

CSI5180. Machine Learning for Bioinformatics Applications

Fundamentals of Machine Learning — Training

by

Marcel Turcotte

Preamble

Fundamentals of Machine Learning — Training

In this lecture, we introduce we focus on training learning algorithms. This will include the need for 2, 3 or k sets, tuning the hyperparameters values, as well as concepts such as under- and over-fitting the data.

General objective :

- ✦ **Describe** the fundamental concepts of machine learning

Learning objectives

- ❖ **Describe** the role of the **training**, **validation**, and **test** sets
- ❖ **Clarify** the concepts of **under-** and **over-** fitting the data
- ❖ **Explain** the process of tuning hyperparameters values

Reading:

- ❖ Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Mining* **10**:35 (2017).
- ❖ Boulesteix, A.-L. Ten simple rules for reducing overoptimistic reporting in methodological computational research. *PLoS Comput Biol* **11**:e1004191 (2015).
- ❖ Domingos, P. A few useful things to know about machine learning. *Commun Acm* **55**:7887 (2012).

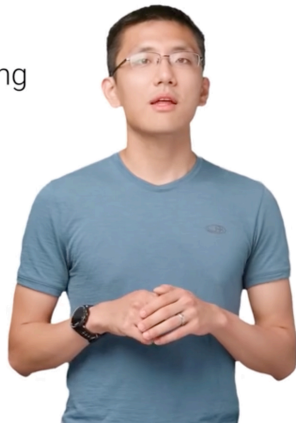
Plan

1. Preamble
2. Problem
3. Testing
4. Under- and over- fitting
5. 7-Steps workflow
6. Prologue

The 7 Steps of Machine Learning

7 Steps of Machine Learning

- Gathering Data
- Preparing that Data
- Choosing a Model
- Training
- Evaluation
- Hyperparameter Tuning
- Prediction



<https://youtu.be/nKW8Ndu7Mjw>

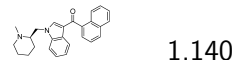
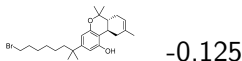
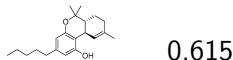
Problem

Supervised learning - regression

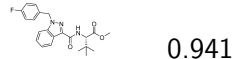
- ❖ The **data set** is a collection of **labelled** examples.
 - ❖ $\{(x_i, y_i)\}_{i=1}^N$
 - ❖ Each x_i is a **feature vector** with D dimensions.
 - ❖ $x_i^{(j)}$ is the value of the **feature** j of the example i , for $j \in 1 \dots D$ and $i \in 1 \dots N$.
 - ❖ The **label** y_i is a **real number**.
- ❖ **Problem:** given the data set as input, create a “**model**” that can be used to predict the value of y for an unseen x .

QSAR

- ❖ **QSAR** stands for **Quantitative Structure-Activity Relationship**
- ❖ As a machine learning problem,
 - ❖ Each x_i is a **chemical compound**
 - ❖ y_i is the **biological activity** of the compound x_i
 - ❖ Examples of biological activity include **toxicology** and **biodegradability**



...



HIV-1 reverse transcriptase inhibitors

- ❖ Viira, B., García-Sosa, A. T. & Maran, U. Chemical structure and correlation analysis of HIV-1 NNRT and NRT inhibitors and database-curated, published inhibition constants with chemical structure in diverse datasets. *J Mol Graph Model* **76**:205223 (2017).

HIV-1 reverse transcriptase inhibitors

- ❖ Viira, B., García-Sosa, A. T. & Maran, U. Chemical structure and correlation analysis of HIV-1 NNRT and NRT inhibitors and database-curated, published inhibition constants with chemical structure in diverse datasets. *J Mol Graph Model* **76**:205223 (2017).
- ❖ “Human immunodeficiency virus type 1 reverse transcriptase (HIV-1 RT) has been one of the most important targets for anti-HIV drug development due to it being an essential step in retroviral replication.”

HIV-1 reverse transcriptase inhibitors

- ❖ Viira, B., García-Sosa, A. T. & Maran, U. Chemical structure and correlation analysis of HIV-1 NNRT and NRT inhibitors and database-curated, published inhibition constants with chemical structure in diverse datasets. *J Mol Graph Model* **76**:205223 (2017).
- ❖ “Human immunodeficiency virus type 1 reverse transcriptase (HIV-1 RT) has been one of the most important targets for anti-HIV drug development due to it being an essential step in retroviral replication.”
- ❖ “Many small molecule compounds (. . .) have been studied over the years.”

