Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Mesterséges Intelligencia és Rendszertervezés Tanszék

Neural networks - regularization

Lecturer: Dr. Gábor Hullám

Questions of teaching an artificial neural network

- How big (how many layers, how many processing elements per layer) network should we choose?
- How to choose the value of learning rate, α?
- What initial weight values should we set?
- How to choose a training and testing sample set?
- How to use the teaching points, how often should we change the weights of the network?

How big a network should we choose?

- How big a network to choose? (how many layers, how many processing elements per layer)

- There is no exact calculation to determine this.

1.) Starting from a 'larger' network and then reducing network redundancy to a minimum.

- Pruning techniques
- Regularization

$$C_r(\mathbf{w}) = C(\mathbf{w}) + \lambda \sum_{i,j} |w_{ij}|$$

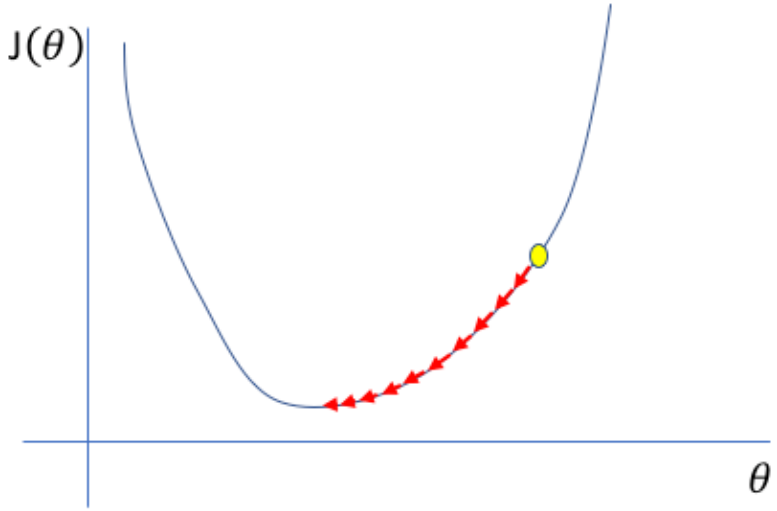
How big a network to choose?

- How big a network to choose? (how many layers, how many processing elements per layer)

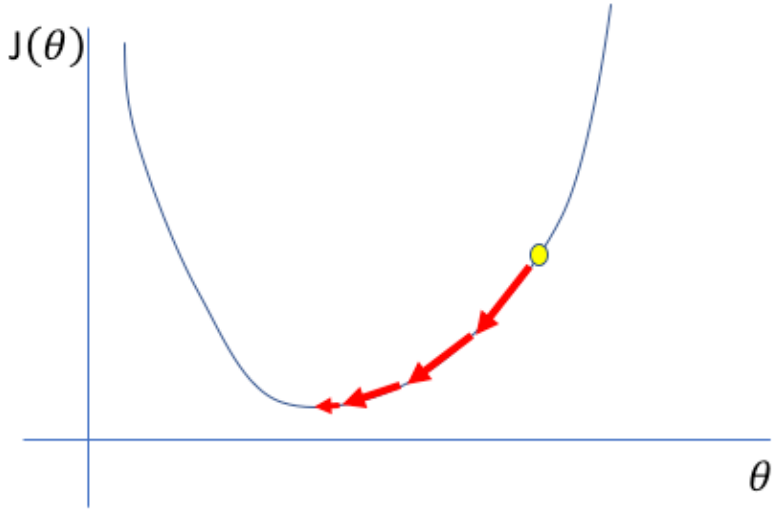
2.) Starting from a 'smaller' network, then gradually expanding (with processing elements or layers)
We examine whether it can solve the task.

The two approaches do not necessarily lead to the same result!

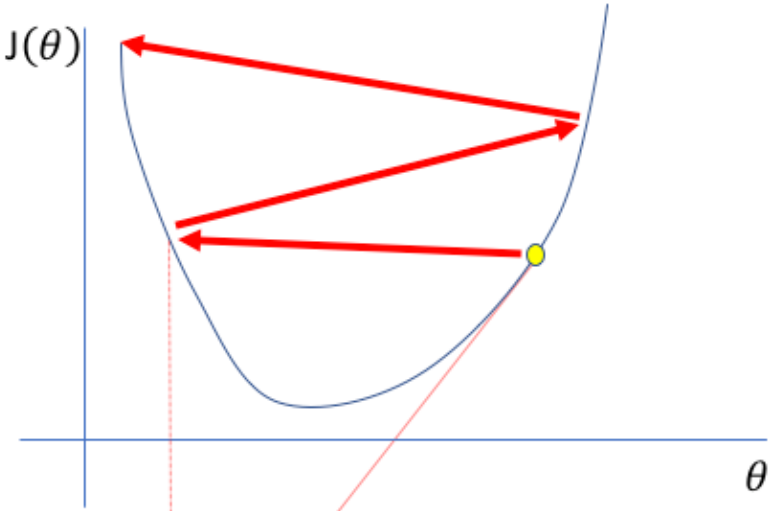
How to choose the value of the learning rate?



