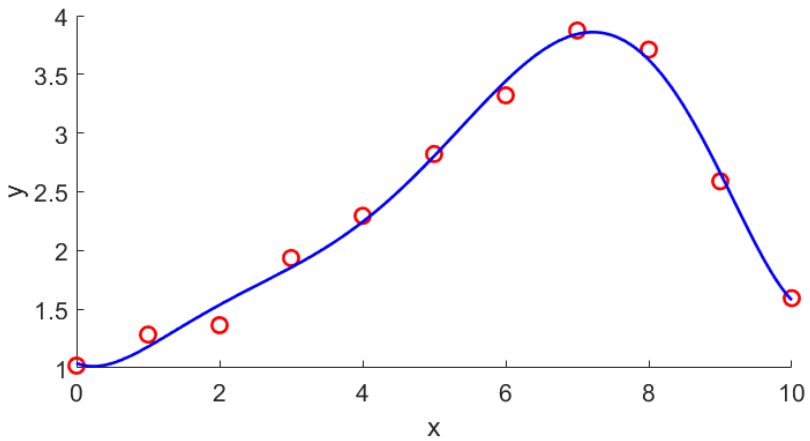
order $= 3$



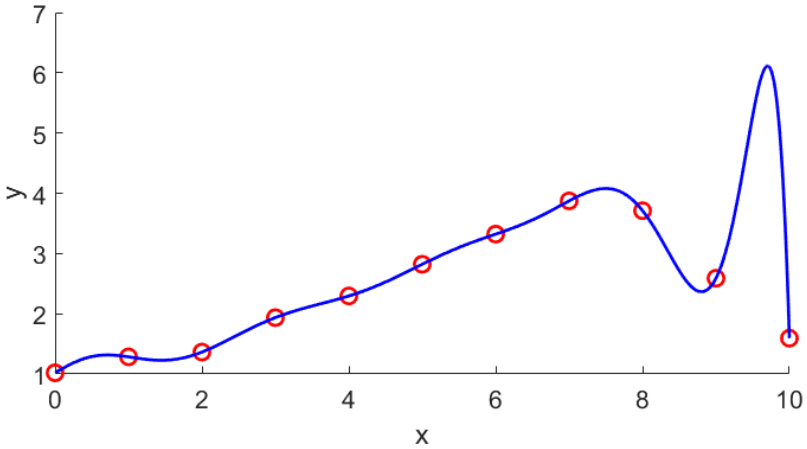
Approximation

Without knowing the model and relying only on data set: Can we decide which model is correct?

order $= 6$

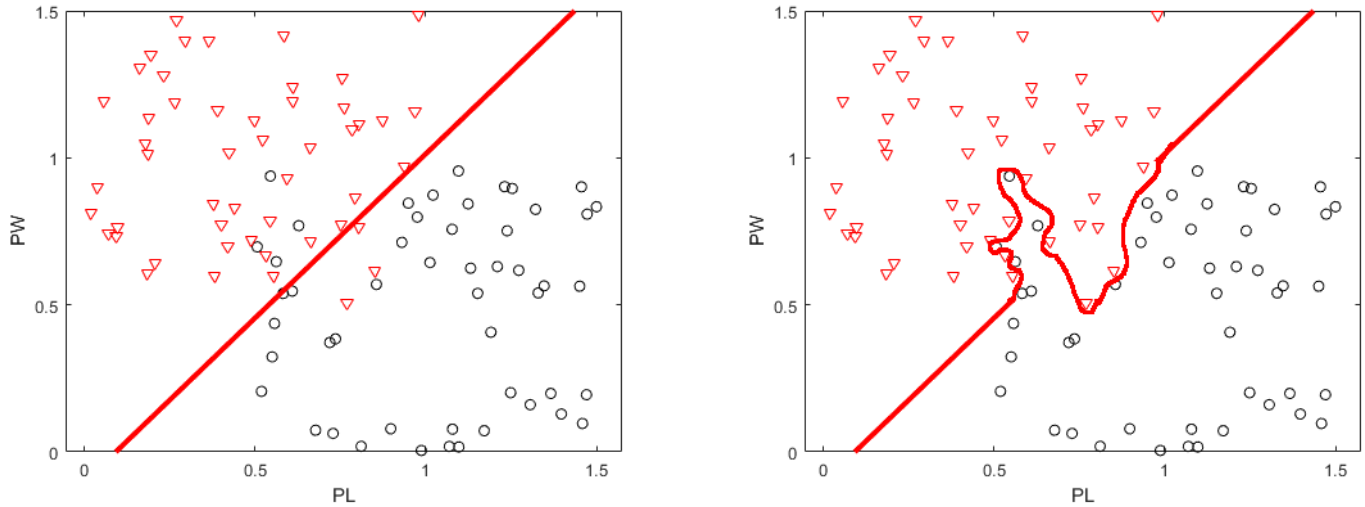


order $= 12$



Classification

Classification decision boundary with use of ANN:



- data without noise - the aim of the network is to classify new inputs as accurately as possible,
- data contains noise - the network should NOT training data to be classified as accurately as possible, because it will likely lead to incorrect results.

Generalization Revision

A network is said to **generalize well** when the input - output mapping computed by the network is correct for test data never used in creating or training the network,

- In back-propagation learning, we typically start to compute the synaptic weights of a multilayer perceptron by loading as many of the training examples as possible into the network. The hope is that the neural network so designed will generalize well.
- Generalization goal is to recover an underlying true function.

