Deep Learning for Physicists

Lecture #3: Practical methodology

Kosmas Kepesidis

What was covered in the previous lecture…

- During training, the data set is used multiple times – each time is called ***Epoch***
- Parameter optimization is done in smaller steps using only samples of the data set called ***minibatches***
- Weight coefficients are initialized using random numbers following a distribution (normal or uniform)
- Common ***objective functions*** for regression are ***MAE***, ***MSE***, ***RMAE*** and for classification we use ***cross-entropy***
- With the help of the chain rule of partial derivatives (***backpropagation***), ***stochastic gradient descent*** minimizes ***objective function***
- ***Learning rate*** corresponds to the steps size of the optimization procedure

Outline

- Practical methodology
  - Criteria for model training
  - Train, validation and test data sets
  - Monitoring
  - Regularization
  - Hyperparameters

Criteria for model training

Criteria for model training

- Objective function for regression:
  - Distance measure between predictions $f(x_i)$ and target values $y(x_i)$, where i runs over data points
    - Mean absolute error (MAE) – *Manhattan norm* $\mathcal{L} = \frac{1}{k}\sum_{i=1}^{k}|f(x_i) - y(x_i)|$
    - Mean squared error (MSE) $\mathcal{L} = \frac{1}{k}\sum_{i=1}^{k}[f(x_i) - y(x_i)]^2$
    - Root mean squared error (RMSE) – *Euclidean norm* $\mathcal{L} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}[f(x_i) - y(x_i)]^2}$

Criteria for model training

- Model generalization: regression case

  - n data points (x_i, y_i) generated by a stochastic process: $y_i = g(x_i) + \varepsilon$

    - $g(x_i)$: probability distribution

    - ε: noise term following standard normal distribution $\sim N(0, \sigma)$

  - A network output $f(x)$ is optimized using $MSE = \left\langle (f - y)^2 \right\rangle$

  - Bias-variance relation: $MSE = B[\langle f \rangle, \langle g \rangle]^2 + V[f] + \sigma^2$

    - $B[\langle f \rangle, \langle g \rangle]$: bias term, i.e. displacement of the of the expectation value of the network prediction versus true value

    - $V[f]$: variance of network predictions

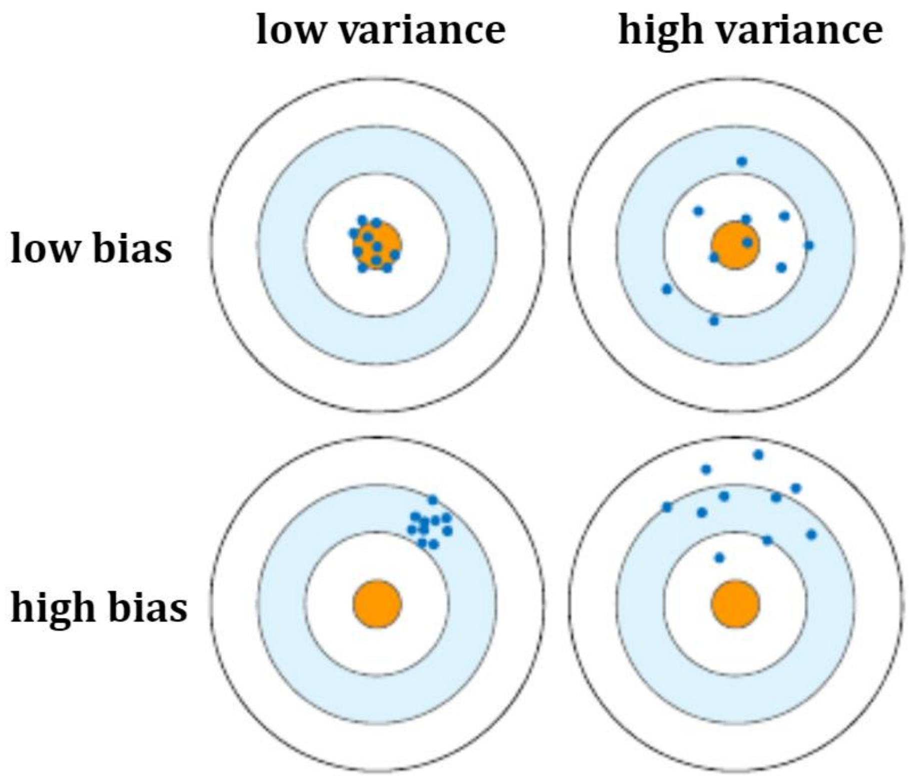
    - σ^2: noise variance – irreducible uncertainty

C. Sammut and G. I. Webb (eds.), Encyclopedia of Machine Learning and Data Mining, 2nd edn., Springer Reference

Criteria for model training

- Model generalization: regression case

  - $MSE = B[\langle f \rangle, \langle g \rangle]^2 + V[f] + \sigma^2$



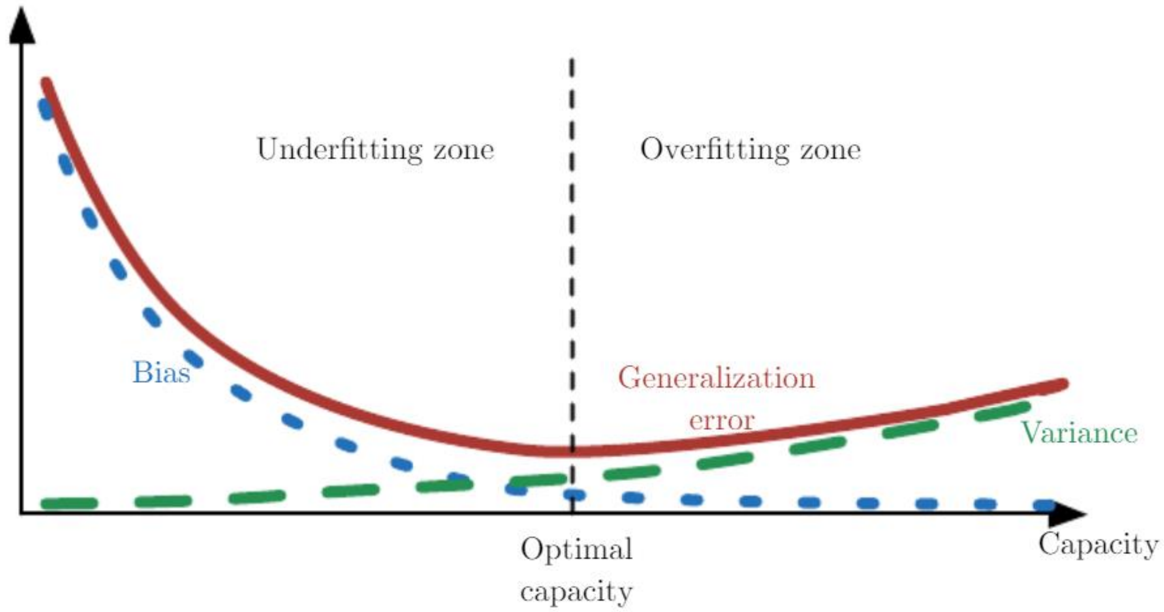
  - Models with low bias and low variance have **generalization** capability

