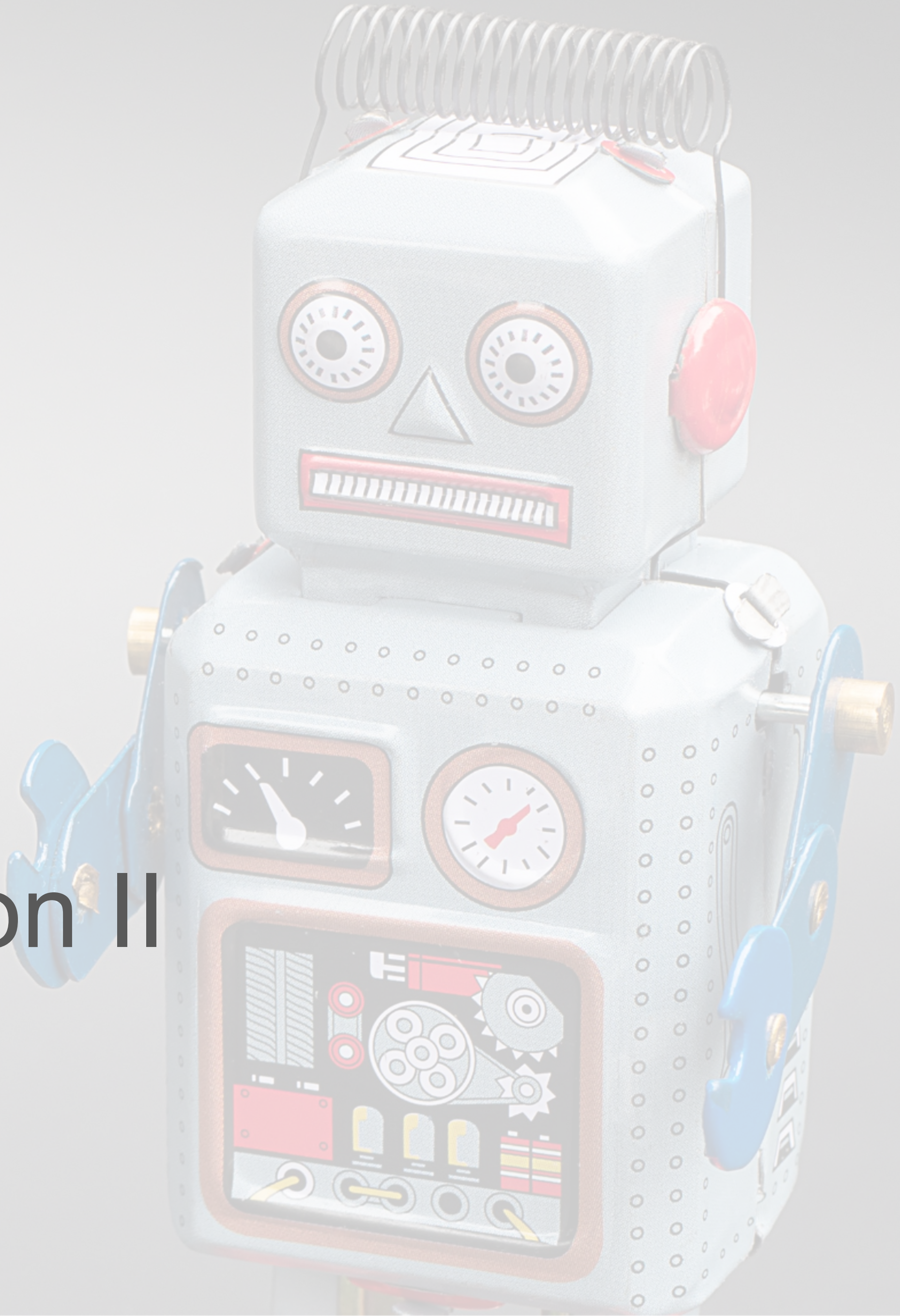
Support Session II



Deep Learning with Python
Shamsi Abdurakhmanova
Aalto University
3.11.22

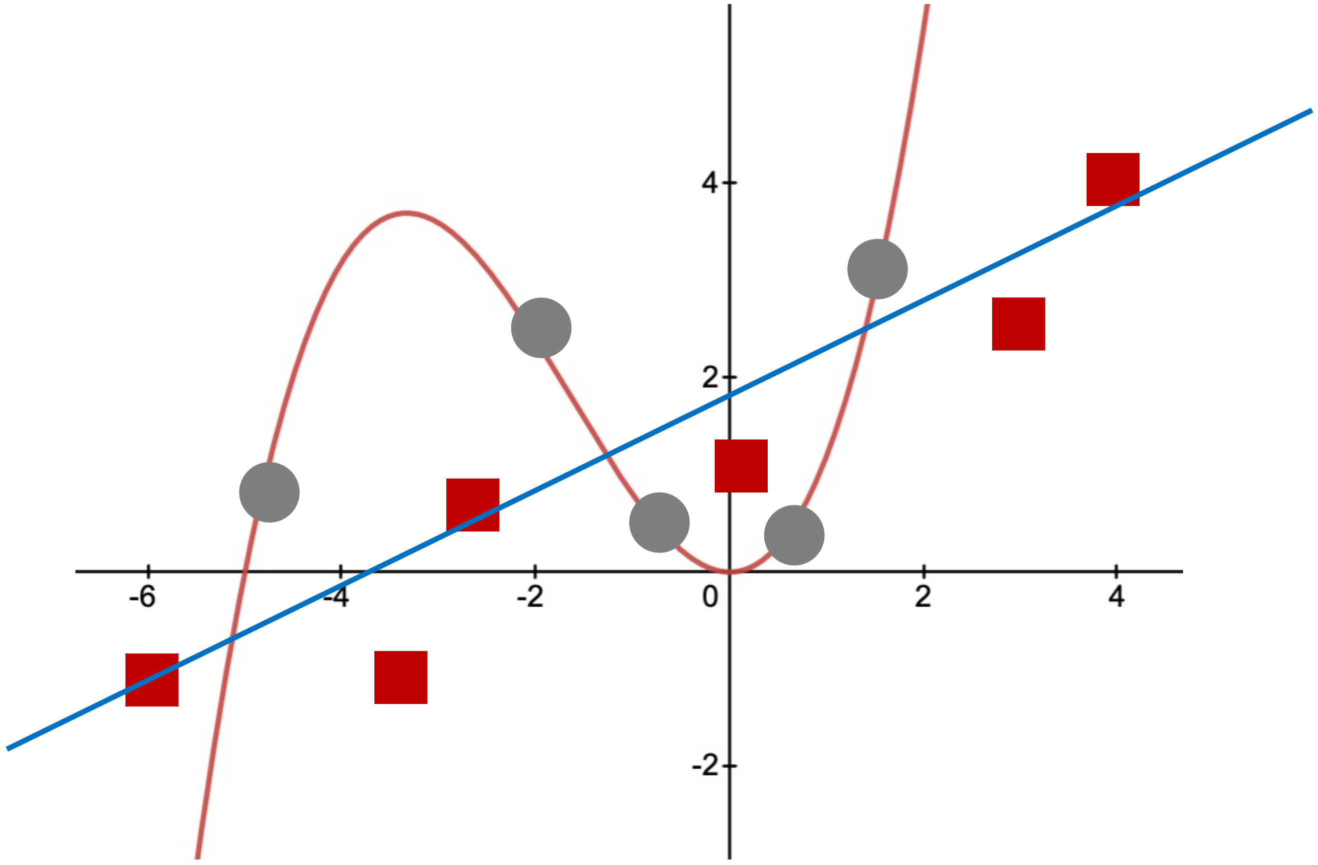
✓ Content

- Model selection and validation
- Python classes
- Implement ANN with np.arrays and Python classes

Model Selection & Validation

Overfitting

- Model performs well on training data, but much worse on new instances



Overfitting

- Model performs well on training data, but much worse on new instances

✓ Collect more high-quality data (low noise, representative)