Statistical View

Assume training data set D:

$$D = x_{i,p}, t_p, i = 1, ..., n$$

The data set consists of an output t_p for each input pattern $x_{i,p}$.

Generally, the training data generated by some unknown function $f(x)$ will contains random noise ϵ:

$$y_p = f(x_{i,p}) + \epsilon_p$$

We can define a statistical expectation operator P that averages over all possible training patterns:

$$f(x_i) = P(y|x_i)$$

Statistical View

The neural network training problem is to construct an output function $y_{net}(x_i, W, D)$ of the network weights $W = w_{i,j}^{(n)}$, based on the data D, that best approximates model $f(x_i)$ by minimizing the sum-squared cost function:

$$E(w) = \frac{1}{2} \sum_{p} (t_p - y_{net}(x_{i,p}, W, D))^2$$

where error:

$$e = t_p - y_{net}(x_{i,p}, W, D)$$

Statistical View

However to achieve get good generalization:

- good generalisation do not need necessarily to achieve that minimum,
- good generalisation minimize the difference between the network's outputs $y_{net}(x_{i,p}, W, D)$ and the underlying real function $f(x_i) = E_D(y|x_i)$.

The sum squared error function depends on the specific training set D:

$$[E_D(y|x_i) - y_{net}(x_{i,p}, W, D)]^2$$

The network training produce should average results over all possible noisy training sets.

Bias and Variance

If we define the expectation E_D as average value over all possible training sets D:

$$Var(e) = E_D[e^2] - (E_D[e])^2$$
$$E_D[e^2] = (E_D[e])^2 + Var(e)$$
$$E_D[e^2] = Bias^2 + Variance$$

This error function consists of two positive components:

- *bias* - difference between the average network output $E_D[y_{net}]$ and the function $f(x_i)$ - approximation error.
- *variance* - variance of the approximating function y_{net} over all the training sets D - sensitivity of the ANN results on the particular choice of data D.

Bias and Variance

Bias–variance tradeoff:

- $y_{net} = c$ - variance will be zero and bias will be large,
- $y_{net} = f_{complex}(x)$ - variance term will be large and bias will be zero.

If the variance is zero:

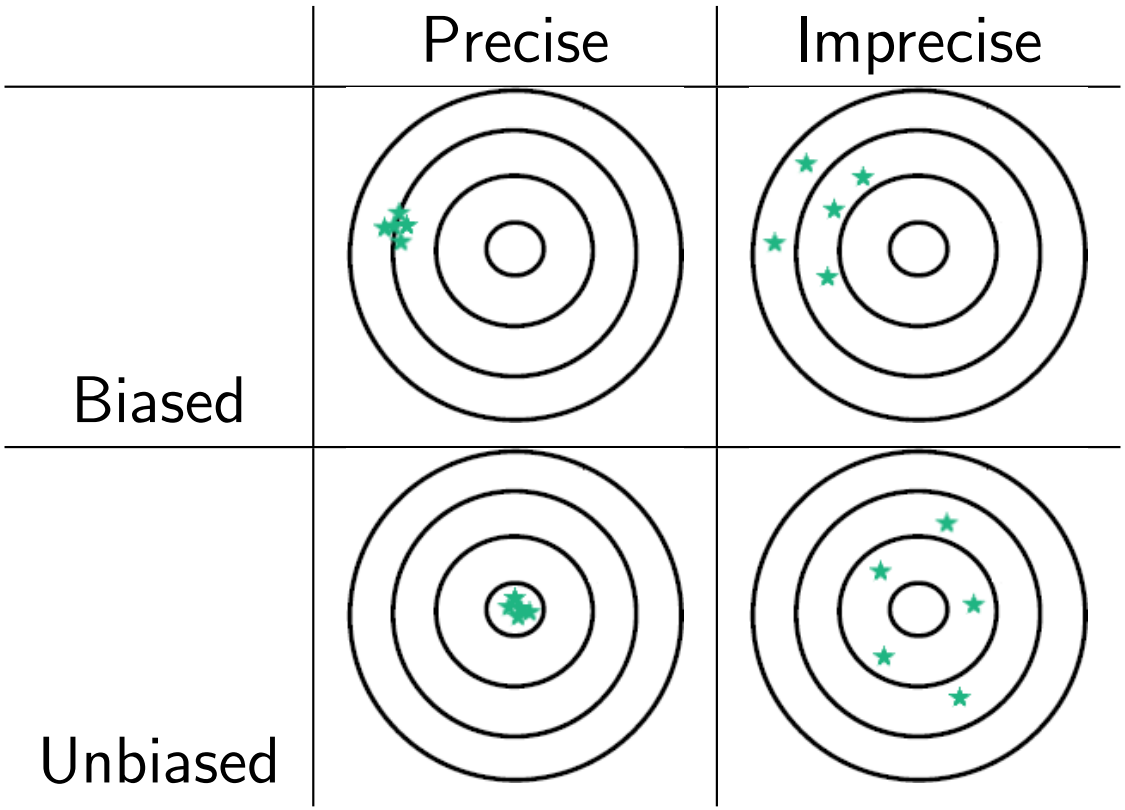
$$E_D[y_{net} - E_D(y_{net})^2] = E_D[f(x_i) + \epsilon - E_D[f(x_i) + \epsilon)^2]] = E_D[(\epsilon)^2]$$

If the bias is zero:

$$E_D[y_{net}] = E_D[t(x_i)] = E_D[f(x_i) + \epsilon] = E_D[y|x_i]$$

The variance of the noise on the data can be substantial.