C. Sammut and G. I. Webb (eds.), Encyclopedia of Machine Learning and Data Mining, 2nd edn., Springer Reference

Criteria for model training

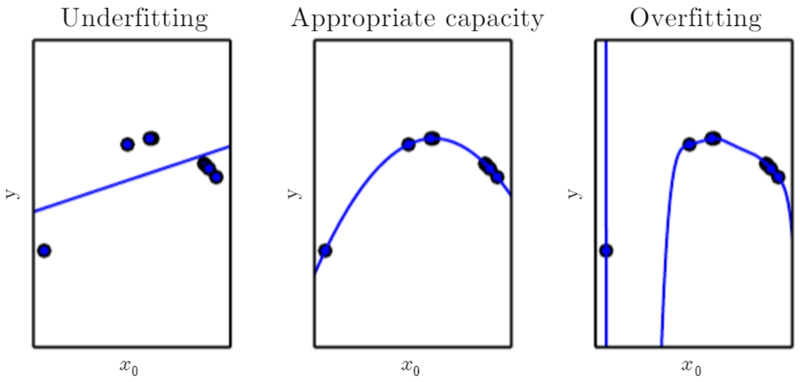
- Model generalization

  - Bias-variance tradeoff:



Criteria for model training

- **Overfitting/underfitting**
  - Network model should capture complexity of true distribution without being more complicated that it should
    - Oversimplified models can cause underfitting
    - Too complicated models can potentially capture fluctuations in the data (overfitting)

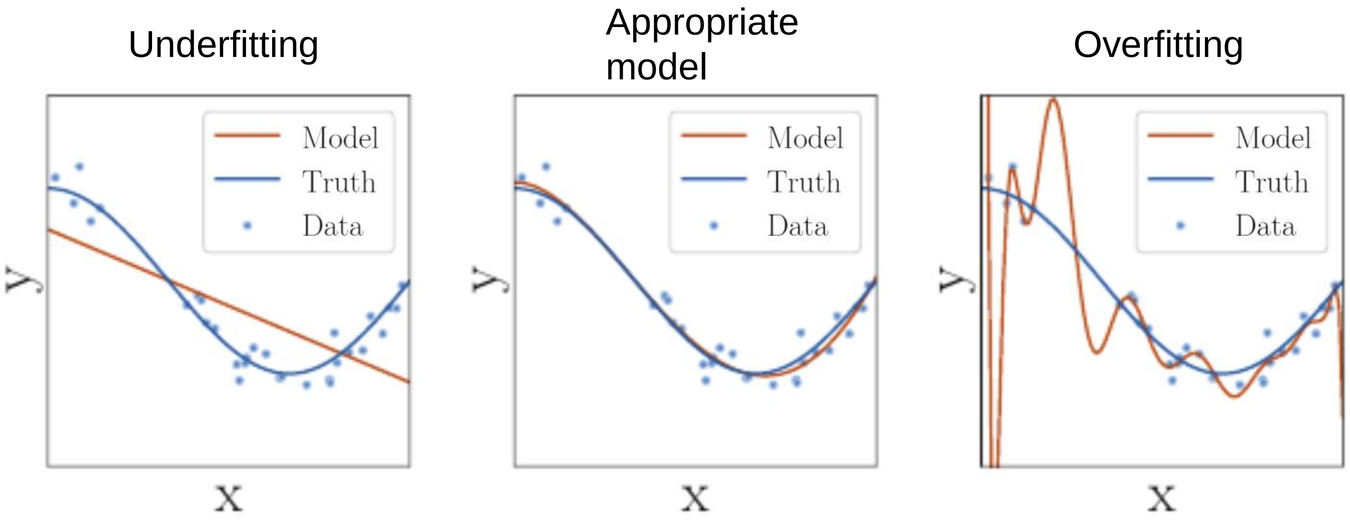


    - DNNs have many parameters and are prone to overfitting
    - training should be monitored!

Criteria for model training

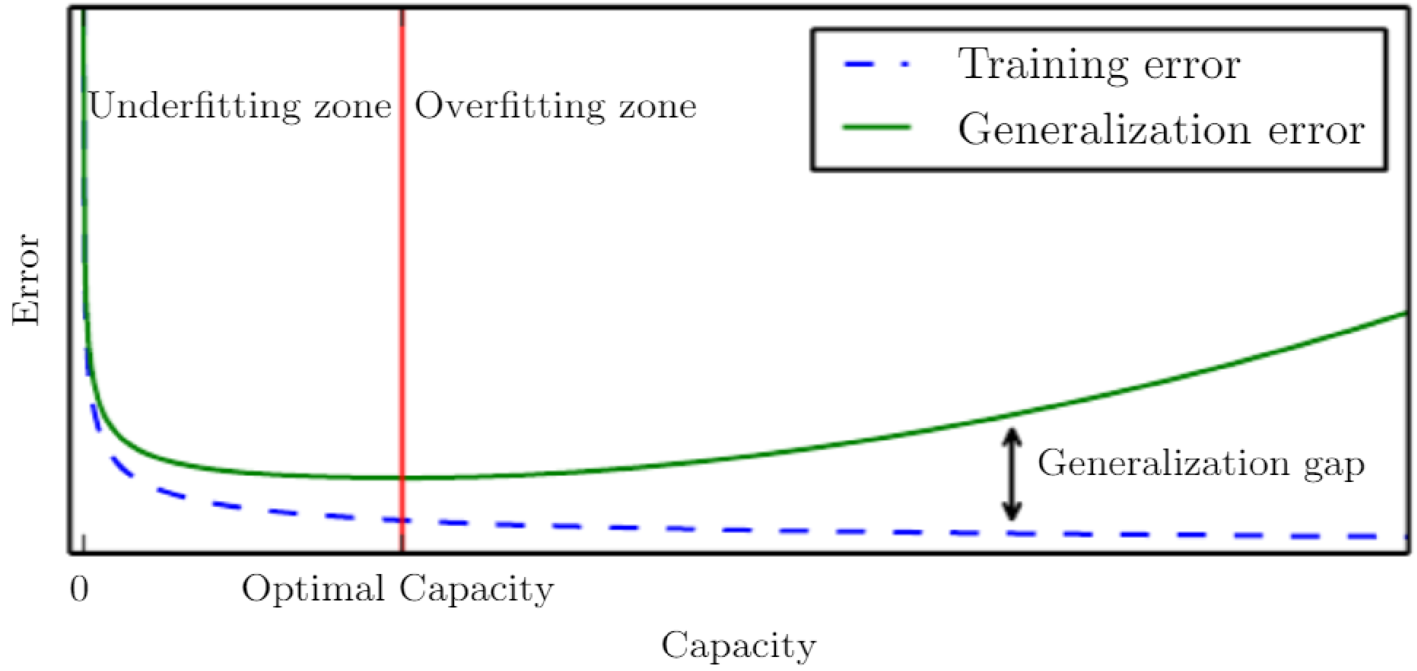
- **Overfitting/underfitting**

  - Network model should capture complexity of true distribution without being more complicated that it should



Criteria for model training

- **Overfitting/underfitting**



Criteria for model training

- **Overfitting/underfitting**

  - Network model should capture complexity of true distribution without being more complicated that it should

    - Learner finds a pattern in the data that is not actually true in the real world: overfitting
      - Happens when you have too many hypotheses and not enough data to tell them apart
      - The more data, the more “bad” hypotheses are eliminated

    - If the hypothesis space is not constrained, there may never be enough data
      - There is often a parameter that allows you to constrain (regularize) the learner

Criteria for model training

- **Overfitting/underfitting**
  - ➢Why does overfitting occur and how we avoid it:
    - ▪ Happens when you have too many hypotheses and not enough data to tell them apart
→ collect more data
    - ▪ Data is noisy
→ collect better data (reduce noise)
    - ▪ Models are too complex → use less complex models
    - ▪ Aggressive loss optimization → optimize less

Criteria for model training

- Evaluation metrics for classification tasks
    - ➢**Confusion matrix** for binary classification

		Predicted class	
		Positive	Negative
Actual class	Positive	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

Criteria for model training

- Evaluation metrics for classification tasks
  - **Confusion matrix** for binary classification
    - Example: spam emails

	spam (predicted)	not_spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not_spam (actual)	12 (FP)	556 (TN)

    - Question: how accurate is this classifier?