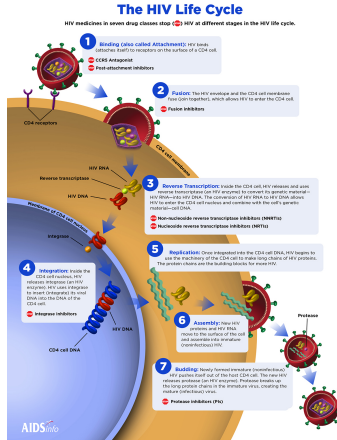
HIV-1 reverse transcriptase inhibitors

- ❖ Viira, B., García-Sosa, A. T. & Maran, U. Chemical structure and correlation analysis of HIV-1 NNRT and NRT inhibitors and database-curated, published inhibition constants with chemical structure in diverse datasets. *J Mol Graph Model* **76**:205223 (2017).
- ❖ “Human immunodeficiency virus type 1 reverse transcriptase (HIV-1 RT) has been one of the most important targets for anti-HIV drug development due to it being an essential step in retroviral replication.”
- ❖ “Many small molecule compounds (. . .) have been studied over the years.”
- ❖ “Due to mutations and other influencing factors, the search for new inhibitor molecules for HIV-1 is ongoing.”

HIV-1 reverse transcriptase inhibitors

- ❖ Viira, B., García-Sosa, A. T. & Maran, U. Chemical structure and correlation analysis of HIV-1 NNRT and NRT inhibitors and database-curated, published inhibition constants with chemical structure in diverse datasets. *J Mol Graph Model* **76**:205223 (2017).
- ❖ “Human immunodeficiency virus type 1 reverse transcriptase (HIV-1 RT) has been one of the most important targets for anti-HIV drug development due to it being an essential step in retroviral replication.”
- ❖ “Many small molecule compounds (. . .) have been studied over the years.”
- ❖ “Due to mutations and other influencing factors, the search for new inhibitor molecules for HIV-1 is ongoing.”
- ❖ “Our recent design, modelling, and synthesis effort in the search for new compounds has resulted in two new, small, low toxicity (. . .) inhibitors.”

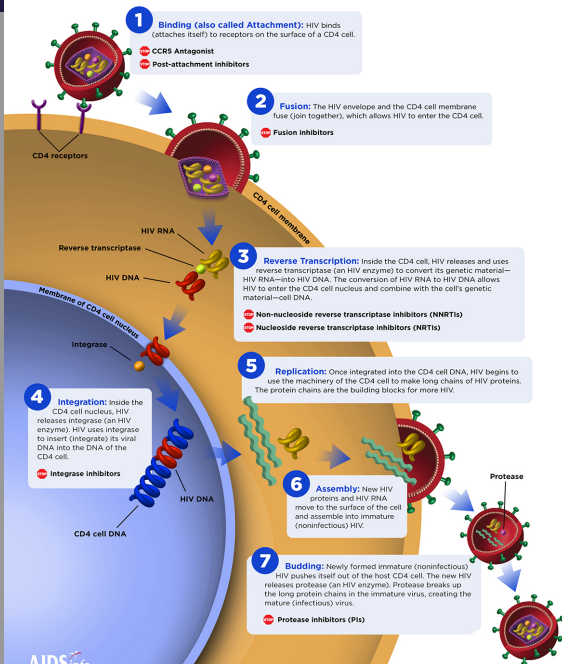
HIV Life Cycle



<https://aidsinfo.nih.gov/understanding-hiv-aids>

The HIV Life Cycle

HIV medicines in seven drug classes stop (🛑) HIV at different stages in the HIV life cycle.



HIV-1 reverse transcriptase inhibitors

- ✚ Each compound (**example**) in **ChemDB** has features such as the **number of atoms, area, solvation, coulombic, molecular weight, XLogP**, etc.

HIV-1 reverse transcriptase inhibitors

- ❖ Each compound (**example**) in **ChemDB** has features such as the **number of atoms, area, solvation, coulombic, molecular weight, XLogP**, etc.
- ❖ A possible solution, a model, would look something like this:

$$\hat{y} = 44.418 - 35.133 \times x^{(1)} - 13.518 \times x^{(2)} + 0.766 \times x^{(3)}$$

Testing

Two sets!