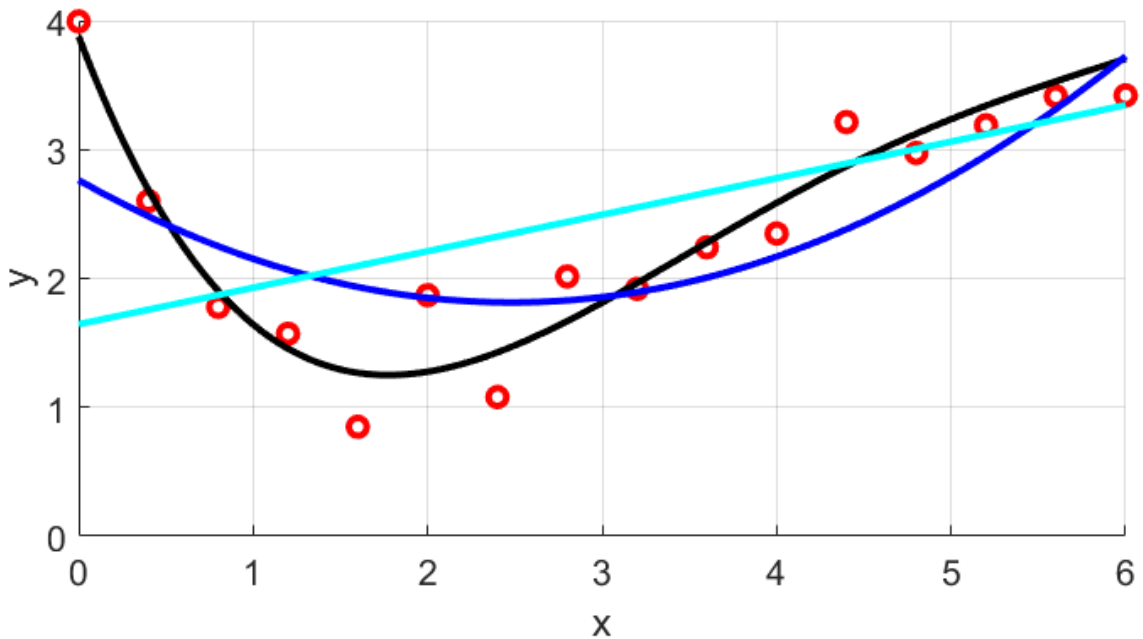
Bias and Variance



Bias and Variance

The function approximation will omit most data points - results:

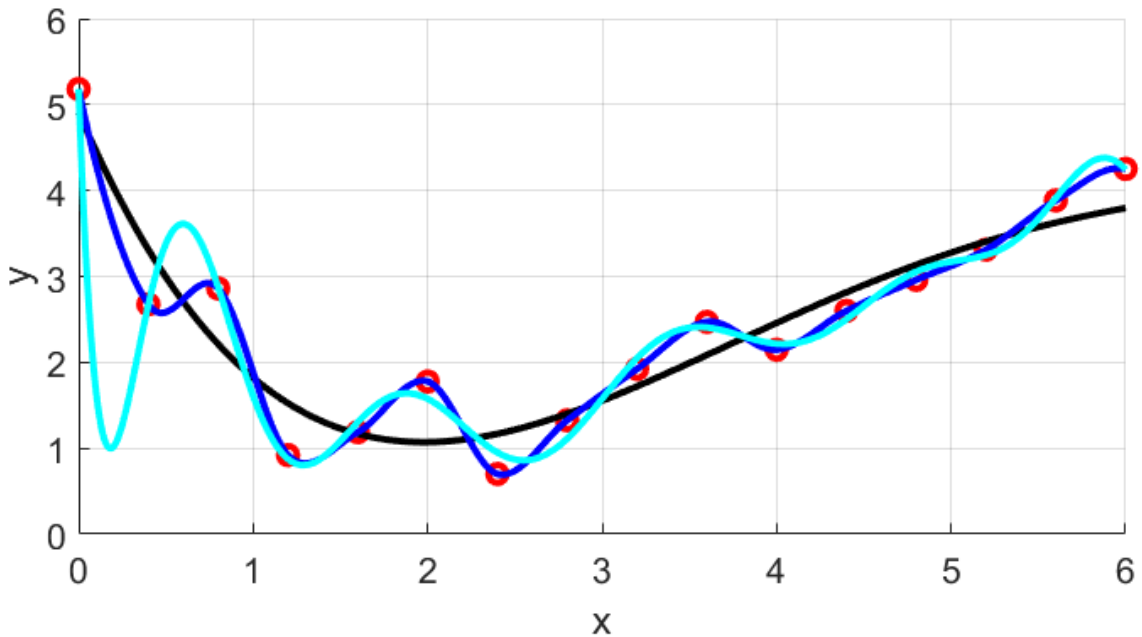
- Big approximation errors (high bias)
- No variation between data sets (low variance)



Bias and Variance

The function approximation will go through all data point - results:

- No approximation errors (low bias)
- Variation between data sets (high variance)



Bias and Variance

In case of other than regression goals we have to check additional measures e.g. confusion matrix.

		Predicted class		
		Dog	Cat	Horse
Actual class	Dog	10	10	2
	Cat	15	16	1
	Horse	0	1	15

Where is the problem of classification?

Over-fitting and Under-fitting

There is clearly a trade-off between minimizing the bias and minimizing the variance:

- Over-fitting (overtraining) - occur to network which is too closely fitted to the training data. Networks tend to have a large variance and hence give a large expected generalization error.
- Under-fitting (undertraining) - network decrease the variance by too much smoothing of the network outputs. Networks tend to have large bias and the expected generalization error is large again.