Criteria for model training

- Evaluation metrics for classification tasks

  - **Confusion matrix** for binary classification

    - **Accuracy**:

      how many predictions are correct?

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Overall measure of classification performance
- Can be problematic if classes are imbalanced

		Predicted class	
		Positive	Negative
Actual class	Positive	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

Criteria for model training

- Evaluation metrics for classification tasks

  - **Confusion matrix** for binary classification

    - **Precision**:
    how many of the positive predictions are correct?

$$precision = \frac{TP}{TP + FP}$$

    - **Recall** (Sensitivity):
    how many of positives were correctly identified?

$$recall = \frac{TP}{TP + FN}$$

		Predicted class	
		Positive	Negative
Actual class	Positive	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

Criteria for model training

- Evaluation metrics for classification tasks

  - **Confusion matrix** for binary classification

    - **Precision - Recall**:

    Can be combined into a single measure: $F1$-score

$$F1 = 2\frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

    - $F1$-score favors classifiers with balanced precision and recall

		Predicted class	
		Positive	Negative
Actual class	Positive	True positives (TP)	False negatives (FN)
	Negative	False positives (FP)	True negatives (TN)

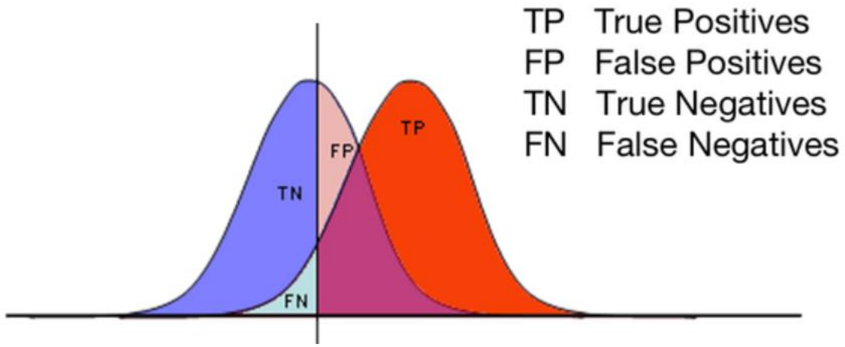
Criteria for model training

- Evaluation metrics for classification tasks

  - **Precision/recall tradeoff**:

  One typically has to chose between high precision or high recall – increasing one reduces the other

  - Consider classification task: class A or class $\bar{A}$ (i.e. *not* class A)
    - A fixed threshold for the classification can be used: $p_A > 0.5$
    - Alternatively, $p_A > v, \quad v \in [0,1]$
    - Threshold v is referred to as **operating point** and it is chosen such the values of precision and recall



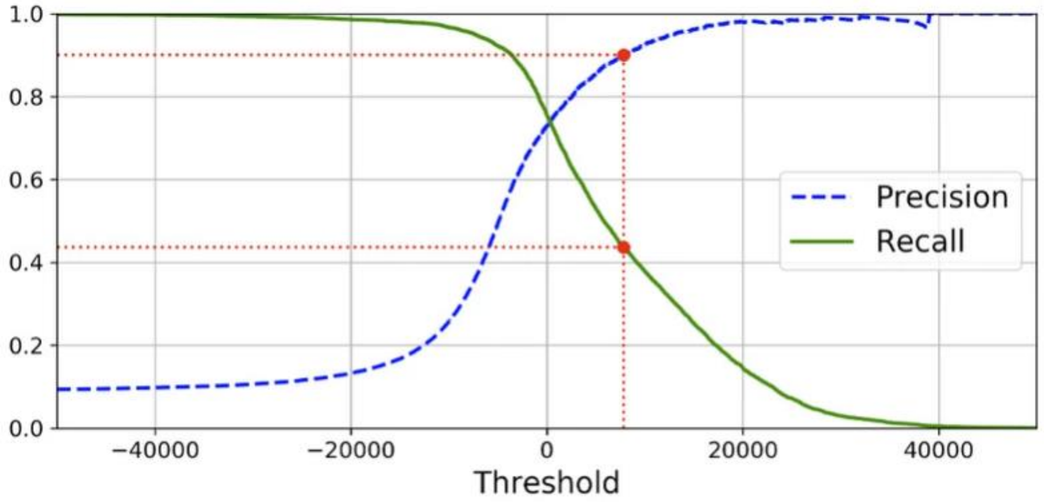
Source: https://datascience-george.medium.com

Criteria for model training

- Evaluation metrics for classification tasks

  ➢ **Precision/recall tradeoff**:

  One typically has to chose between high precision or high recall – increasing one reduces the other

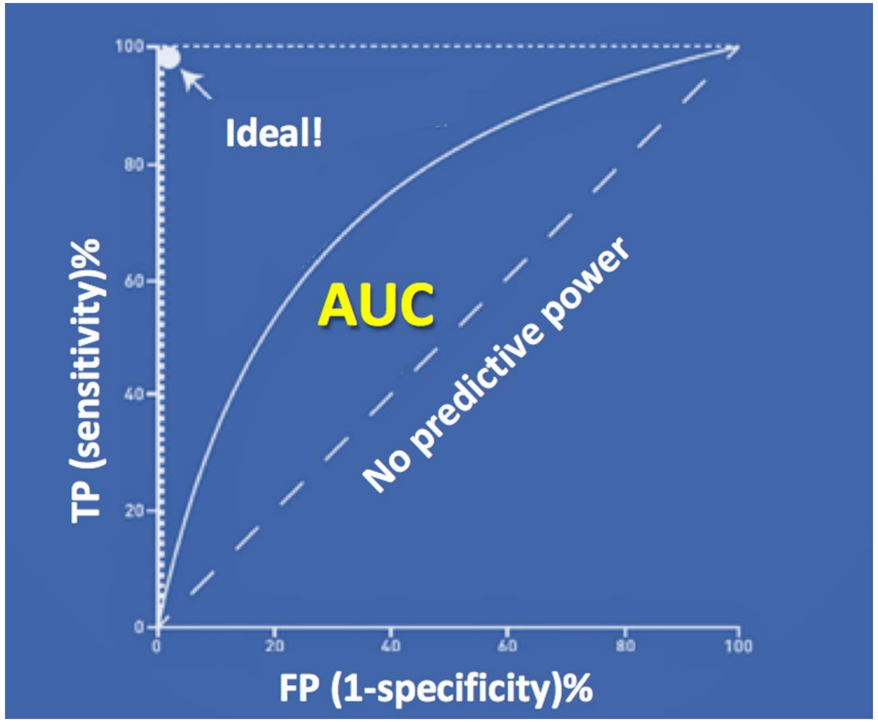


Source: A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow

Criteria for model training

- Evaluation metrics for classification tasks
- **receiver operating characteristic** (ROC) curve: a very useful tool for binary classification

➢ Y-axis: **true-positive rate** (TPR) – also known as recall
➢ X-axis: **false-positive rate** (FPR)



$$TPR = \frac{TP}{TP + FN} = \text{recall} \qquad FPR = \frac{FP}{FP + TN}$$

➢ **Area under the curve** (AUC):
➢ a threshold-independent metric for binary classification!