A small learning rate requires many updates before reaching the minimum point

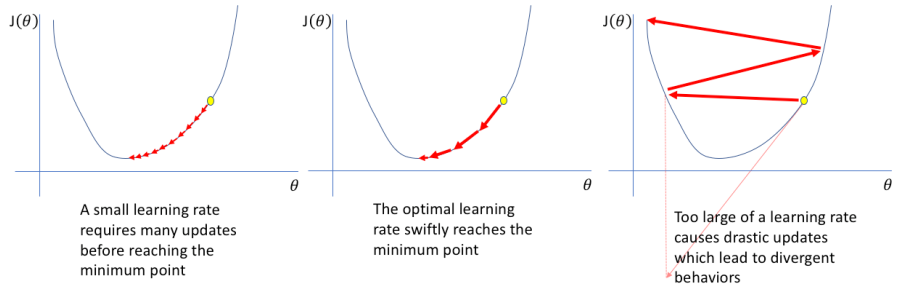


The optimal learning rate swiftly reaches the minimum point



Too large of a learning rate causes drastic updates which lead to divergent behaviors

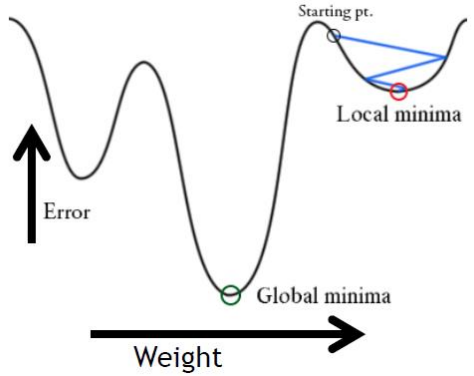
Learning rate selection



- There is no clear method for determining α
- Use variable, step-dependent α
  - Starting from an initial large value, we reduce α in m steps.
- Use an adaptive *α*
  - változtatását attól tesszük függővé, hogy a súlymódosítások csökkentik-e a kritériumfüggvény értékét

What initial weight values to set?

- There is no clear method
- In the absence of a priori knowledge, random weight adjustment is best suited
  - each weight can be chosen for different values of a uniformly distributed probability variable



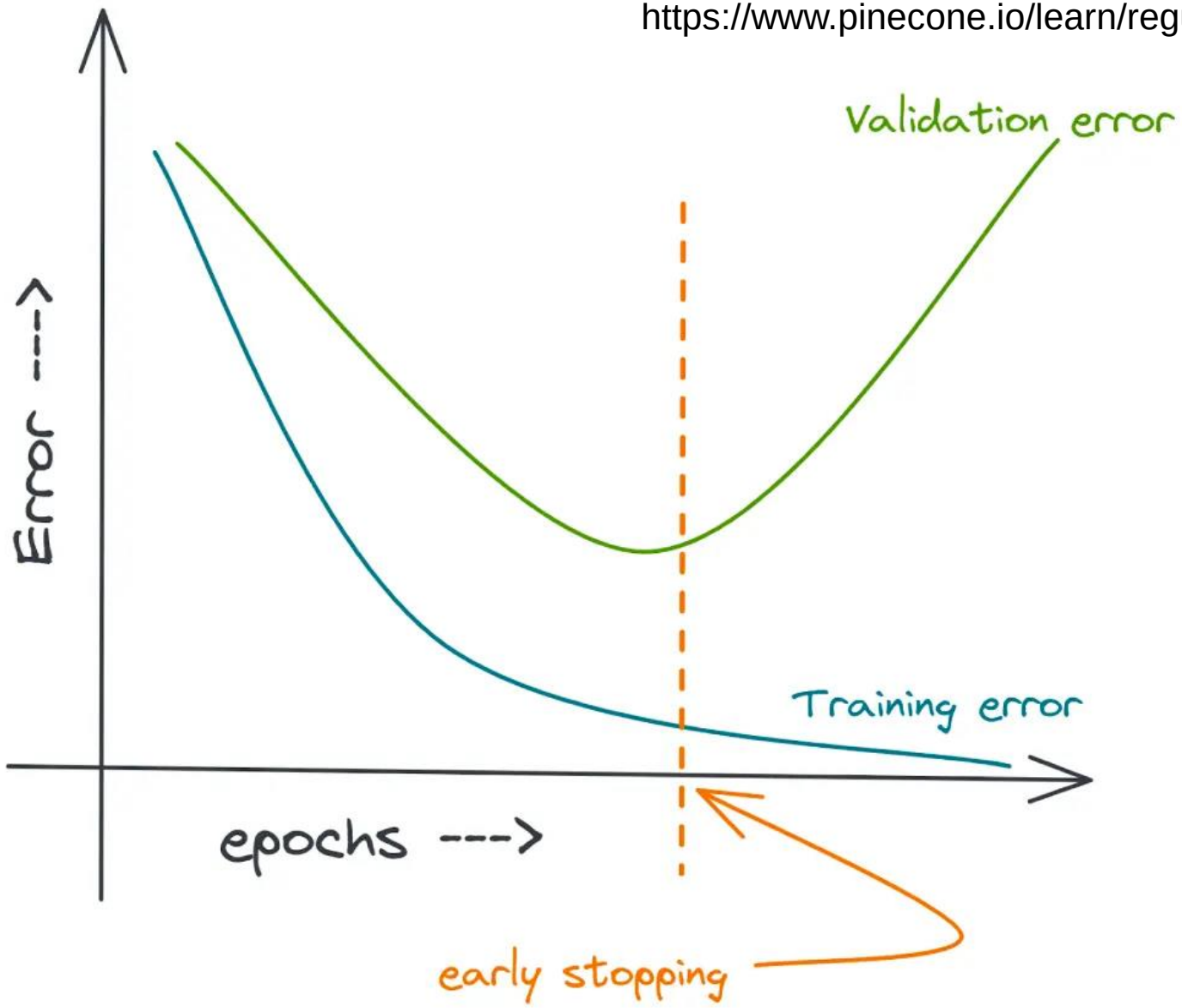
- With the Sigmoid activation function, avoid saturation of the output of hidden layers at the beginning of the training
  - The larger the network, the smaller random values you should choose

How to choose training, validation and testing dataset?

- Teaching dataset: We use it to train the model
- Validation dataset: We don't use it for weight adjustment, but we do use it to control when teaching stops
- Testing dataset: We don't use it for teaching, only for evaluation

How long should we teach the network?

https://www.pinecone.io/learn/regularization-in-neural-networks/



How to choose training, validation and testing dataset?