Over-fitting

To prevent over-fitting we need to make sure that:

- Stop the training early – before it has had time to learn the training data too well.
- Restrict the number of adjustable parameters the network has – e.g. by reducing the number of hidden units, or by forcing connections to share the same weight values.
- Add some form of regularization term to the error function to encourage smoother network mappings.
- Data augmentation methods - data generation methods.

Under-fitting

To prevent under-fitting we need to make sure that:

- The architecture of network has enough hidden units to represent to required mappings.
- We train the network for long enough so that loss function is sufficiently minimized.

The networks to generalize well need to avoid both underfitting and over-fitting of the training data.

Cross Validation

- The cross validation is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. The first step is to partition the available data set is randomly into:
  - training (estimation) subset, used to select the model,
  - validation/test subset, used to validate or test the model.
- The motivation here is to validate the model on a data set different from the one used for parameter estimation.

Holdout Validation

The input data is split into two different datasets labeled as a training and a testing dataset:

- Larger dataset represent training subset, e.g. 60/40 or 70/30 or 80/20 split,
- The training and test dataset is created with equal distribution of different classes of data (stratification).
- Validation dataset can be additionally split into test and validation dataset.

Variants of Cross-Validation

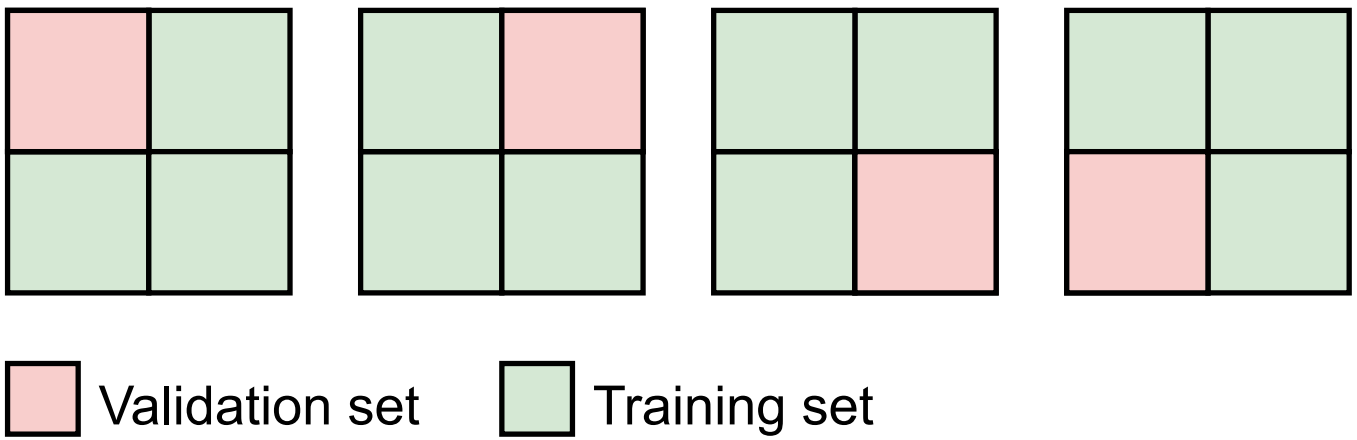
K-fold cross-validation:

- Divide all the training data at random into K distinct subsets, train the network using K–1 subsets, and test the network on the remaining subset.
- The process of training and testing is then repeated for each of the K possible choices of the subset omitted from the training.
- The average performance on the K omitted subsets is then our estimate of the generalization performance.
- If K is made equal to the full sample size, it is called leave-one-out cross validation.

The parameter is usually $K \sim 10$.

Variants of Cross-Validation

K-fold cross-validation for $K = 4$



Cross-Validation

Advantageous of cross-validation:

- A very easy to implement algorithm,
- Provides a great estimate of the true error of a ANN,
- It can indicate problematic examples in a data set,
- Computational cost scales with the number of data subsets, so the cost can be prohibitive,
- We do not obtain one predictor, but many.

Improving Generalization

Generalization

Generalization is influenced by three factors:

- the size of the training sample and how representative the training sample is of the environment of interest (impact of noise)
- the architecture of the neural network,
- the physical complexity of the problem.

We usually have no control over the last factor.

Generalization

We may view the issue of generalization from two different perspectives:

- The architecture of the network is fixed (as a result of the problem), and the issue to be resolved is that of determining the size of the training sample needed for a good generalization to occur.
- The size of the training sample is fixed, and the issue of interest is that of determining the best architecture of network for achieving good generalization

Both of these viewpoints are valid.

Data scaling

- Approach by using direct operation on the testing data to improve generalization is in most cases cheating. But, the data quality have direct impact on network performance.
- Data scaling (data normalization) is a method used to standardize the range of independent variables or features of data. Example: The majority of classifiers based on Euclidean distance between two points. If the first features has a 0-1000 range of values and second 0-1 range, the distance will be governed by first feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Data scaling

Training data scaling

- Normalization, which scales all numeric variables to assumed range (min-max normalization):

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Standardization, which scale to assumed distribution:

$$x_{std} = \frac{x - \bar{x}}{\sigma}$$

where: $\bar{x}$ - mean value of x, σ standard deviation of x.

In case of 24bit image we can use scaling: $x_{norm} = x/255$.