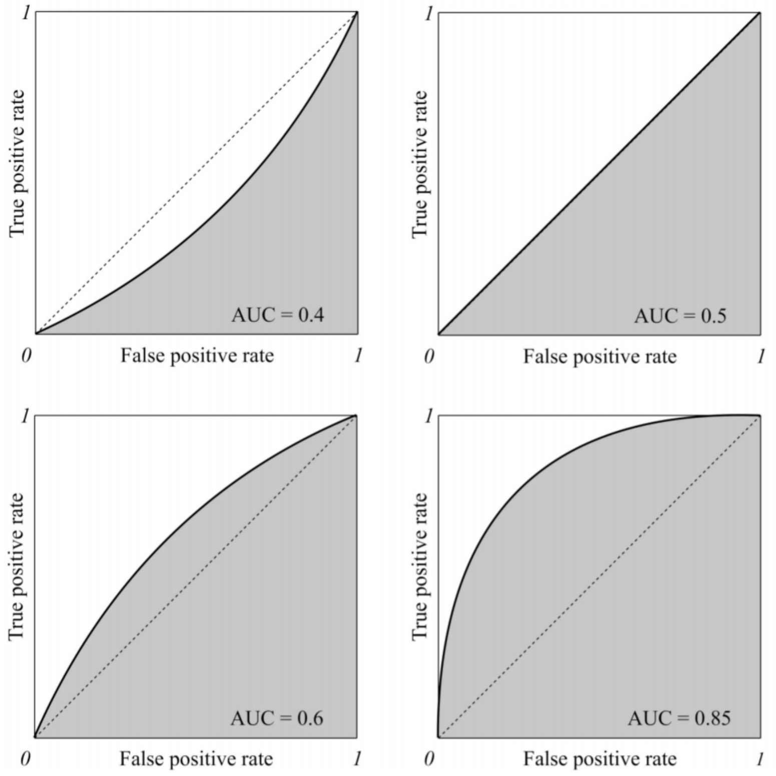
Criteria for model training

- Evaluation metrics for classification tasks
- **receiver operating characteristic** (ROC) curve: a very useful tool for binary classification

  - Y-axis: **true-positive rate** (TPR) – also known as recall
  - X-axis: **false-positive rate** (FPR)

$$TPR = \frac{TP}{TP + FN} = \text{recall} \qquad FPR = \frac{FP}{FP + TN}$$

  - **Area under the curve** (AUC):
  - a threshold-independent metric for binary classification!



Source: A. Burkov, Machine Learning Engineering

Criteria for model training

- Evaluation metrics for classification tasks
- **receiver operating characteristic** (ROC) curve: a very useful tool for binary classification

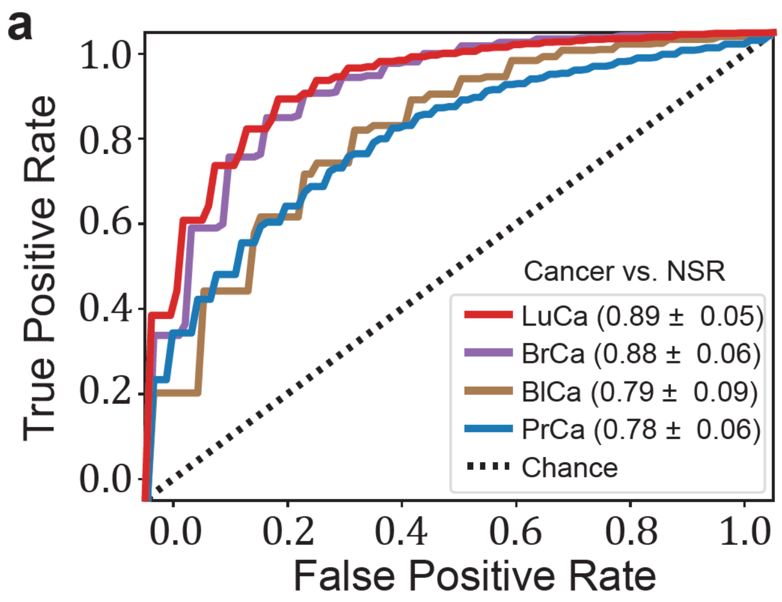
➢ Y-axis: **true-positive rate** (TPR) – also known as recall

➢ X-axis: **false-positive rate** (FPR)

$$TPR = \frac{TP}{TP + FN} = \text{recall} \qquad FPR = \frac{FP}{FP + TN}$$

➢ **Area under the curve** (AUC):

➢ a threshold-independent metric for binary classification!



Train, validation and test data sets

Train, validation and test data sets

- Only a subset of the available data can be used for training – the ability of a trained network to generalize must be evaluated on independent data

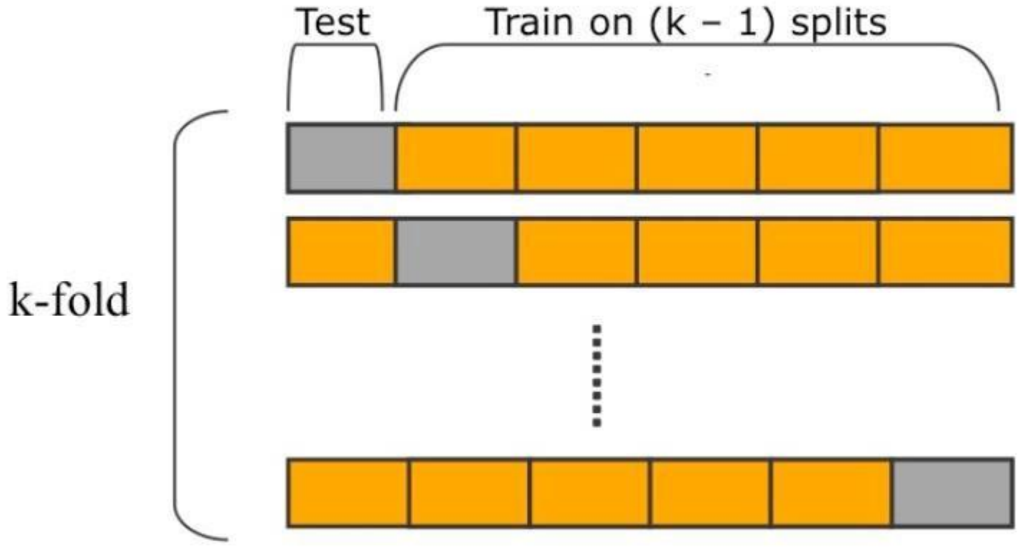
1. *Train set:*
   - Used for model training