- For a large number of samples, it is not a problem to divide the data set into up to three equal parts

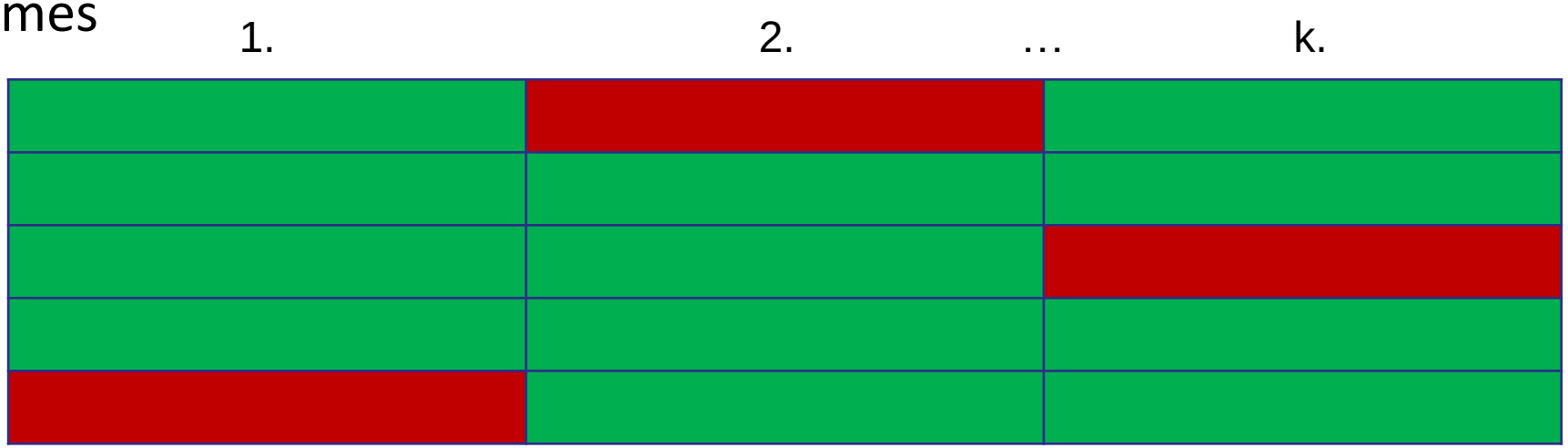
- However, for most real problems, the pattern is "never enough"
  - There are theoretical and practical limits to the number of training points required based on the size of the mesh and the expected error on the test kit
    - Theoretical minimum sample size: probably approximately correct – PAC learning
      - Less applicable in practice
    - Minimum practical sample size: based on previous experience
  - Application of cross-validation methods

Dataset partitioning

- Rule of thumb: Training – Validation – Test : 80%-10%-10%
- Datasets can be partitioned multiple times

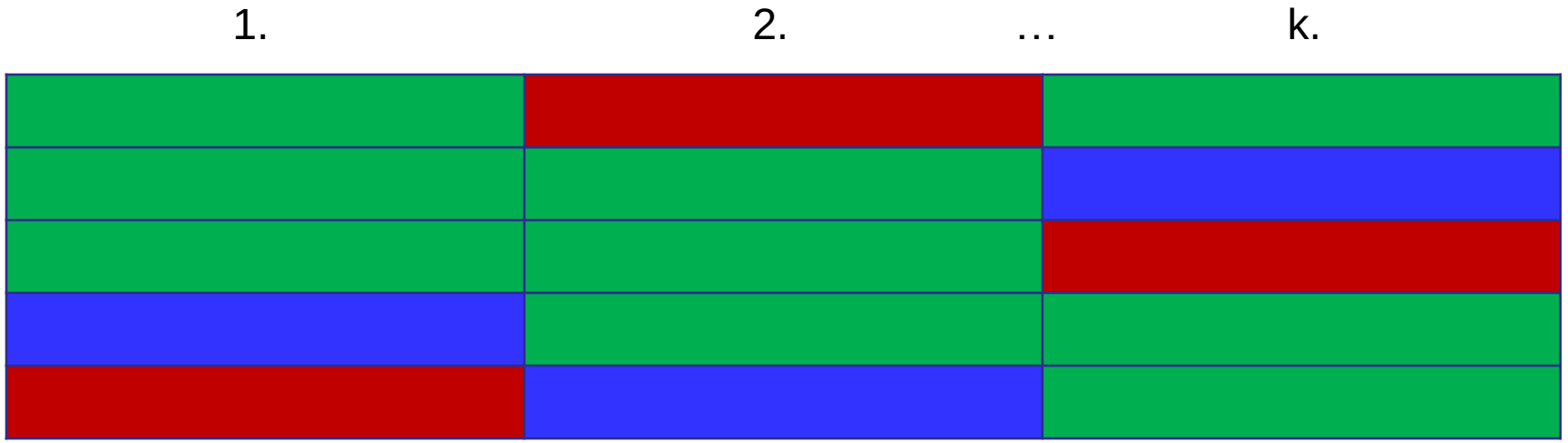
Cross-validation methods

- K-cross-validation : Partition a dataset into k parts
  - When training, k-1 partitions will be used as the training, The k-th is the test data set
  - Training repeated K times