Batch normalization

- Batch Normalization also has a beneficial effect on the gradient flow through the network, by reducing the dependence of gradients on the scale of the parameters or of their initial values.
- Weights problem:
  - The distribution of each layer's inputs changes during training, as the parameters of the previous layers change.
  - The saturation of activation function and the resulting problem of vanishing gradients,
  - The neural network training problems, i.e. a small perturbation in the initial layers, leads to a large change in the later layers.

Ioffe Sergey, Szegedy Christian, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", 2015

Batch normalization

Batch normalization is a part of the model architecture:

- Batch normalization is normalization step that fixes the means and variances of layer inputs.
- Adds two trainable parameters to each layer, so the normalized output is multiplied by a "standard deviation" parameter γ and add a "mean" parameter β.
- The use of γ and β is to keep the mean and variance as 0 and 1 or any fix mean and variance.

Batch normalization

Batch Normalizing Transform, applied to activation x over a mini-batch:

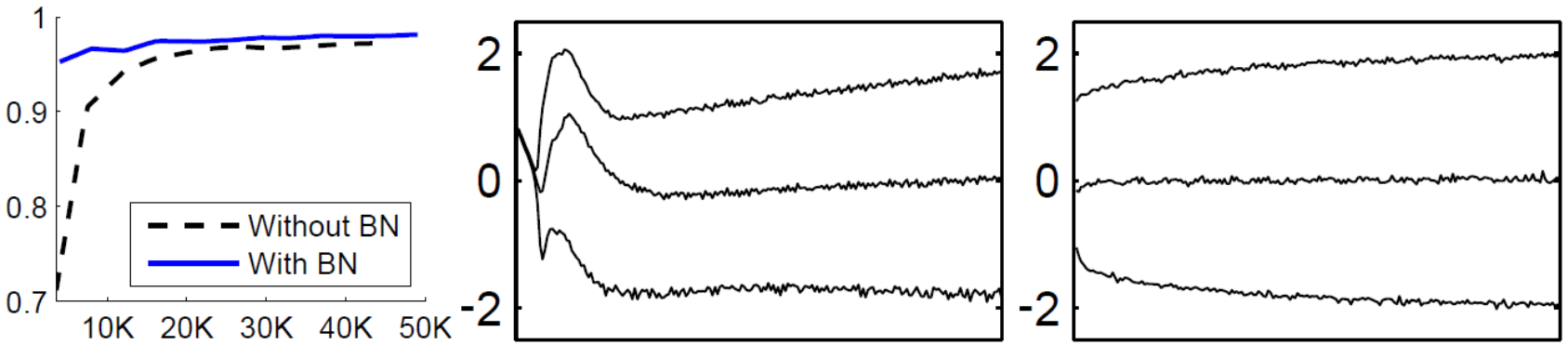
- Input: Values of x over a mini-batch: $B = \{x_{1,...,m}\}$,
- Parameters to be learned: γ and β.
- Linear transformations: $y_i = BN_{\gamma,\beta}(x_i)$

Algorithm ($m > 1$):

1. mean: $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$,
2. variance.: $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$,
3. normalize: $\hat{x}_i = \frac{x_i - \mu_B}{\sigma_B^2 + \epsilon}$,
4. scale and shift: $y_i = \gamma\hat{x}_i + \beta = BN_{\gamma,\beta}(x_i)$

For a layer with d-dimensional input $x = (x^{(1)}, ..., x^{(d)})$, we will normalize each dimension separately.

Batch normalization



(Left): The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. (Middle, Right): The evolution of input distributions to a typical sigmoid, over the course of training, shown as $\{15, 50, 85\}$th percentiles.

Data normalization

Characteristic of input data normalization:

- Allows us to use much higher learning rates and be less careful about weight initialization
- Can prevent the weights from becoming too small - numerical stability.
- Outliers in your data set - data normalization will lead to a very small useful interval,
- Leads to faster convergence. Data after standardization are not bounded, unlike normalization.

Regularization

Best Model:

- model obtained by finding the right number of parameters,
- large model that has been regularized appropriately.

Regularization is a process of introducing additional information or functions in order to solve an ill-posed problem or to prevent overfitting:

- Directly penalize by the number of parameters,
- Restricting the value of the parameters.

Regularization

The general method of regularization adds a penalty term $E_{penalty}$ to the sum squared error E_{SSE} cost function and creates modified loss function - total risk function:

$$R = E_{SSE} + \lambda E_{penalty}$$

where λ the regularization parameter controls the trade-off between reducing the error E_{SSE} and increasing the smoothing $E_{penalty}$. This modifies the gradient descent weight updates so

$$\Delta w_{h,g} = -\eta \frac{\partial E_{SSE}(w_{i,j}^{(n)})}{\partial w_{h,g}^{(m)}} - \eta\lambda \frac{\partial E_{penalty}(w_{i,j}^{(n)})}{\partial w_{h,g}^{(m)}}$$

Regularization

Simple and effective form of complex regularization is called the weight-decay procedure (L2 regularization):

$$E_{penalty} = -\frac{1}{2}\sum_{h,g,n}(w_{h,g}^{(m)})^2$$

In conventional curve fitting this regularizer is known as ridge regression. We can see why it is called weight decay when we observe the extra term in the weight updates:

$$-\eta\lambda\frac{\partial E_w(w_{i,j}^{(n)})}{\partial w_{h,g}^{(m)}} = -\eta\lambda w_{h,g}^{(m)}$$

In each epoch the weights decay in proportion to their size.