✓ Simplify model:

- Choose "simpler" model with less parameters
- Add regularization to existing model

→ keep track of *n/m* ratio (n.o. features/ n.o. samples)

→ $n << m$

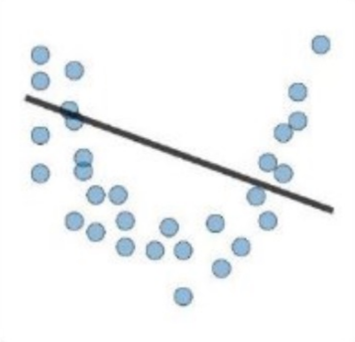
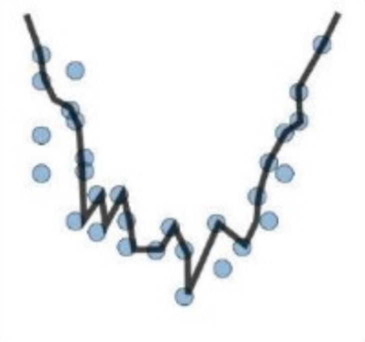
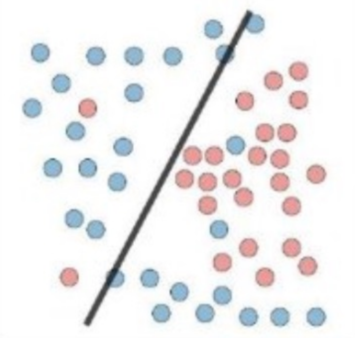
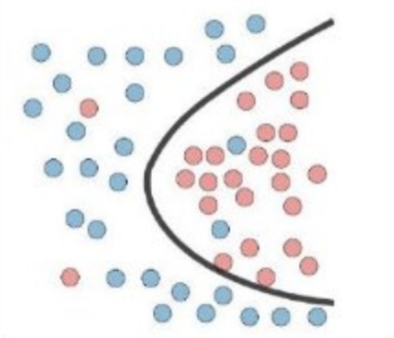
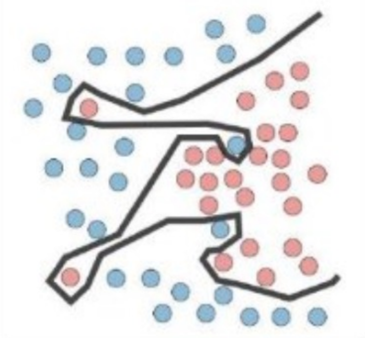
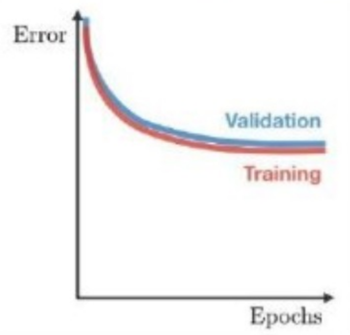
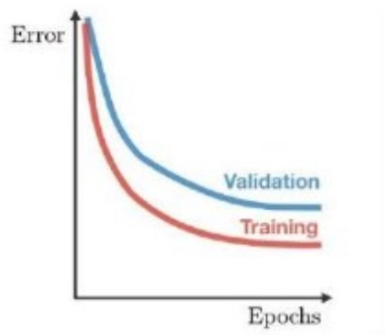
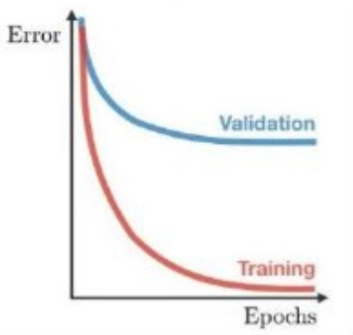
Underfitting