- ✦ **Training set** versus **test set**

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing
- ❖ **Training set:** the data set used for **training** your model

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing
- ❖ **Training set:** the data set used for **training** your model
- ❖ **Test set:** an **independent** set that used at the very end to evaluate the performance of your model

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing
- ❖ **Training set:** the data set used for **training** your model
- ❖ **Test set:** an **independent** set that used at the very end to evaluate the performance of your model
 - ❖ **Generalization error:** error rate on new cases

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing
- ❖ **Training set:** the data set used for **training** your model
- ❖ **Test set:** an **independent** set that used at the very end to evaluate the performance of your model
 - ❖ **Generalization error:** error rate on new cases
- ❖ In most cases, the **training error** will be **low**, this because most learning algorithms are designed to find a set of values for their (weights) parameters such that the training error is low. However, the **generalization error** can still be **high**, we say that **the model is overfitting the training data**.

Two sets!

- ❖ **Training set** versus **test set**
- ❖ **Rule of thumb:** keep **80 %** of your data for **training** and use the remaining **20 %** of your data for testing
- ❖ **Training set:** the data set used for **training** your model
- ❖ **Test set:** an **independent** set that used at the very end to evaluate the performance of your model
 - ❖ **Generalization error:** error rate on new cases
- ❖ In most cases, the **training error** will be **low**, this because most learning algorithms are designed to find a set of values for their (weights) parameters such that the training error is low. However, the **generalization error** can still be **high**, we say that **the model is overfitting the training data**.
- ❖ If the **training error** is high, we say that **the model is underfitting the training data**.

Under- and over- fitting

Underfitting and overfitting

- ❖ **Underfitting** and **overfitting** are two important concepts for machine learning projects
- ❖ We will use a **regression** task to illustrate those two concepts

Linear Regression

- ❖ A **linear model** assumes that the value of the label, \hat{y}_i , can be expressed as a **linear combination** of the feature values, $x_i^{(j)}$:

$$\hat{y}_i = h(x_i) = \theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \dots + \theta_D x_i^{(D)}$$

Linear Regression

- ❖ A **linear model** assumes that the value of the label, \hat{y}_i , can be expressed as a **linear combination** of the feature values, $x_i^{(j)}$:

$$\hat{y}_i = h(x_i) = \theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \dots + \theta_D x_i^{(D)}$$

- ❖ Here, θ_j is the j th parameter of the (linear) **model**, with θ_0 being the **bias** term/parameter, $\theta_1 \dots \theta_D$ being the **feature weights**.

Linear Regression

- ❖ A **linear model** assumes that the value of the label, \hat{y}_i , can be expressed as a **linear combination** of the feature values, $x_i^{(j)}$:

$$\hat{y}_i = h(x_i) = \theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \dots + \theta_D x_i^{(D)}$$

- ❖ Here, θ_j is the j th parameter of the (linear) **model**, with θ_0 being the **bias** term/parameter, $\theta_1 \dots \theta_D$ being the **feature weights**.
- ❖ **Problem:** find values for all the model parameters so that the model “**best fit**” the training data.

Linear Regression

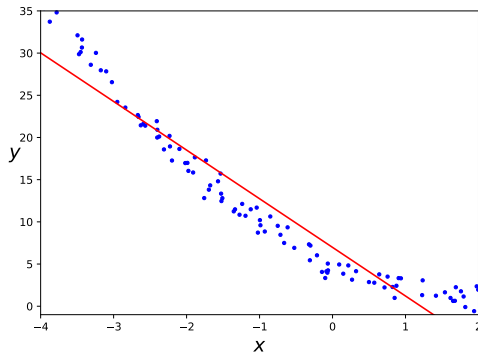
- ❖ A **linear model** assumes that the value of the label, \hat{y}_i , can be expressed as a **linear combination** of the feature values, $x_i^{(j)}$:

$$\hat{y}_i = h(x_i) = \theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \dots + \theta_D x_i^{(D)}$$

- ❖ Here, θ_j is the j th parameter of the (linear) **model**, with θ_0 being the **bias** term/parameter, $\theta_1 \dots \theta_D$ being the **feature weights**.
- ❖ **Problem:** find values for all the model parameters so that the model “**best fit**” the training data.
 - ❖ The **Root Mean Square Error** is a common performance measure for regression problems.