2. *Validation set:*
   - Used for evaluating prediction quality
   - Improvements are made to the model afterwards
   - The network is retrained and reevaluated in an iterative procedure

3. *Test set:*
   - Used as a final test of the prediction performance
   - It cannot influence any decisions about the model

Train, validation and test data sets

- **cross-validation** (CV): a more advance method for evaluating network's performance
  - ➢ Split data into k equally-sized partitions
  - ➢ Use each part as a test set and combine the $k-1$ others for training
  - ➢ Obtain k results and average them

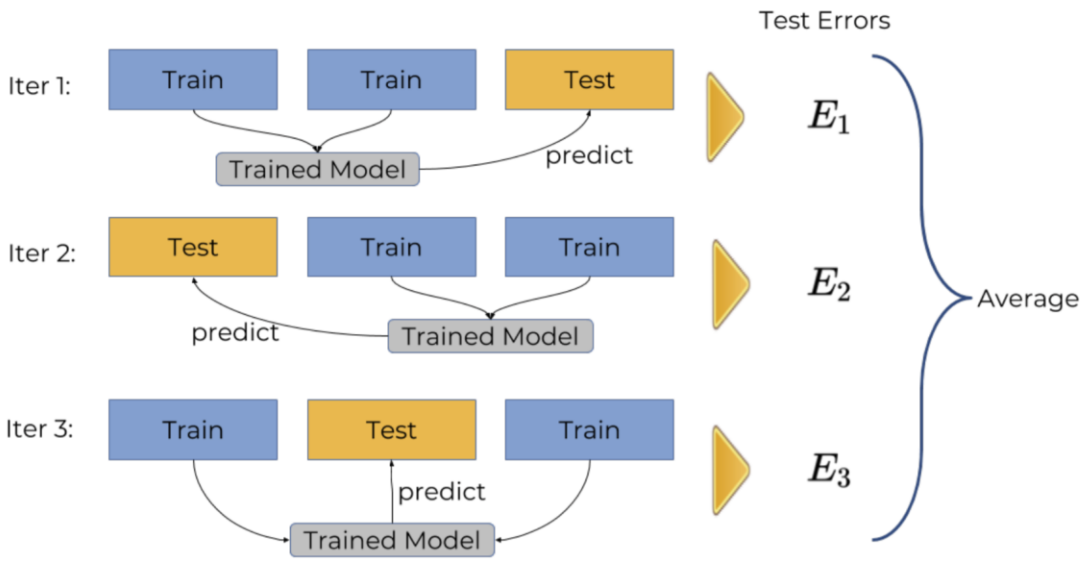


Train, validation and test data sets

- **cross-validation** (CV): a more advance method for evaluating network's performance

  - ➢ Split data into k equally-sized partitions
  - ➢ Use each part as a test set and combine the $k-1$ others for training
  - ➢ Obtain k results and average them

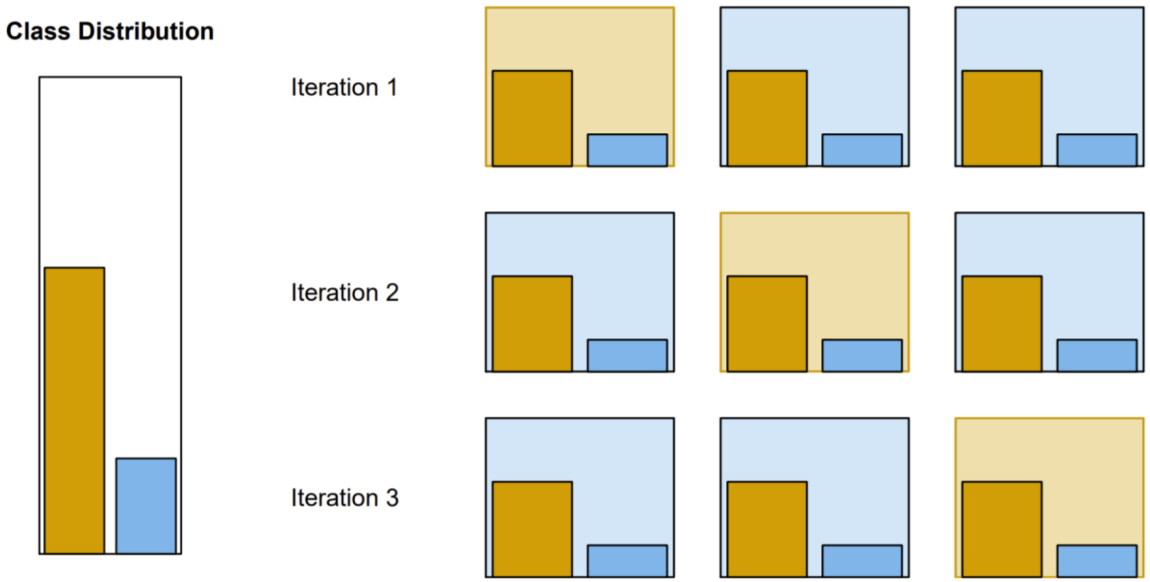
  - ➢ Example: 3-fold cross-validation



Train, validation and test data sets

- **cross-validation** (CV): a more advance method for evaluating network's performance