- Model is too simple to learn underlying structure of the data

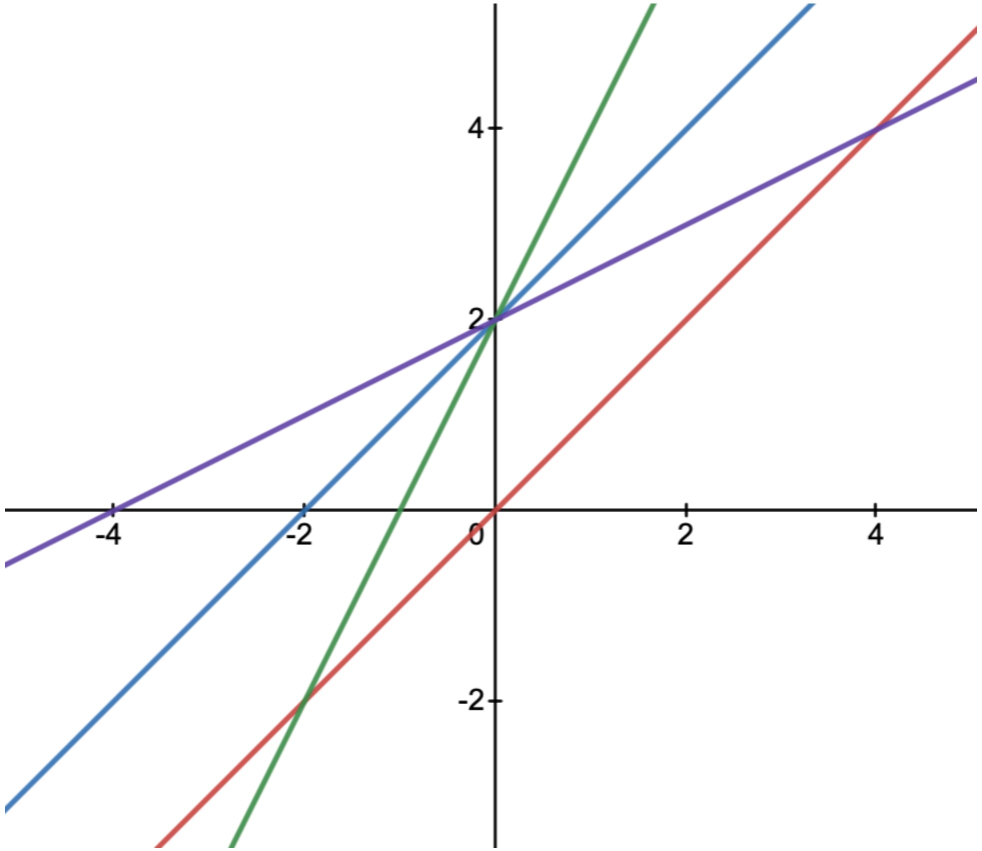
✓ Use more powerful model (more params)

✓ Reduce regularization

✓ Use better features (feature engineering)

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			
Deep learning			
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

Estimate performance of the trained model



- Hypothesis space:

All functions of type
$h(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

- Training:

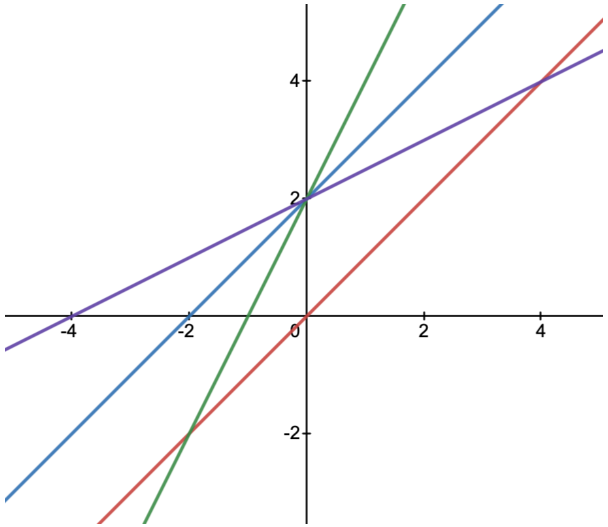
Choose hypothesis $h(\mathbf{x})$ with optimal parameters $\mathbf{w}^*$ (low training error)

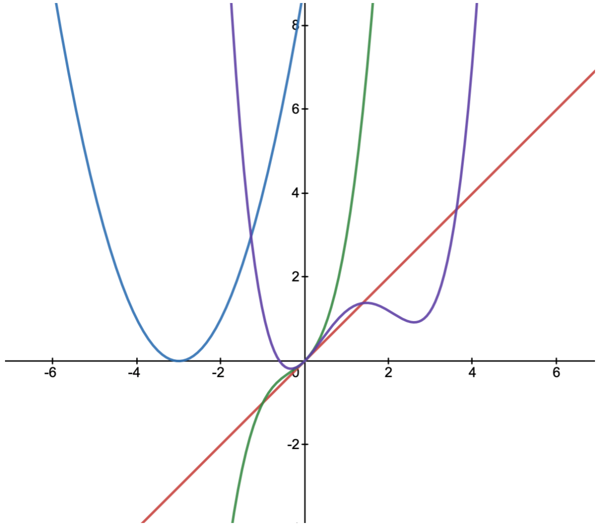
Estimate performance of the trained model