Regularization

The impact of λ value on training process:

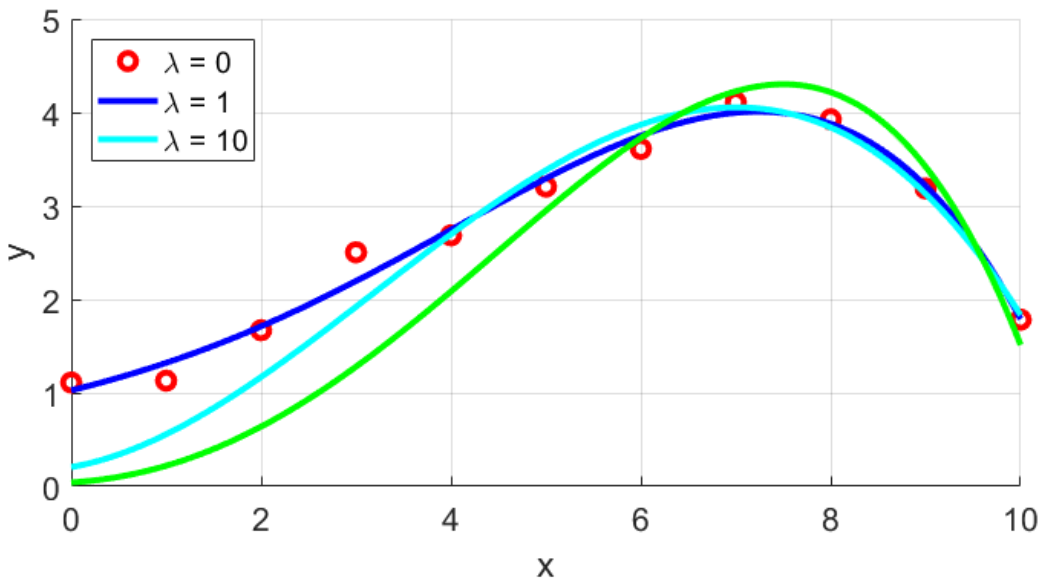
- When is zero - the back-propagation learning process is unconstrained, with the network being completely determined from the training examples.
- When is infinitely large - the implication is that the constraint imposed by the complexity penalty is by itself sufficient to specify the network, which is another way of saying that the training examples are unreliable.

In practical applications, the regularization parameter λ is assigned a value somewhere between these two limiting cases.

Regularization

The impact of λ on regression parameters:

- $\lambda = 0,\ y = 1.89 - 0.57x + 0.29x^2 - 0.02x^3$
- $\lambda = 1,\ y = 0.29 + 0.18x + 0.19x^2 - 0.02x^3$
- $\lambda = 10,\ y = 0.05 + 0.07x + 0.11x^2 + 0.01x^3$



Should all parameters be subject to regularization?

Regularization

The weights of the network are grouped roughly into two categories:

- weights that have a significant influence on the network's performance,
- weights that have practically little or no influence on the network's performance.

The weights in the latter category are referred to as excess weights.

The use of complexity regularization encourages the excess weights to assume values close to zero, while permitting other weights to retain their relatively large values and thereby improve generalization

Weight Restriction

Weight Restriction:

- Restrict the number of hidden units (reduce the number of weights). We can use method of validation to find the best number.
- Many weights in the network can be divided into certain groups with equal value:
  - If there are symmetries in the problem, we can enforce hard weight sharing by building them into the network in advance.
  - In other problems we can use soft weight sharing where sets of weights are encouraged to have similar values by the learning algorithm.

Bagging method

Bagging (short for bootstrap aggregating) is a technique for reducing generalization error by combining several models:

- The idea is to train several different models separately (changes in training data or structure), then have all of the models vote on the output for test examples.
- This is an example of a general strategy in machine learning called model averaging.

The term "dropout" refers to dropping out units (both hidden and visible) in a neural network during training. This prevents units from co-adapting too much.

Dropout method

Dropout method is an efficient way to average many large neural nets:

- Each time we present a training example, we randomly omit each hidden unit with probability p. Dropout samples from an exponential number of different "thinned" 2^H possible architectures (H - number of units).
- In most cases, we can effectively remove a unit from a network by multiplying its output value by zero (equivalent of appropriate weight $= 0$).

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014.