Dataset partitioning

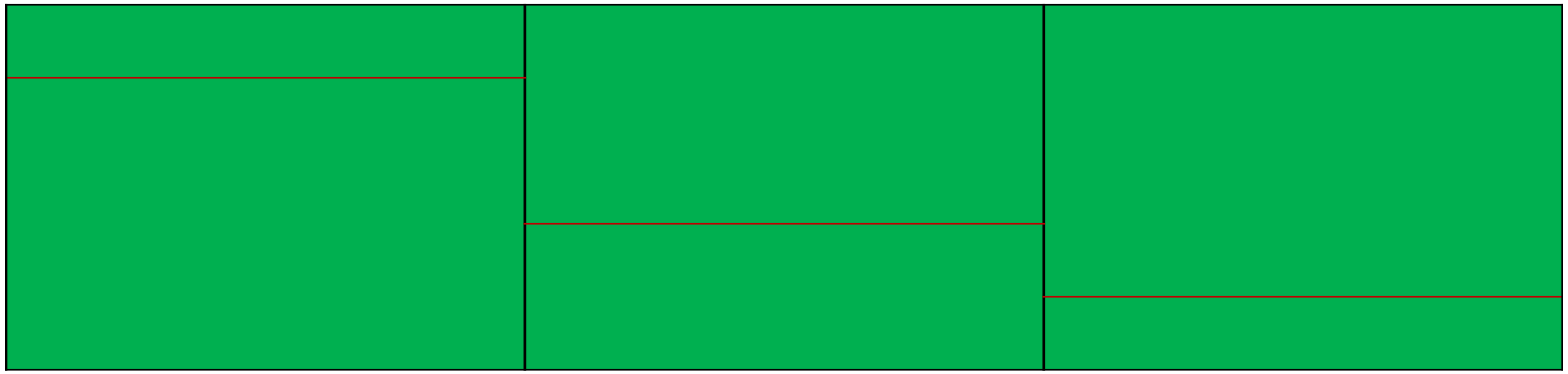
- K-cross-validation : if Training – Validation – Test datasets are needed
- For each training: k-2 partitions will serve as the training dataset, one remaining partition for validation and the other for test data set
- Training is repeated k - times



Dataset partitioning

- Leave-one-out Cross-validation
  - There is always only one sample of the test, all other samples are for training

1. 2. … n.



Cross-validated performance metrics

- It can be used to investigate a possible range of performance metrics
  - range
  - mean, median, standard deviation
  - confidence interval: CI, pl. 95%
- Measures robustness

In case of k=3

- TPR Average: 0.84
- Standard deviation: 0.0374
- 95%-os confidence interval (Student-t): 0.84±0.0929
- [0.7471, 0.9329]

In case of k=10

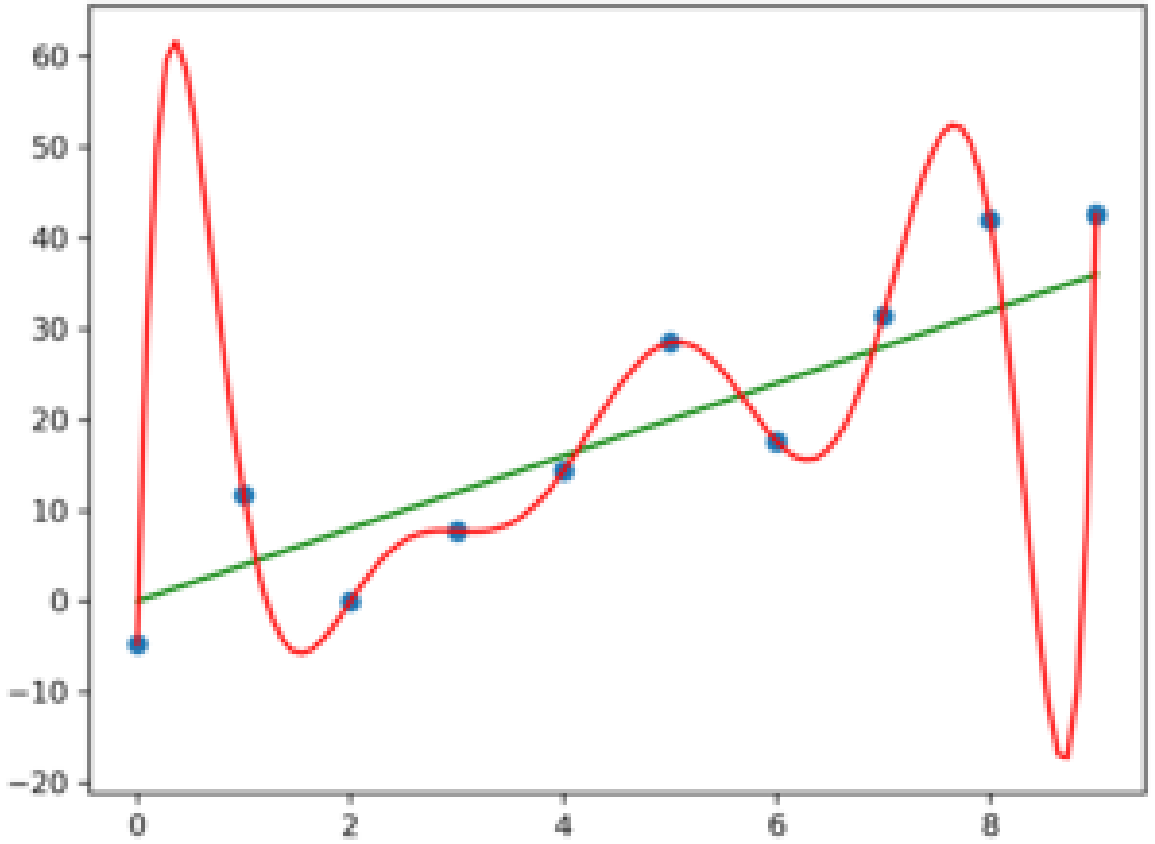
- 0.84±0.02677
- [0.8132, 0.8668]

How to use the samples, how often should we change the weights of the network?

- Per sample
  - Weight adjustment per sample
- Per batch
  - Weight modification per size k batch of samples
  - Only after using the entire training set

How to protect the model from overfitting?