  - K-fold cross-validation can be performed in a stratified way



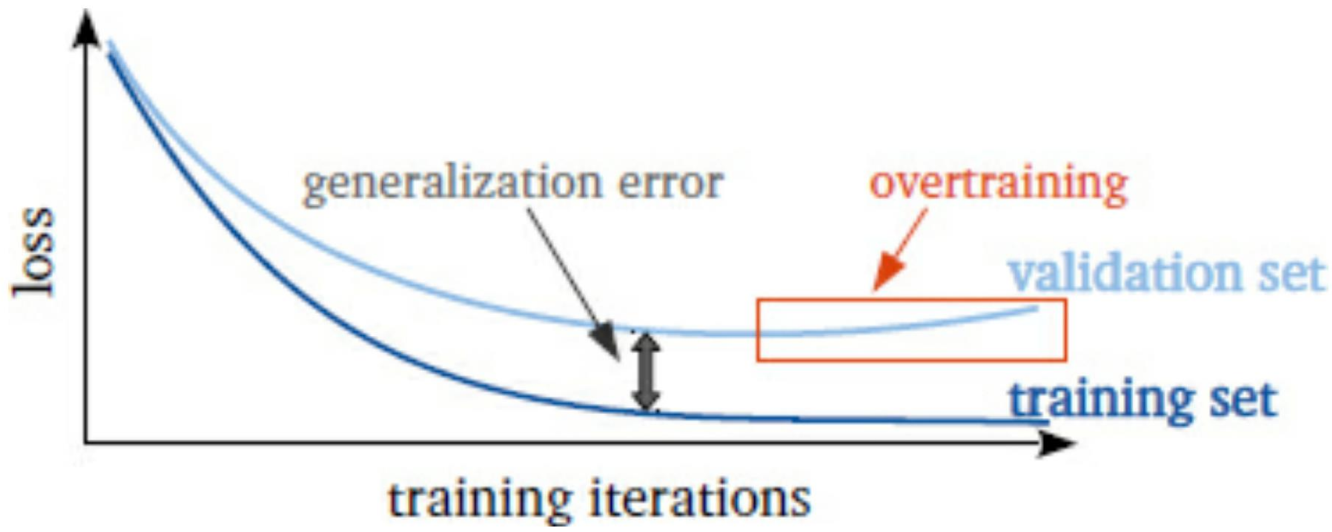
Train, validation and test data sets

- **cross-validation** (CV): a more advance method for evaluating network's performance
  - ➢ $k = 5$ or 10 are common choices: they use 80% or 90% of the data for training
  - ➢ For $k = N$ we obtain a method known as **leave-one-out** (LOO)
    - N-1 data points for training and 1 for testing
  - ➢ Performance estimates tend to be pessimistically biased (as the size of the training sets is smaller than n and we learn less)
  - ➢ This bias increases as k gets smaller. LOO is nearly unbiased, but has high variance
  - ➢ Repeated k-fold CV (multiple random partitions) can improve error estimation for small sample size.

Monitoring

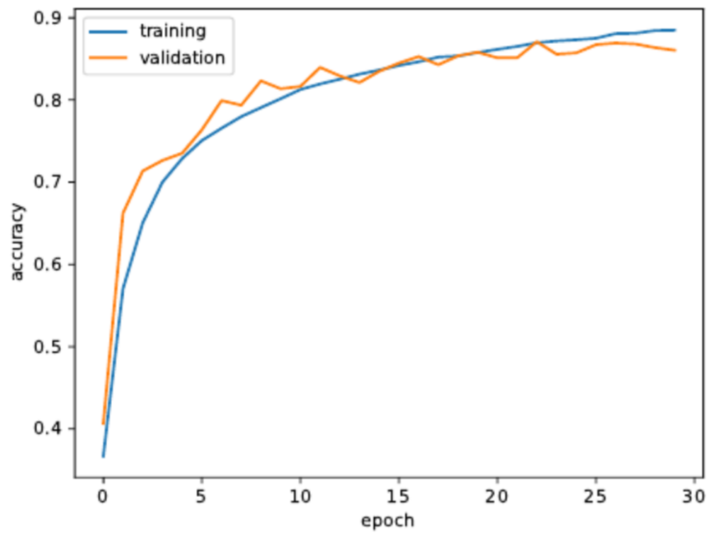
Monitoring

- Objective functions during training can provide useful information about the quality of model – can be plotted as function of epochs
  - Evaluate objective function on both the *train* and *validation* sets
  - Gap between them is referred to as **generalization error**



Monitoring

- Objective functions during training can provide useful information about the quality of model – can be plotted as function of epochs
    - ➢ Evaluate objective function on both the *train* and *validation* sets
    - ➢ Gap between them is referred to as **generalization error**
    - ➢ For classification tasks: *accuracy* or *AUC* are used instead



Regularization

Regularization

- It is possible that optimization gets stuck at an “unwanted” *local minimum,* resulting into overfitting – several **regularization** methods exist for overcoming this issue

  - Problem with “unwanted” local minima is that the network could learn properties in the training data that are not generally valid

  - Definition: *“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error”* I. Goodfelow et al.

Regularization

- Regularization methods:

  1. **Early stopping**

     - Critical point in training when/if the value of the objective function evaluated on the validation set starts rising – model starts to overfit

     - At this point, training is terminated to avoid further overfitting

Regularization

- Regularization methods:

2. Norm penalties

- ➢ Add a “penalty” term in the objective function to prevent the formation of large-valued parameter weights W – penalties are parameter-dependent
- ➢ Two variations:

1. L_1 norm $$\mathcal{L} = \mathcal{L}_{MSE} + \sum_{i=1}^{N}\sum_{j=1}^{N} |W_{i,j}|$$

1. L_2 norm $$\mathcal{L} = \mathcal{L}_{MSE} + \sum_{i=1}^{N}\sum_{j=1}^{N} W_{i,j}^2$$

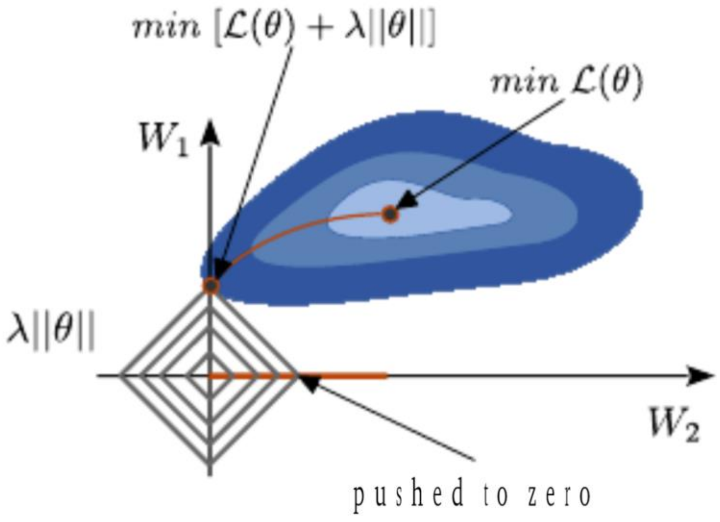
Regularization

- Regularization methods:

2. Norm penalties

1. L_1 norm

$$\mathcal{L} = \mathcal{L}_{MSE} + \sum_{i=1}^{N} \sum_{j=1}^{N} |W_{i,j}|$$



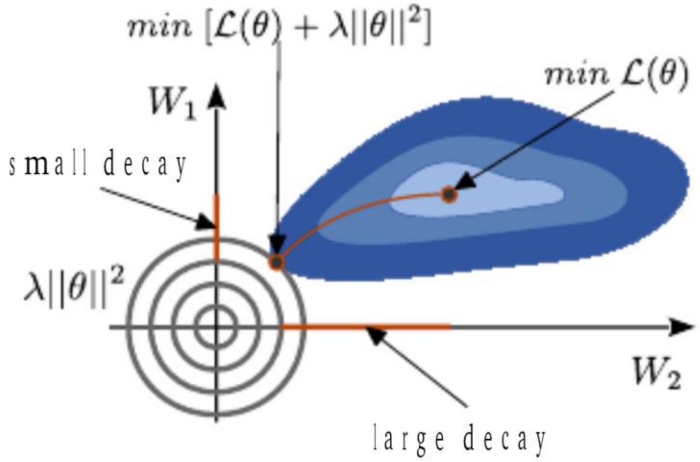
Regularization

- Regularization methods:

2. Norm penalties

1. L_2 norm

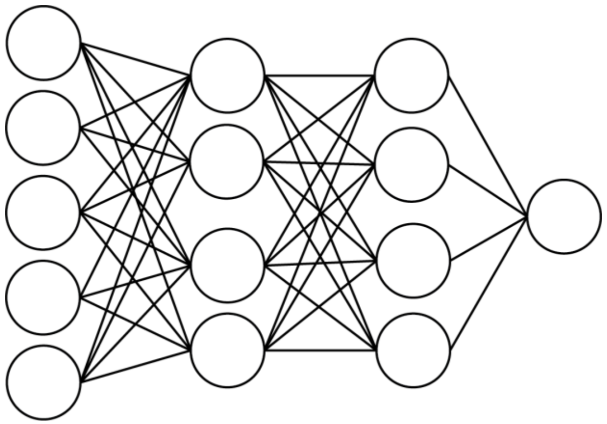
$$\mathcal{L} = \mathcal{L}_{MSE} + \sum_{i=1}^{N}\sum_{j=1}^{N} W_{i,j}^2$$

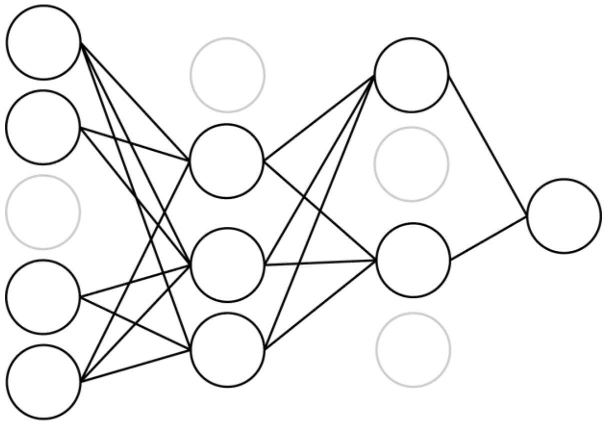


Regularization

- Regularization methods:

  3. Dropout
    - ➢ Individual nodes are temporarily-dropped at random during training
    - ➢ typical dropout rates: 0.2, …, 0.5
    - ➢ It forces the network to created different mappings
    - ➢ This method improves the network's stability
    - ➢ For predictions on new data, all nodes are used





- Example situation:
    - ➢ Sometimes network layers co-adapt to correct mistakes from prior layers
    - ➢ This can potentially be broken up with dropout

Regularization

- Regularization methods:

  4. Additional regularization methods

  - Collect more data!

  - **Data augmentation**:
    - Generate additional data by modifying the already existing. E.g. rotated images

  - **Noise injection**:
    - Alter input values during training using random noise – leads to stabilization

  - **Ensemble training**:
    - Train many different networks on the same data for the same problem and combine their predictions – increases reliability

Hyperparameters

Hyperparameters

- List of **hyperparameters** to be tuned and decisions to be made
    - Number of hidden layers
    - Numbers of nodes per layer
    - Activation functions
    - Network initialization
    - Type and strength of regularization
    - Size of the minibatches
    - Learning rate and learning strategy

Hyperparameters

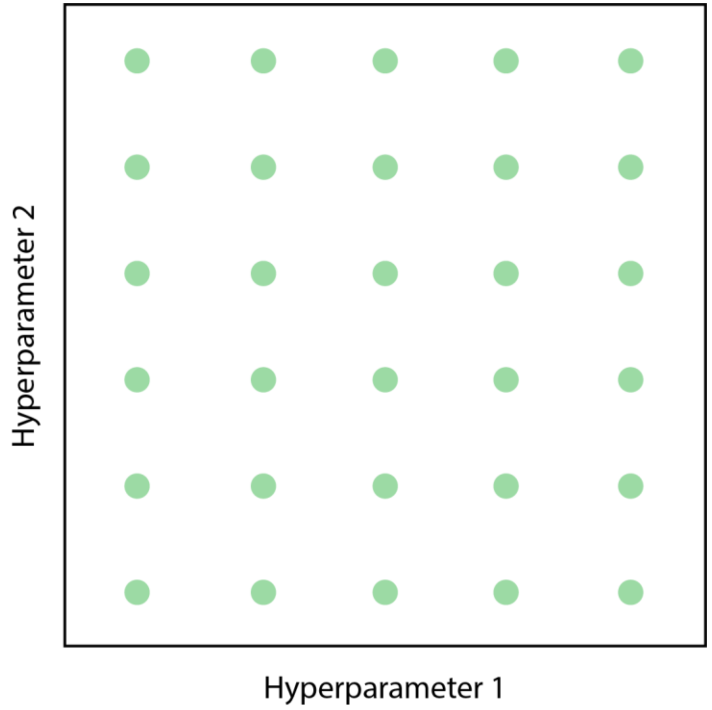
- Methods for **hyperparameter tuning**

1. **Grid search**

   ➢ Create a grid where each point corresponds to particular combinations of hyperparameter values

   ➢ Evaluate model performance for each point using the objective function values on the validation set

   ➢ Good method if the hyperparameter space is not too large



Hyperparameters

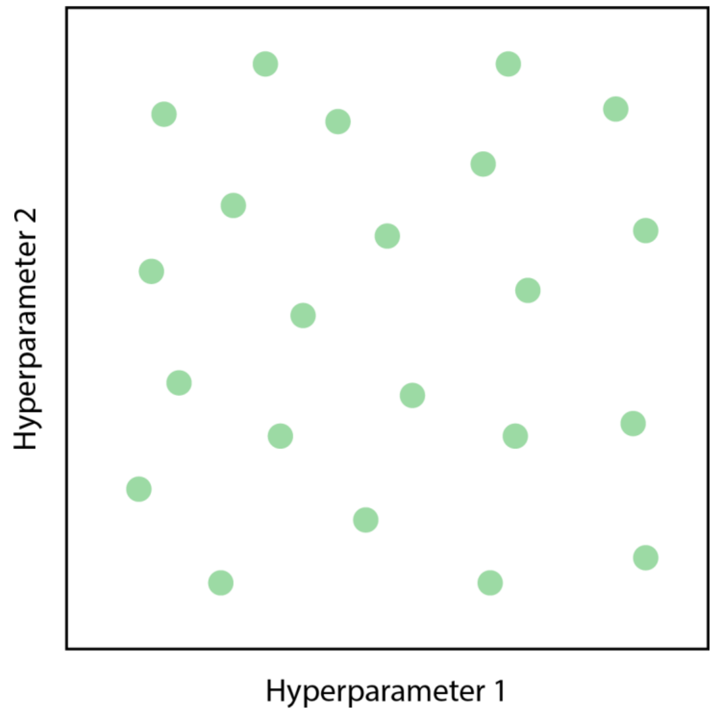
- Methods for **hyperparameter tuning**

  2. **Random search**

     - Instead of a grid, provide statistical distributions for each hyperparameter from which values are randomly sampled

     - More efficient than grid search

     - Works really well!