$$\sqrt{\frac{1}{N} \sum_1^N [h(x_i) - y_i]^2}$$

sklearn.linear_model.LinearRegression



```
from sklearn.linear_model import LinearRegression  
  
lin_reg = LinearRegression()  
lin_reg.fit(X, y)
```

Source code

```
import numpy as np

X = 6 * np.random.rand(100, 1) - 4
y = X ** 2 - 4 * X + 5 + np.random.randn(100, 1)

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X, y)

X_new = np.array([[ -4], [ 2]])
y_pred = lin_reg.predict(X_new)
```

Source code (continued)

```
plt.plot(X, y, "b.")
plt.plot(X_new, y_pred, "r-")
plt.xlabel("$x$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([-4, 2, -1, 35])
save_fig("regression_linear-01")
plt.show()
```

Source code (continued)

```
import os
import matplotlib as mpl
import matplotlib.pyplot as plt

def save_fig(fig_id, tight_layout=True, fig_extension="pdf", resolution=300):
    path = os.path.join(fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

The need for a validation set

- ❖ **Wait a minute!** How do we know that a **linear model** is suitable for this application?

The need for a validation set

- ❖ **Wait a minute!** How do we know that a **linear model** is suitable for this application?
- ❖ **We don't!**

The need for a validation set

- ❖ **Wait a minute!** How do we know that a **linear model** is suitable for this application?
- ❖ **We don't!**
- ❖ **Solution:** we might want to “test” alternative hypotheses.

Three sets!

- ❖ **Hyperparameter:** a parameter whose value is not learnt by the algorithm, but set by the user.

Three sets!

- ❖ **Hyperparameter**: a parameter whose value is not learnt by the algorithm, but set by the user.
 - ❖ Examples include **learning rate**, **soft-margin** and **hard-margin** for SVM, **regularization weight** for regression, **number of layers** and **optimization algorithm** for ANN, but also **the degree of a polynomial** in the case of a regression, and many more.

Three sets!

- ❖ **Hyperparameter:** a parameter whose value is not learnt by the algorithm, but set by the user.
 - ❖ Examples include **learning rate**, **soft-margin** and **hard-margin** for SVM, **regularization weight** for regression, **number of layers** and **optimization algorithm** for ANN, but also **the degree of a polynomial** in the case of a regression, and many more.
- ❖ **Validation set:** a third data set is used to determine the “optimal” values of the parameters.

Three sets!

- ❖ **Hyperparameter:** a parameter whose value is not learnt by the algorithm, but set by the user.
 - ❖ Examples include **learning rate**, **soft-margin** and **hard-margin** for SVM, **regularization weight** for regression, **number of layers** and **optimization algorithm** for ANN, but also **the degree of a polynomial** in the case of a regression, and many more.
- ❖ **Validation set:** a third data set is used to determine the “optimal” values of the parameters.
- ❖ **Rule of thumb:** keep **70 %** of your data for **training**, **15 %** of your data is used for **validation**, and **15 %** of your data is used for testing.

Three sets!

- ❖ **Hyperparameter:** a parameter whose value is not learnt by the algorithm, but set by the user.