- Overfitting occurs when the answers to the patterns of the training data set are received with very small errors, while the network responds to the validation data set with an increasing error.



https://en.wikipedia.org/wiki/Overfitting

How to protect the model from overfitting?

- The network's responses are too aligned with the mapping given by a finite number of training points
- Cause: the size (degree of freedom) of the network is too large compared to the number of training points

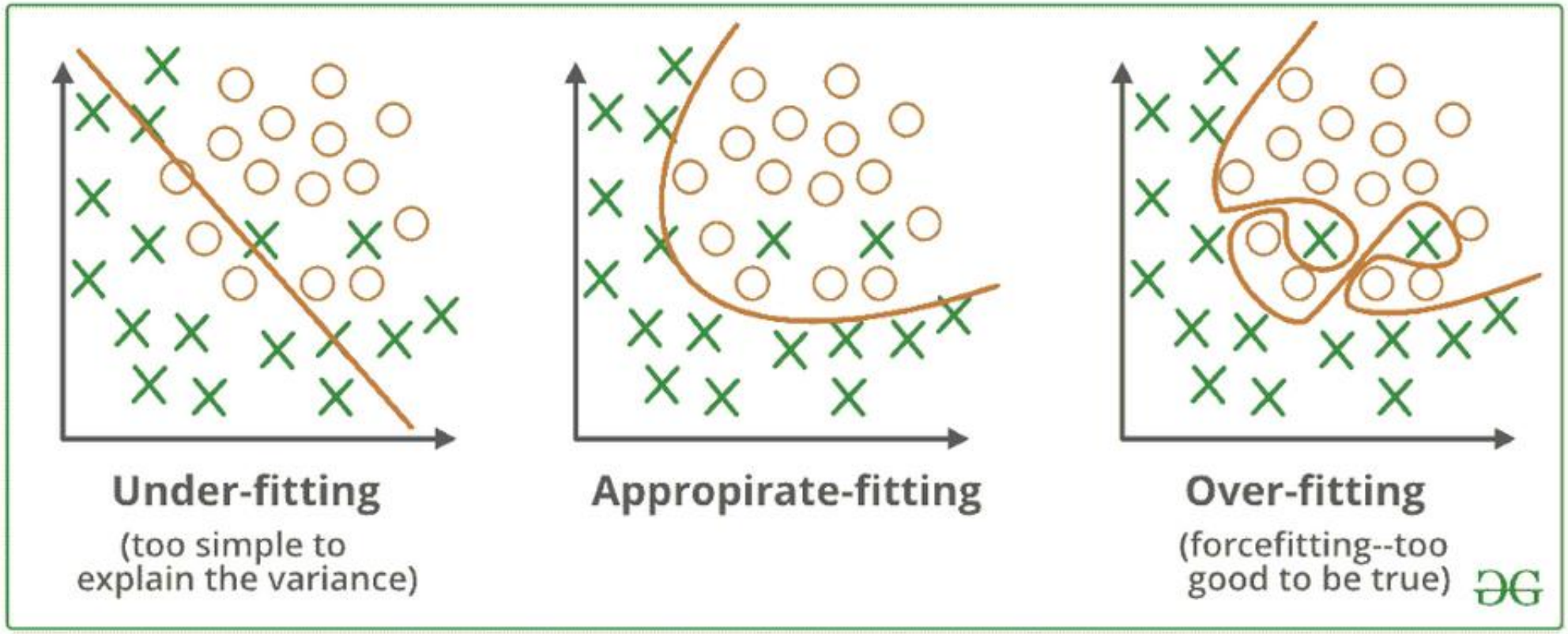
Why overfitting should be avoided?

- Overlearning impairs generalizing ability.

Methods

- Regularization techniques
- Validation dataset– early stopping

Overfitting - underfitting



https://www.geeksforgeeks.org/regularization-in-machine-learning/

Bias – Variance compromise

- Balancing bias and variance
- Regularization can help balance between
  - Model bias (underfitting) and
  - Model variance (overfitting)

- There is an inverse relationship between bias and variance. When one decreases, the other tends to increase, and vice versa.

Model bias

Bias refers to the errors that occur when trying to fit a statistical model to real data

- The model is unable to learn patterns in the available data and therefore performs poorly.
- If we use an overly simplistic model to fit into the data, we are more likely to face a situation of high bias

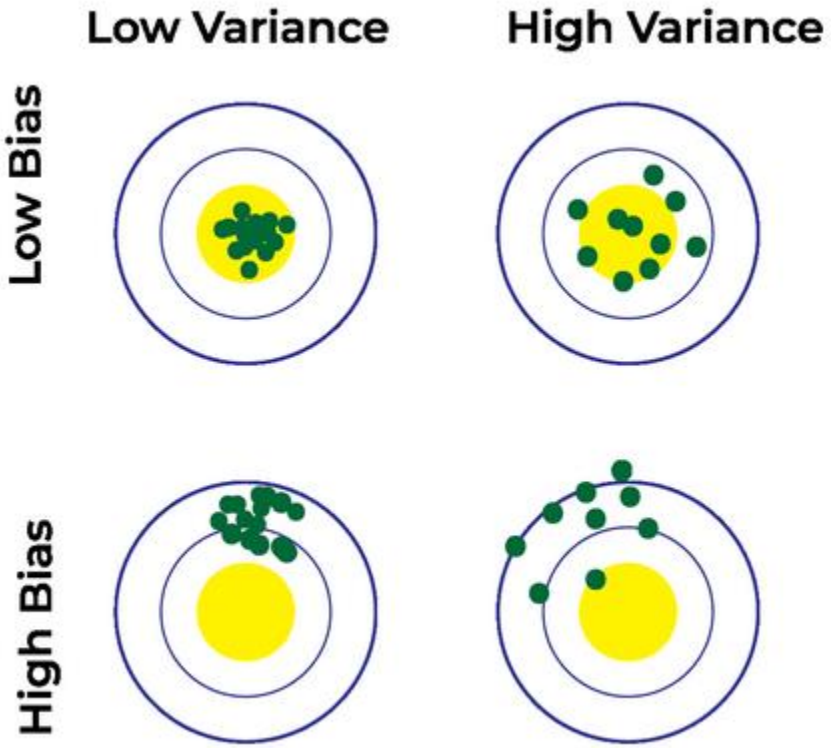
Model variance

- Variance refers to the error that occurs when trying to make predictions using data previously unseen by the model.
- High variance occurs when the model learns the noise present in the data.