- Want to know how model will perform on new data (generalization property)

- ✓ Split dataset into ***training*** and ***test*** sets
- ✓ Train model only on training set
- ✓ Estimate generalization error on test set

Choose between two models





Hypothesis space:

All functions of type
$h(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

All functions of type
$h(\mathbf{x}) = w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + \cdots$

Training:

Choose hypothesis $h(\mathbf{x})$ with optimal parameters $\mathbf{w}^*$ (low training error)

Choose hypothesis $h(\mathbf{x})$ with optimal parameters $\mathbf{w}^*$ (low training error)

Choose between two models

- ✓ Split dataset into ***training*** and ***test*** sets
- ✓ Train both models on training set
- ✓ Choose the best performing model on training set
- ✓ Estimate generalization error on test set

→ increased chance that chosen model overfits training set

Choose between two models

- ✓ Split dataset into ***training*** and ***test*** sets
- ✓ Train both models on training set
- ✓ Choose the best performing model on test set
- ✓ Estimate generalization error on test set

→ increased chance that chosen model overfits test set

Generalization error must be estimated on “new” data, not used to train or choose a model!

- ✓ Split data into training, validation, test sets
- ✓ Training set – to tune model parameters (weights & biases)
- ✓ Validation set - to tune model ***hyperparameters***,
  model selection
- ✓ Test set – to estimate generalization error

- ***Parameters*** of a model – learnt during training
- ***Hyperparameters*** of a model – cannot be learnt during training; must be set before training

✓ Parameters of a model – weights and bias

✓ Hyperparameters of a model – amount of regularization, n.o.features, n.o. layers in ANN,…

Model selection

- Choose between several model (e.g. linear vs polynomial regression)

✓ Train both models on ***training*** set

✓ Choose model which perfoms best on ***validation*** set

✓ Estimate generalization error on ***test*** set

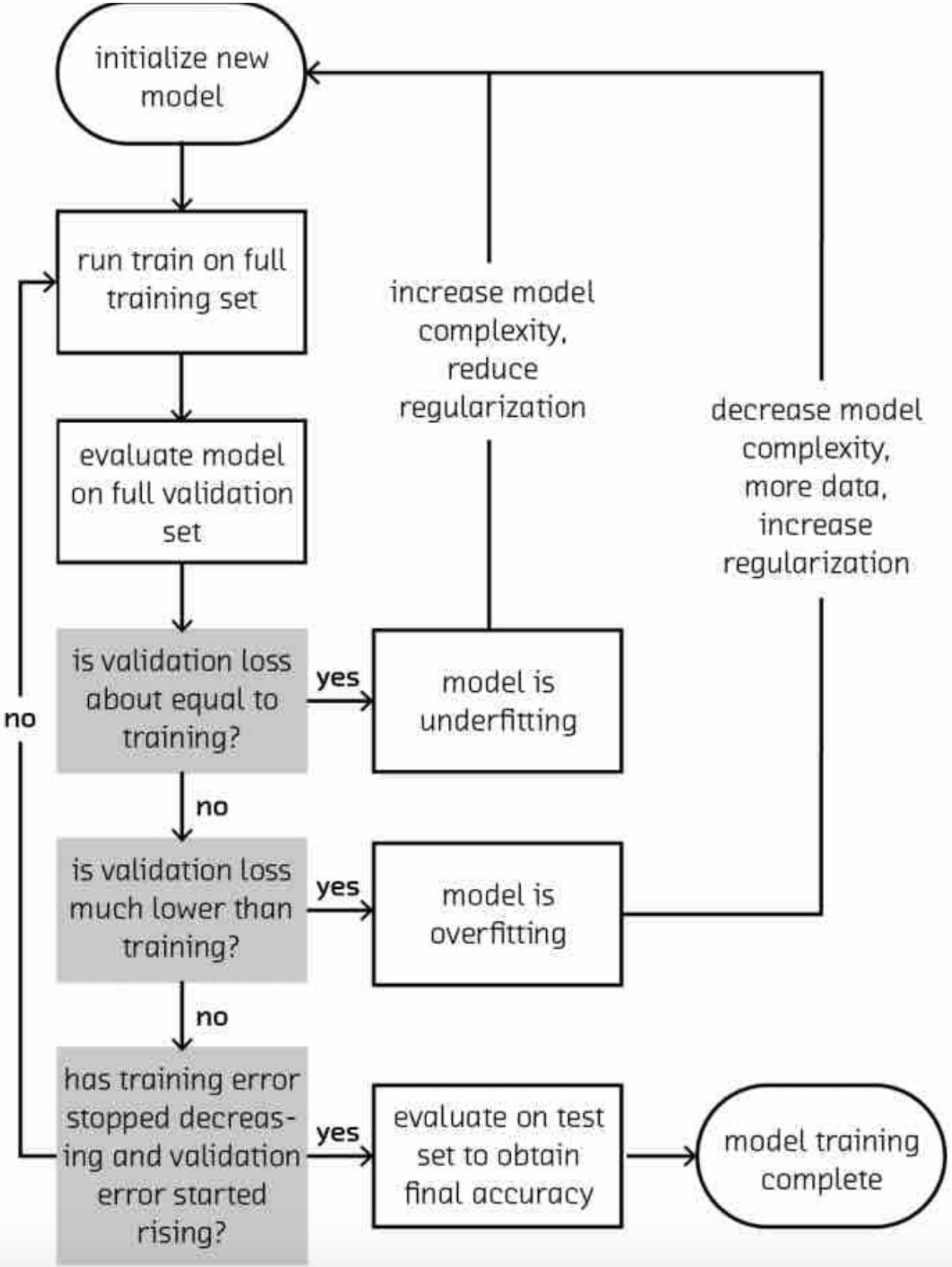
Hyperparameter tuning

- Choose between different values of regularization parameter (param C for logistic regression)