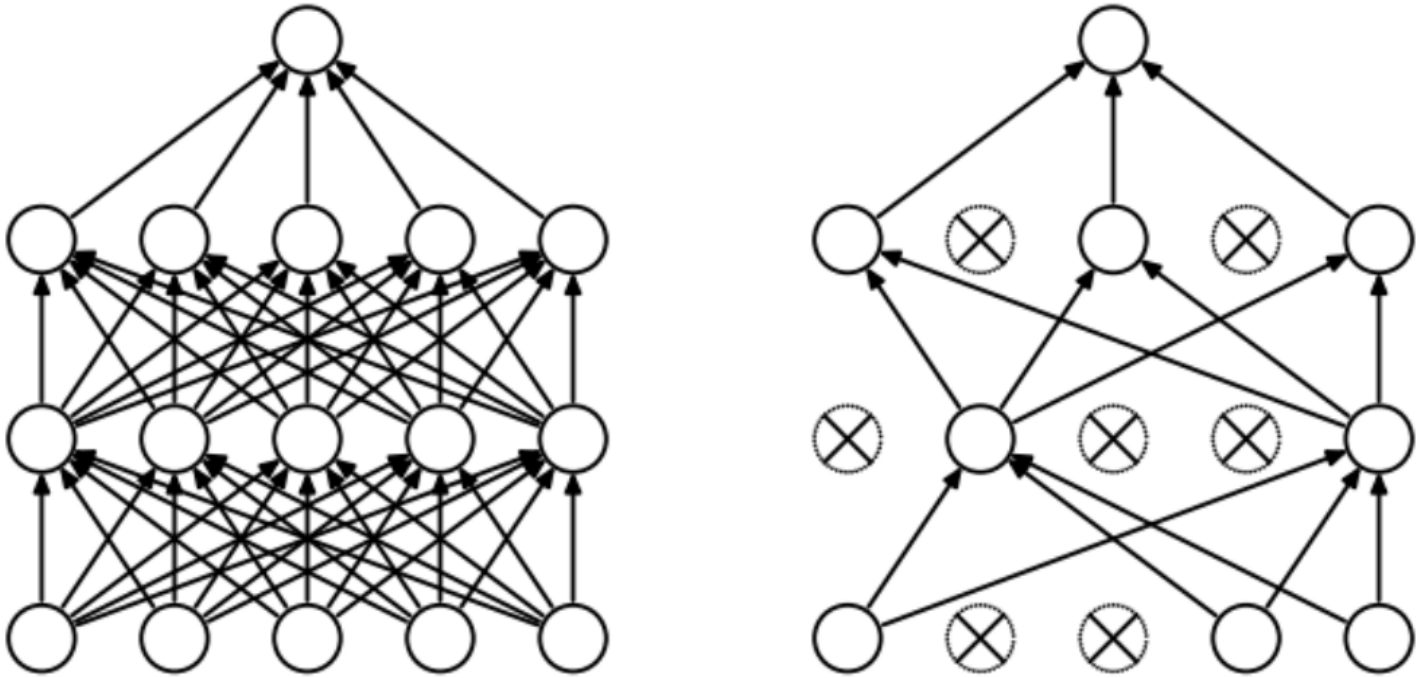
Dropout method

Dropout method is an efficient way to average many large neural nets:

- At test time, it is not feasible to explicitly average the predictions from exponentially many thinned models.
- The models share parameters with other NN, with each model inheriting a different subset of parameters from the parent neural network.
- It is easy to approximate the effect of averaging the predictions (using the geometric mean as average) of all these thinned networks by simply using a single unthinned network that has smaller weights.

Dropout method

An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

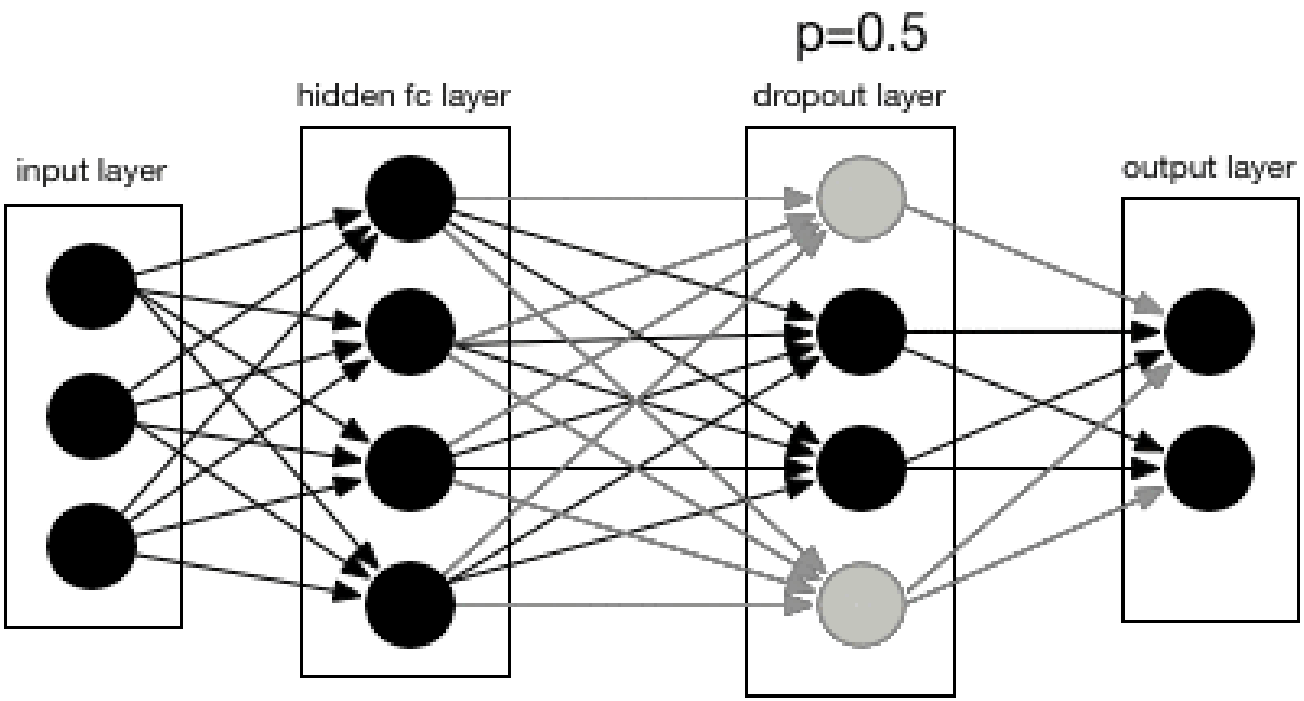


(a) Standard Neural Net

(b) After applying dropout.