Bias – variance combinations

Low Bias, Low Variance:
- This is the perfect scenario.
- Generalize well and make consistent, accurate predictions.
- In reality, however, this is typically not feasible.

High Bias, Low Variance:
- A model with high bias and low variance is considered as underfit.



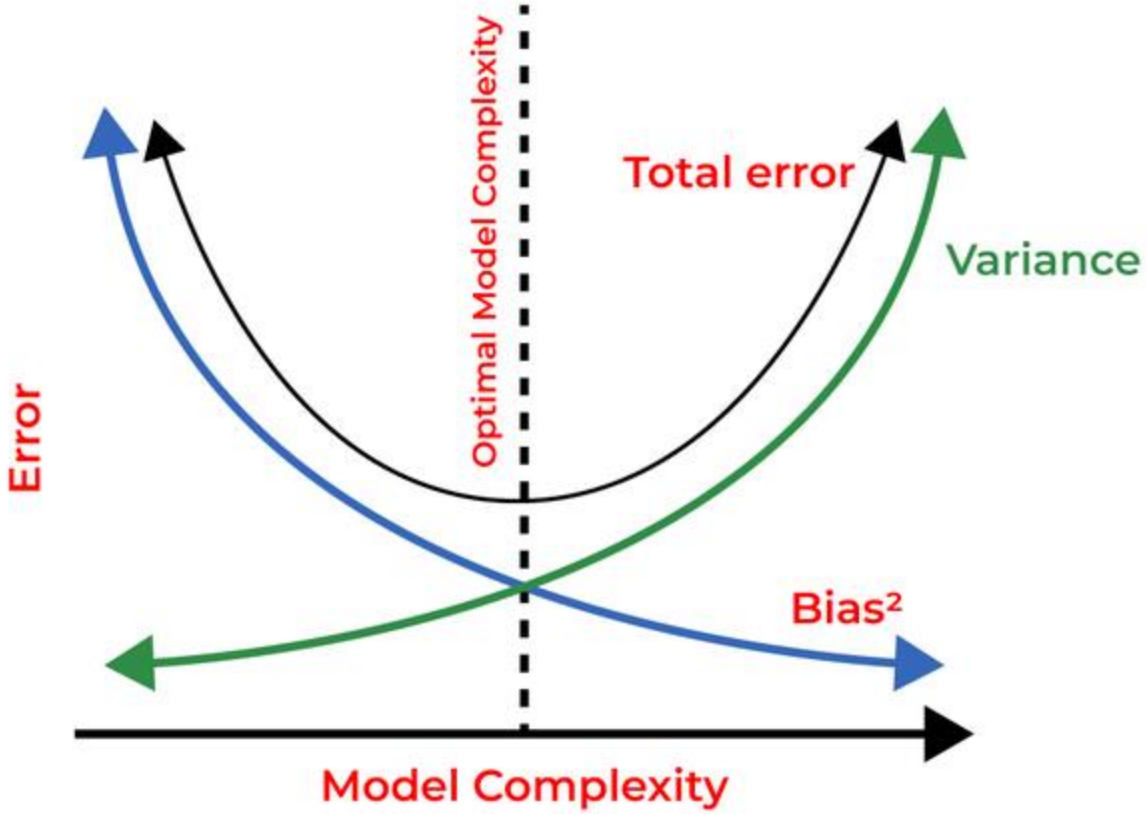
High Variance, Low Bias:
- A model with high variance and low bias is considered as overfitting.

High-Bias, High-Variance:
- Models with high bias and high variance cannot learn the underlying patterns and are too sensitive to changes in training data.
- The model generates generally unreliable and inconsistent predictions.

https://www.geeksforgeeks.org/regularization-in-machine-learning/

Bias – variance compromise

- There is an inverse relationship between bias and variance. When one decreases, the other tends to increase, and vice versa.
- Models that are too simple, with high bias, do not capture the underlying patterns,
- while models that are too complex, with high variance, overfit the noise of the data.



https://www.geeksforgeeks.org/regularization-in-machine-learning/

Regularization

- Early stopping
- Data augmentation
- L1 and L2 regularization
- Add noise to input/labels/gradient
- Dropout

Bala Priya: Regularization in Neural Networks, 2023 összefoglalója alapján

Regularization

Early stopping

- If the error/loss becomes too low and arbitrarily approaches zero, then the network will surely fit beyond the training data set.
- Therefore, if we can heuristically prevent error/loss from being arbitrarily low, the model is less likely to fit beyond the training data set and will generalize better.

Early stopping 2.