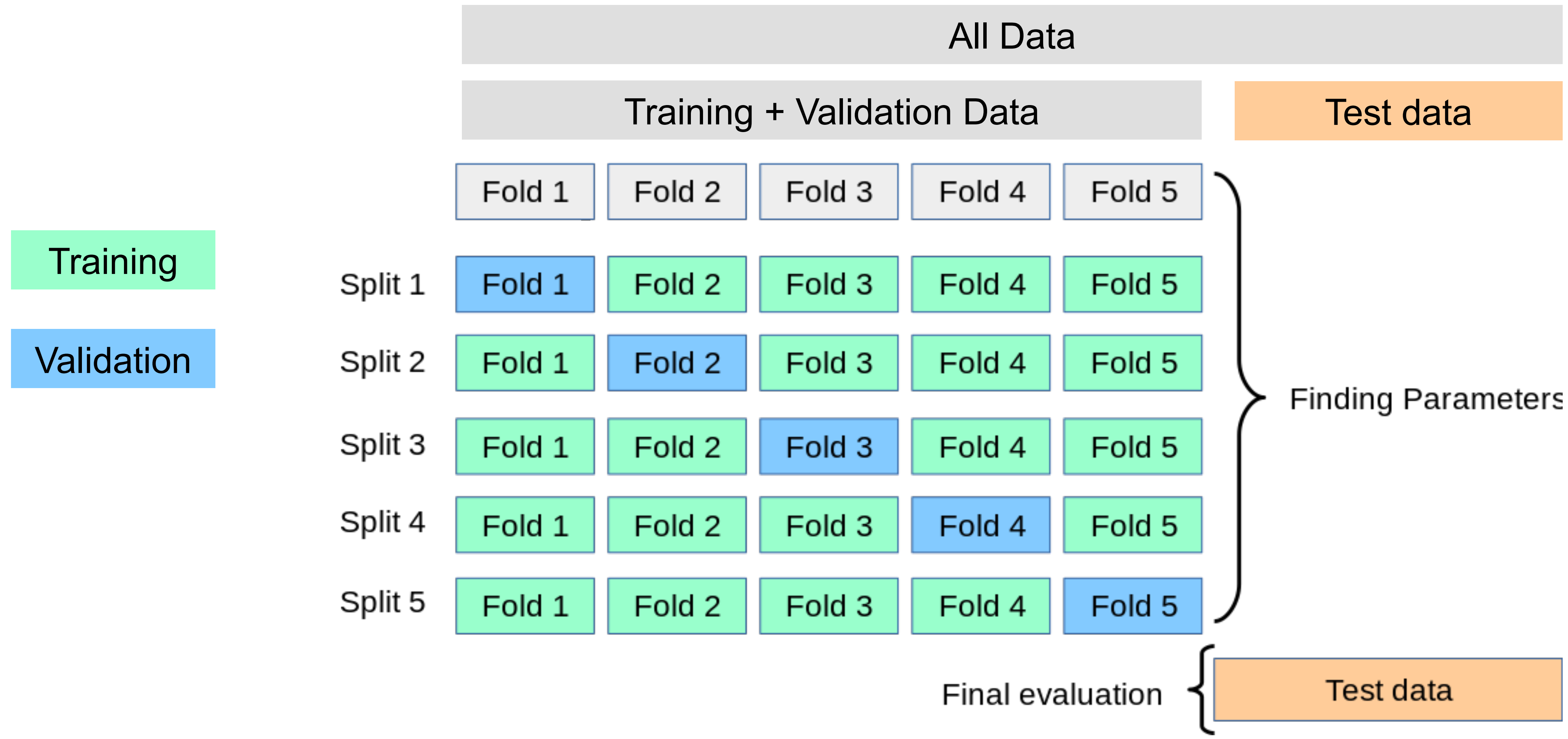
✓ Train several models with different values of param C on ***training*** set

✓ Choose param C values which perfoms best on ***validation*** set

✓ Estimate generalization error on ***test*** set


initialize new model
run train on full training set
evaluate model on full validation set
is validation loss about equal to training?
yes
model is underfitting
increase model complexity, reduce regularization
no
is validation loss much lower than training?
yes
model is overfitting
decrease model complexity, more data, increase regularization
no
has training error stopped decreas-ing and validation error started rising?
yes
evaluate on test set to obtain final accuracy
model training complete
no

5-Fold Cross-Validation



Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

Sebastian Raschka
University of Wisconsin–Madison
Department of Statistics
November 2018
sraschka@wisc.edu

Python classes

Python class objects

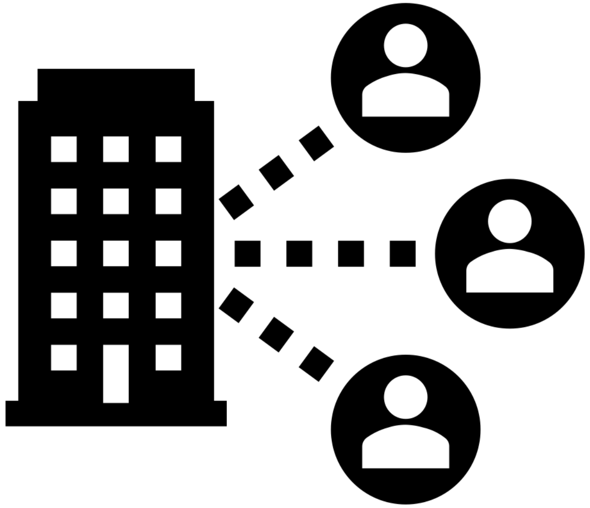
Classes provide a means of bundling data and functionality together. **Creating a new class creates a new *type* of object**, allowing new *instances* of that type to be made.