Dropout implementation

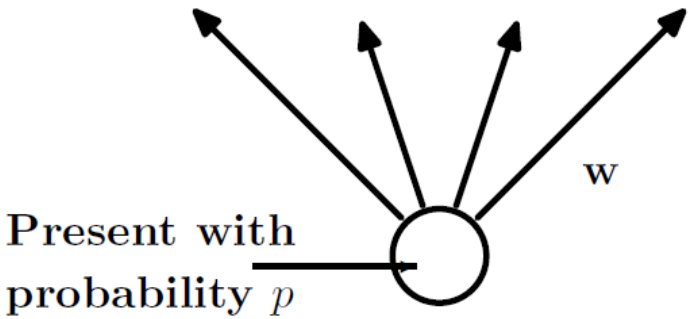
Dropout layer randomly "removes" neurons along with all of their connections.



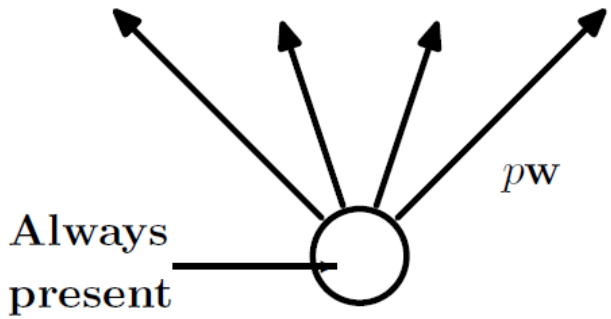
Dropout method

Probability p

- hidden layer: fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5.
- input layer: optimal probability is usually closer to 1 than to 0.5



(a) At training time



(b) At test time

Dropout method

The result of dropout method is final single neural net:

- The weights of this network are scaled-down versions of the trained weights.
- If a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p at test time.
- By doing this scaling, 2^n networks with shared weights can be combined into a single neural network
- The final weight is calculated as weighted arithmetic mean over all probabilities and weights in training time.

Dropout method

Observations:

- Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
- Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.
- In testing phase, the entire network is considered and each activation is reduced by a factor p.
- In some architectures (e.g. CNN) using dropout is tricky.

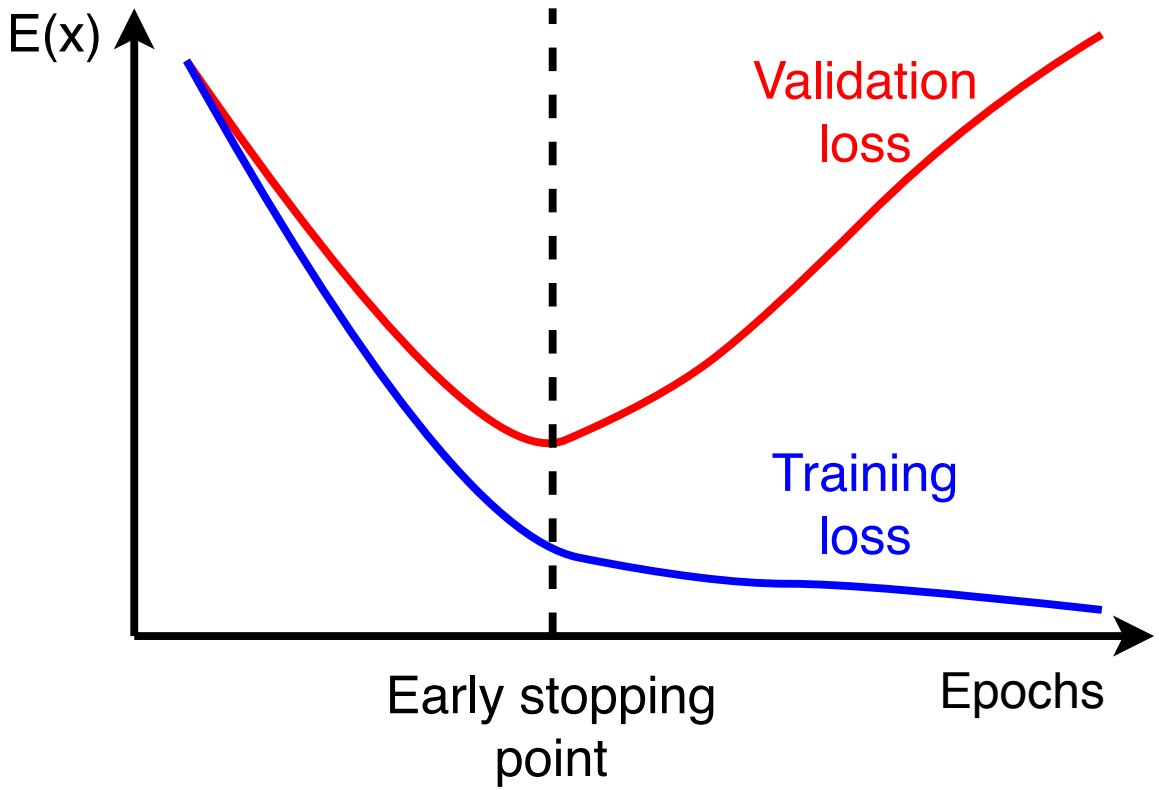
Early-Stopping Method of Training

In case of iterative gradient descent based network training, the error value will naturally decrease with increasing numbers of epochs. Therefore, we may identify over-fitting through the use of cross-validation:

- Train network until the error on the validation set starts rising again, and then stops. That is the point at which we expect the generalization error to start rising as well.
- The problem with of stopping early is that the validation error may go up and down numerous times during training. The safest approach is generally to train to convergence, saving the weights at each epoch, and then go back to weights at the epoch with the lowest validation error.

Early-Stopping Method of Training

In reality, the validation-sample error does not evolve over the number of epochs used for training as smoothly as the idealized curve shown.



Questions