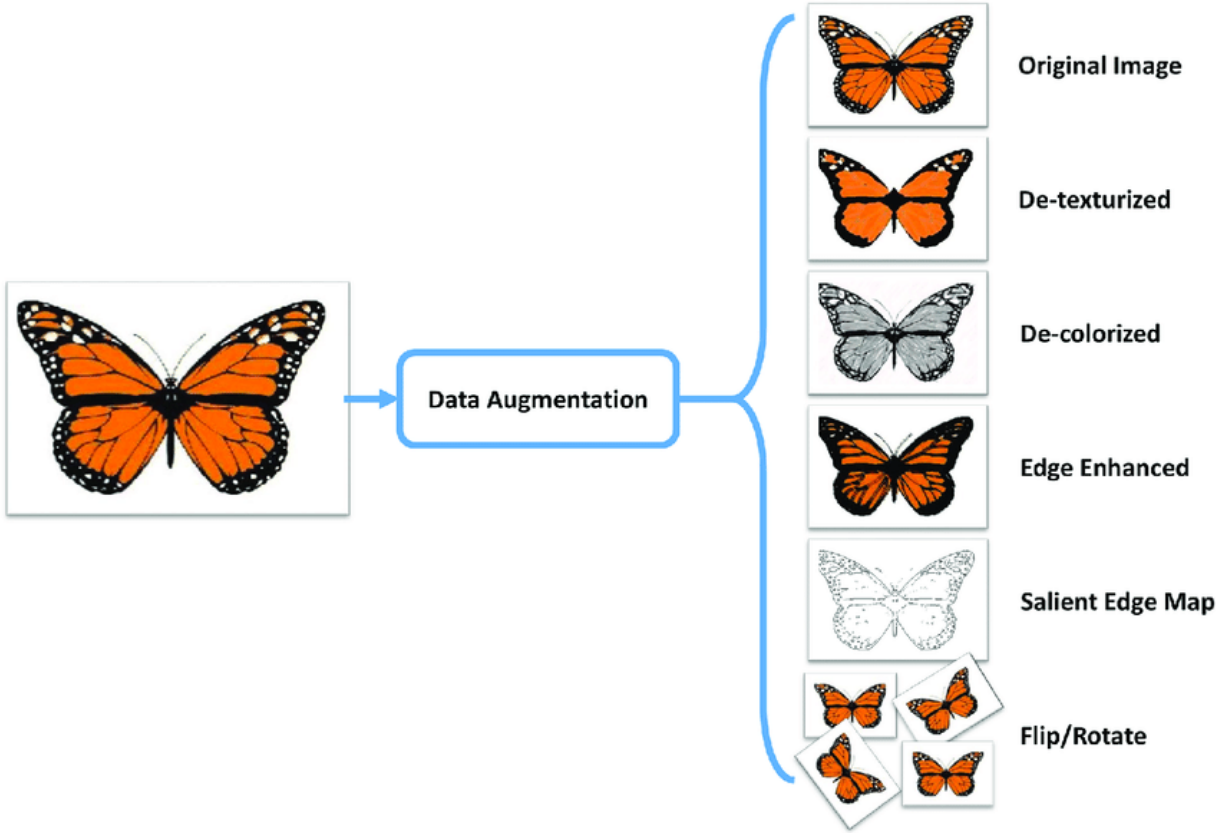
Another way to know when to stop:

- Monitor changes in network weights
- let w_t és w_{t-k} be the weights of epochs *t* and *t-k*
- we calculate the L2 norm of the difference vector
- If this amount is less than ε, we can stop training.

$$||w_t - w_{t-k}||_2 < \epsilon$$

Data augmentation

- Data augmentation is a regularization technique that helps the neural network generalize better by exposing it to more diverse training patterns.



- Because deep neural networks require large training datasets, data augmentation is also useful when we don't have enough data to train a neural network.

L1 and L2 Regularization

- In general, Lp norms (for p>=1) penalize heavier weights. They force the norm of the weight vector to remain sufficiently small. In n-dimensional space, the norm Lp of the vector x is as follows:

$$Lp(\mathbf{x}) = ||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- L1 norm: If p=1, then we get the norm L1, which is the sum of the absolute values of the components of the vector:

$$L_1(\mathbf{x}) = ||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$$

L1 and L2 regularization

- L2 norm: If p=2, then we get the norm L2, which is the Euclidean distance of the point from the origin in n-dimensional vector space:

$$L_2(\mathbf{x}) = ||\mathbf{x}||_2 = (\sum_{i=1}^{n} |x_i|^2)^{1/2}$$

L1 regularization

- Let L and L~ be Loss functions without and with regularization. The regularized loss function L1 is as follows:

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \alpha||\mathbf{w}||_1$$

- where α is the regularization constant.
- If the weights become too large, the second term increases.
- However, since the purpose of optimization is to minimize the loss function, the weights should decrease.

L1 Regularization

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \alpha||\mathbf{w}||_1$$

- where α is the regularization constant.
- If the weights become too large, the second term increases.
- However, since the purpose of optimization is to minimize the loss function, the weights should decrease.
- In particular, L1 regularization promotes the rarity of the weight matrix by forcing weights to zero.
- Thus L1 also performs feature selection.

L2 Regularization

- Similarly, the regularized loss function with L2 regularization (also known as L2 weight loss) is as follows::

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\alpha}{2}||\mathbf{w}||_2{}^2$$

- If the weights become too large, as a result, the second member of the above equation will increase.
- However, since the goal is to minimize the loss function, the algorithm prefers smaller weights.

L2 regularization

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\alpha}{2}||\mathbf{w}||_2{}^2$$

- L2 regularization reduces weights while ensuring that important components of the weight vector are larger than others.

Noise addition to input/labels/gradients

Add noise to...