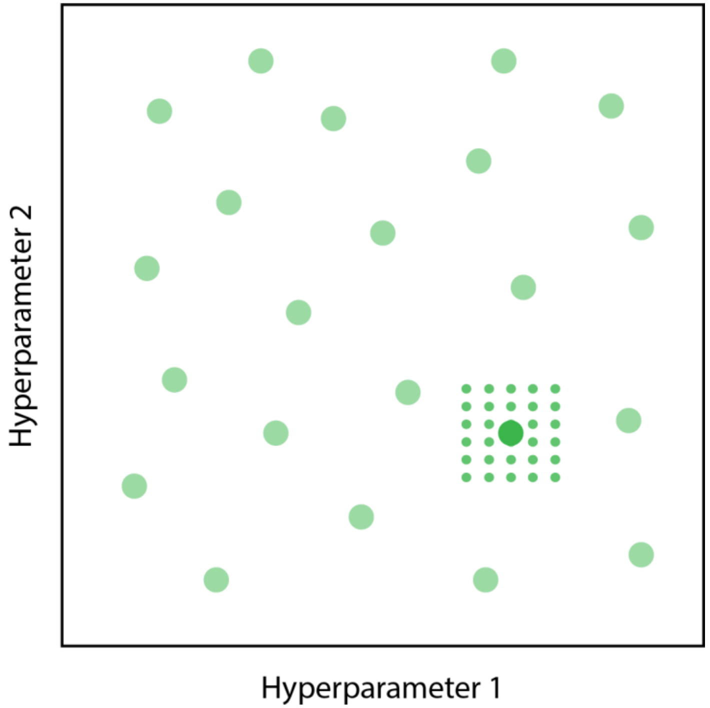


Hyperparameters

- Methods for **hyperparameter tuning**

3. **Coarse-to-fine search**

   - Random search followed by grid search
   - After finding the high-potential regions, investigate the area around it and fine tune the values
   - How many regions to investigate: depends on the time available



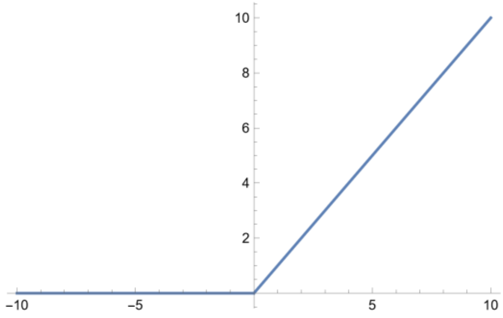
Hyperparameters

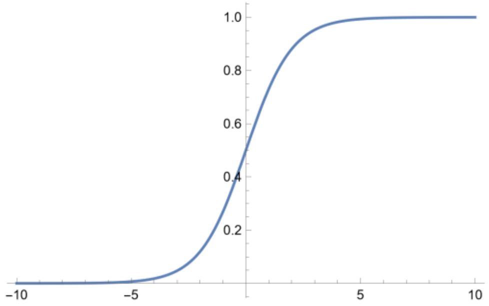
- Methods for **hyperparameter tuning**
  - Other methods
    - **Bayesian** techniques: Differ from grid or random searches in that they use information from past evaluations to choose the next hyperparameter values to evaluate. Bayesian methods can le
    - **Gradient-based** techniques…
    - **Evolutionary-optimization** techniques…
    - Other algorithmic methods...

Activation functions

Activation functions

- One of the most important decisions concerns the choice of the activation function
    - ➢ Good first choice: **ReLU** activation
    - ➢ **Leaky ReLU:** extension of ReLU with negative results also passed on
    - ➢ **Sigmoid** and **hyperbolic tangent (tanh)** function are rarely used in hidden layers because they can lead to vanishing gradient (training stops). They are used in the final output layer to constrain the result into [0,1] or [-1,1] respectively
- More detailed discussion on activation functions will presented in later lectures





Activation functions

a) ReLU

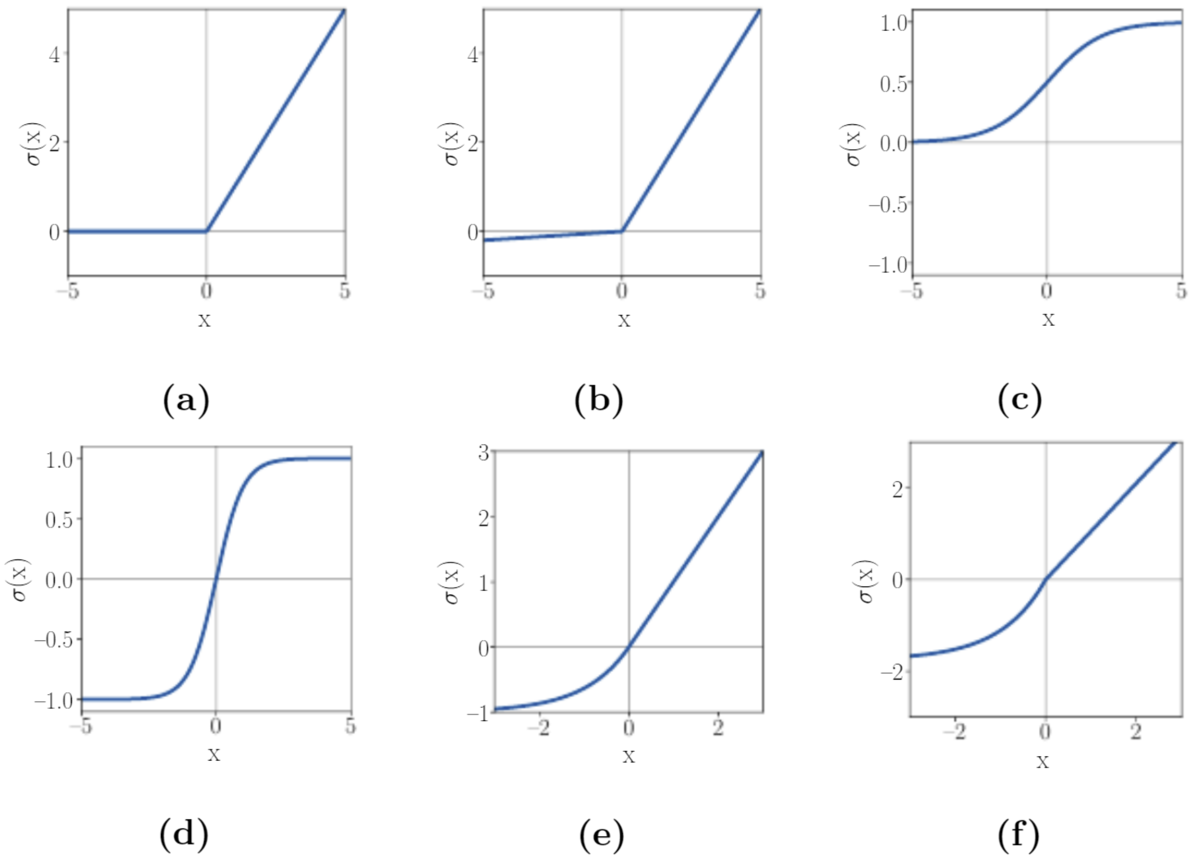
b) Leaky ReLU

c) Sigmoid

d) Tanh

e) ELU

f) SELU



Summary

- The entire data set is usually split into three sets: **train**, **validation** and **test** sets
- A network's ability to **generalize** means that it can provide correct predictions on data that were not used for training/optimizing it
- The **bias-variance tradeoff** shows that reducing bias on model's predictions comes with the expense of increasing it's variance and vice versa
- **Overfitting** is what happens when a network gives good predictions on the train set but fails to generalize, while **underfitting** is when the network fails to capture the complexity of the underline distribution that generates the data
- The training procedure is typically **monitored** using the behavior of the objective function as function of the number of epochs
- **Regularization** methods help reduce overfitting and stabilize network training
- **Hyperparameters** define the type of a network and how it is trained