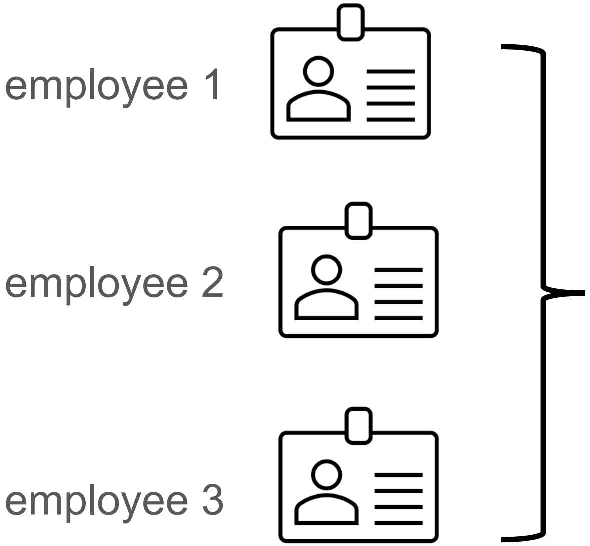
```python
import numpy as np

x = np.arange(5)
print(dir(x)[-20:])
print(type(x))
```

```
['searchsorted', 'setfield', 'setflags', 'shape', 'size', 'sort', 'squeeze', 'std', 'strides', 'sum', 'swapax
es', 'take', 'tobytes', 'tofile', 'tolist', 'tostring', 'trace', 'transpose', 'var', 'view']
<class 'numpy.ndarray'>
```

Python class objects

Company employees





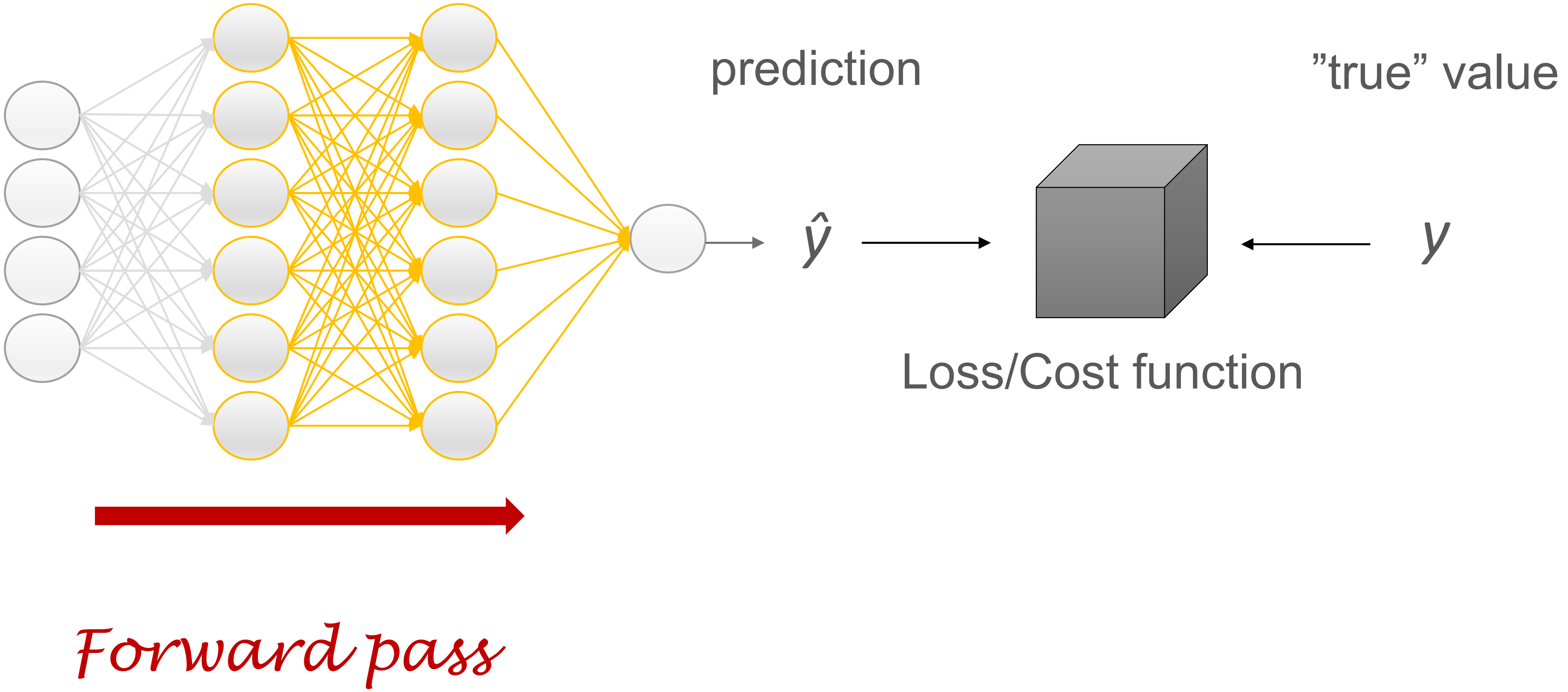
Attributes:

- first name
- last name
- pay
- email

Methods:

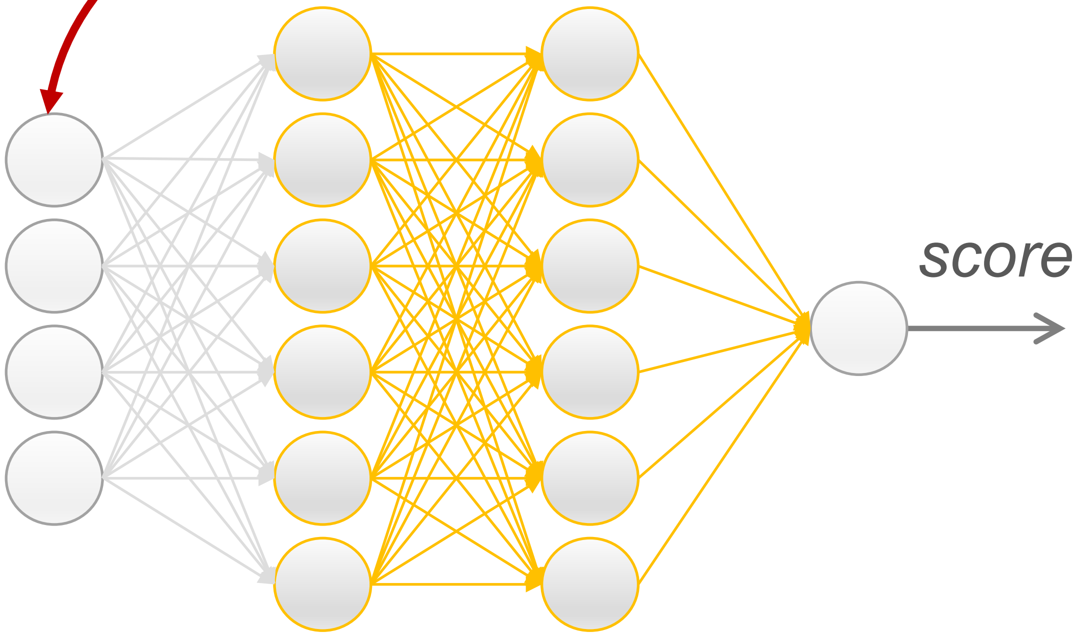
- get full name
- pay raise

Gradient Descent Algorithm



ANN – vectors and matrices

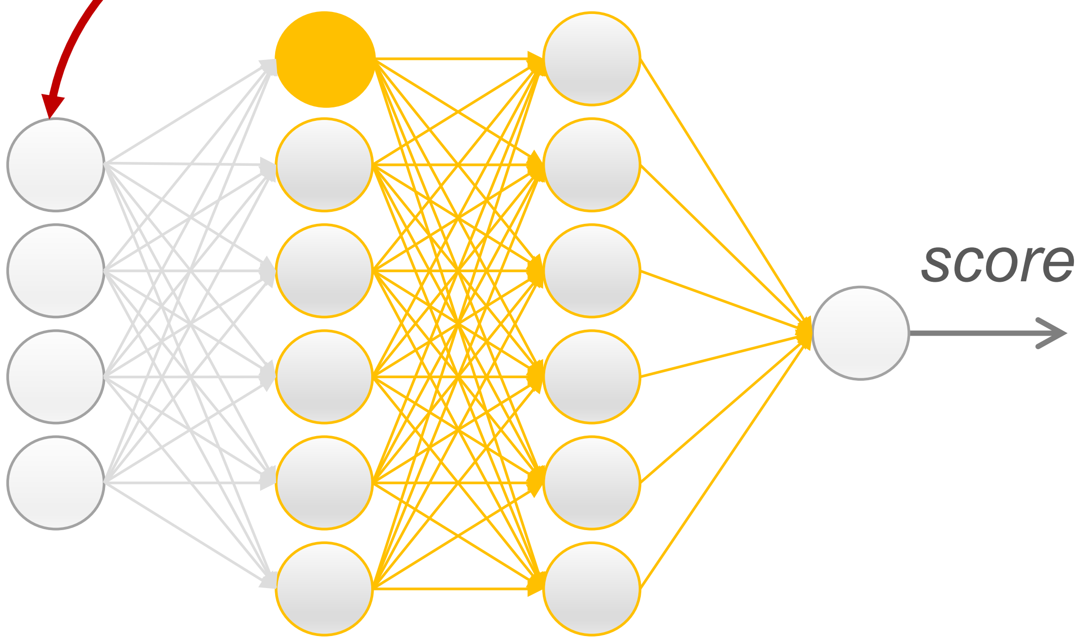
$\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})$ feature vector of ith datapoint



$\mathbf{w}_1^{(1)} = (w_{1,1}^{(1)}, w_{1,2}^{(1)}, w_{1,3}^{(1)}, w_{1,4}^{(1)})$ weight vector of 1st hidden neuron of 1st hidden layer