- Inputs
- Labels
- Gradients

Noise addition to inputs

Add noise to inputs

$$\hat{x}_i = x_i + \epsilon_i$$

$$\hat{y}_i = \sum_{i=1}^{n} w_i \hat{x}_i$$

- For a sum of squares loss function, adding normally distributed noise to inputs is equivalent to L2 regularization

Noise addition to labels

Add noise to labels

- If „noise is added to output labels”, the network cannot learn the training data set completely
- Perturbation of labels
  - Each learning sample is "disrupted" with probability p.
  - For each disturbed sample, the label shall be drawn from a uniform distribution {1,2,3,…,N} regardless of the real label.
- Label smoothing

Class Label	1	2	3	…	k	…	N
With Label Smoothing	$\frac{1-\epsilon}{N-1}$	$\frac{1-\epsilon}{N-1}$	$\frac{1-\epsilon}{N-1}$	…	ϵ	…	$\frac{1-\epsilon}{N-1}$

Noise addition to gradients

Add noise to gradients

$$G_{w_i}^t = G_{w_i}^t + \mathcal{N}(0, \sigma_t^2)$$

Where $G_{w_i}^t$ is the calculated gradient vector (Wi for weights) in step t, and $\mathcal{N}(0, \sigma_t^2)$ is Gaussian noise with an expected value of 0 and variance of σ_t^2.

Variance may vary (decrease) over time (depending on step count)

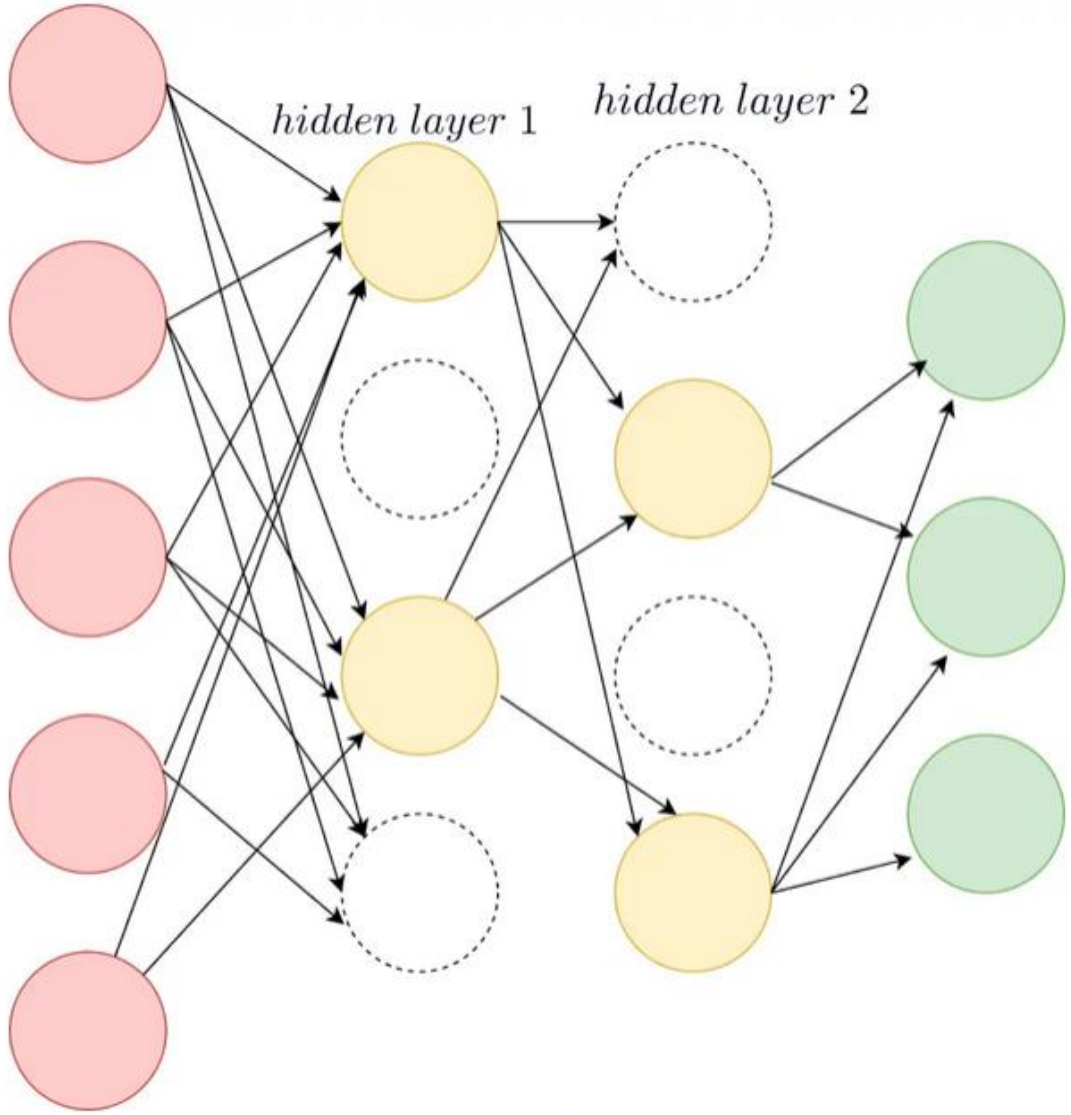
$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

Dropout

- A dropout means disabling neurons in the hidden layers and (optionally) in the input layer.
- During training, each neuron is assigned a "dropout" probability, for example, 0.5.
- With a dropout probability of 0.5, there is a 50% chance that a neuron will participate in training within each training session (batch).

Dropout

- This leads to the fact that the architecture of the network is slightly different in each batch.
- This is equivalent to training different neural networks on different subsets of training data.



Dropout

https://www.pinecone.io/learn/regularization-in-neural-networks/

Hyperparameters

Hyperparameters can answer questions about model design.

- What should be the maximum depth of the decision tree?
- What should be the minimum number of samples required in one leaf node of the decision tree?
- How many trees to include in a random forest?
- How many neurons should be in one layer of the neural network?
- How many layers should a neural network have?
- What learning rate to set for gradient descent?

Tuning hyperparameters

- Model definition
- Define the range of all possible values for each hyperparameter
- Define a sampling method to select hyperparameter values
- Define and calculate criteria to evaluate the goodness of the model
- Selection of a cross-validation method