ANN – vectors and matrices

$$\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})$$



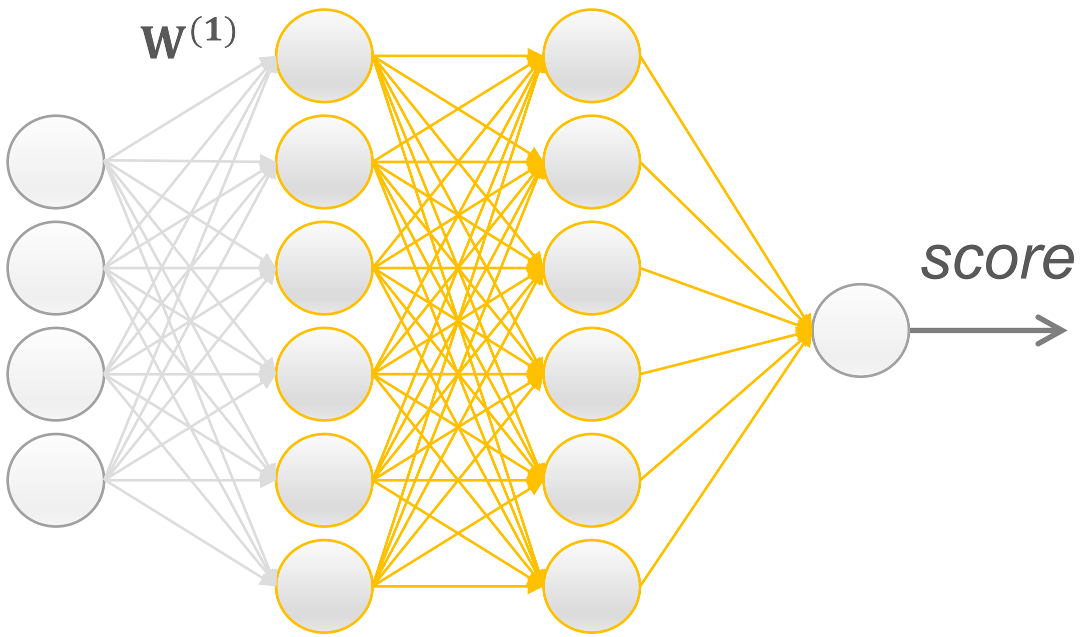
$$\mathbf{w}_1^{(1)} = (w_{1,1}^{(1)}, w_{1,2}^{(1)}, w_{1,3}^{(1)}, w_{1,4}^{(1)})$$

output of 1st hidden neuron of 1st hidden layer for ith datapoint

$$\sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right) = \mathbf{out}$$

(1,4) (4,1) (1,1)

ANN – vectors and matrices



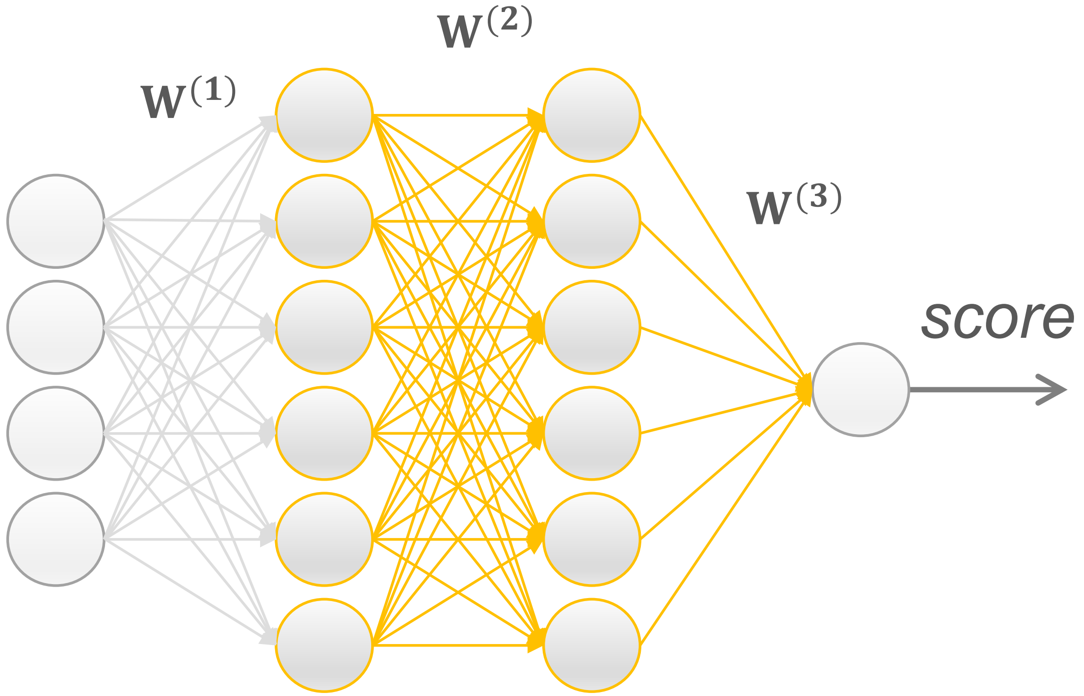
Feature matrix; shape (m,4)

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \dots & x_4^{(1)} \\ \dots & \dots & \dots \\ x_1^{(m)} & \dots & x_4^{(m)} \end{bmatrix}$$

Weight matrix of the 1st hidden layer; shape (4,6)

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & \dots & w_{6,1}^{(1)} \\ \dots & \dots & \dots \\ w_{1,4}^{(1)} & \dots & w_{6,4}^{(1)} \end{bmatrix}$$

ANN – vectors and matrices



output of 1st hidden layer for m datapoints

$$\sigma\left(\mathbf{X}\mathbf{W}^{(1)}\right) = \boldsymbol{h}^{(1)}$$

(m,4) (4,6) (m,6)

output of 2nd hidden layer for m datapoints

$$\sigma\left(\boldsymbol{h}^{(1)}\mathbf{W}^{(2)}\right) = \boldsymbol{h}^{(2)}$$

(m,6) (6,6) (m,6)

output score for m datapoints

$$\boldsymbol{h}^{(2)}\mathbf{W}^{(3)} = score$$

(m,6) (6,1) (m,1)