 - ❖ Examples include **learning rate**, **soft-margin** and **hard-margin** for SVM, **regularization weight** for regression, **number of layers** and **optimization algorithm** for ANN, but also **the degree of a polynomial** in the case of a regression, and many more.
- ❖ **Validation set:** a third data set is used to determine the “optimal” values of the parameters.
- ❖ **Rule of thumb:** keep **70 %** of your data for **training**, **15 %** of your data is used for **validation**, and **15 %** of your data is used for testing.
- ❖ For data sets comprising millions of examples, **1** or **2 %** test set might be enough.

Learning curves

- ❖ One way to **assess our models** is to visualize the **learning curves**:

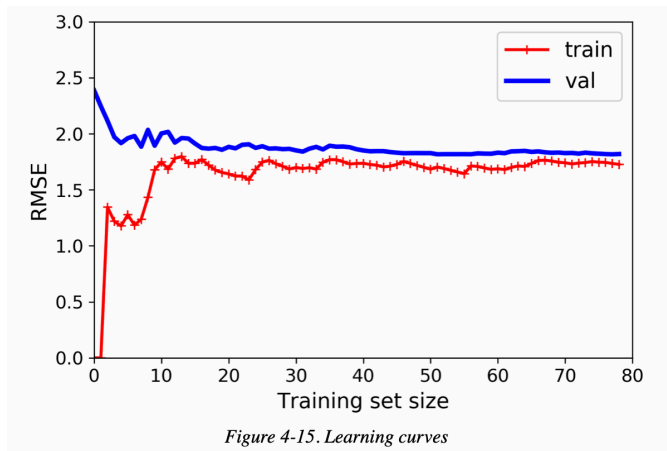
Learning curves

- ❖ One way to **assess our models** is to visualize the **learning curves**:
 - ❖ A **learning curve** shows the performance of our model, here using RMSE, on both the **training set** and the **validation set**.

Learning curves

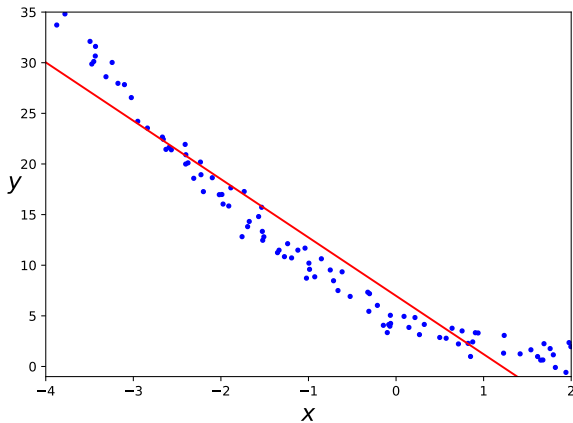
- ❖ One way to **assess our models** is to visualize the **learning curves**:
 - ❖ A **learning curve** shows the performance of our model, here using RMSE, on both the **training set** and the **validation set**.
 - ❖ Multiple **measurements** are obtained by repeatedly training the model on larger and larger subsets of the data.

Linear model - underfitting



Source: Géron 2019

sklearn.linear_model.LinearRegression



Learning curves - underfitting

- ✦ With only one (1) or two (2) examples, the model perfectly “fits” the **training set**.

Learning curves - underfitting

- ❖ With only one (1) or two (2) examples, the model perfectly “fits” the **training set**.
- ❖ As the size of the data set grows, it becomes impossible to fit the training set since the randomly generated examples used to produce this example were generated using a quadratic function. Accordingly the RMSE reaches a plateau and stays there.

Learning curves - underfitting

- ❖ With only one (1) or two (2) examples, the model perfectly “fits” the **training set**.
- ❖ As the size of the data set grows, it becomes impossible to fit the training set since the randomly generated examples used to produce this example were generated using a quadratic function. Accordingly the RMSE reaches a plateau and stays there.
- ❖ With few examples, the model performs badly on the **validation set**. With few examples, it model generalizes poorly.

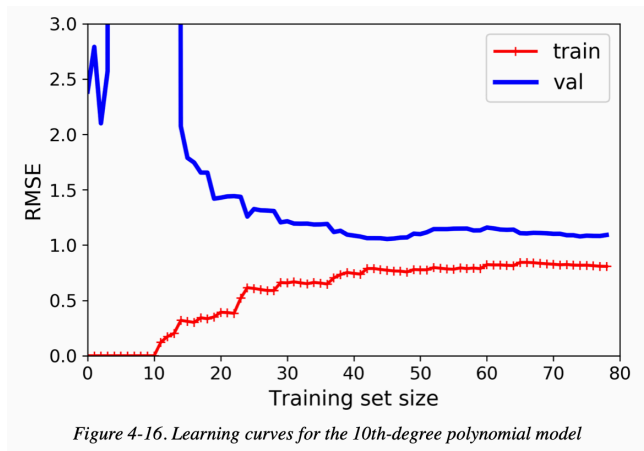
Learning curves - underfitting

- ❖ With only one (1) or two (2) examples, the model perfectly “fits” the **training set**.
- ❖ As the size of the data set grows, it becomes impossible to fit the training set since the randomly generated examples used to produce this example were generated using a quadratic function. Accordingly the RMSE reaches a plateau and stays there.
- ❖ With few examples, the model performs badly on the **validation set**. With few examples, it model generalizes poorly.
- ❖ As the size of the data set grows, the performance on the **validation set** improves. Eventually, the performance does not improve.

Learning curves - underfitting

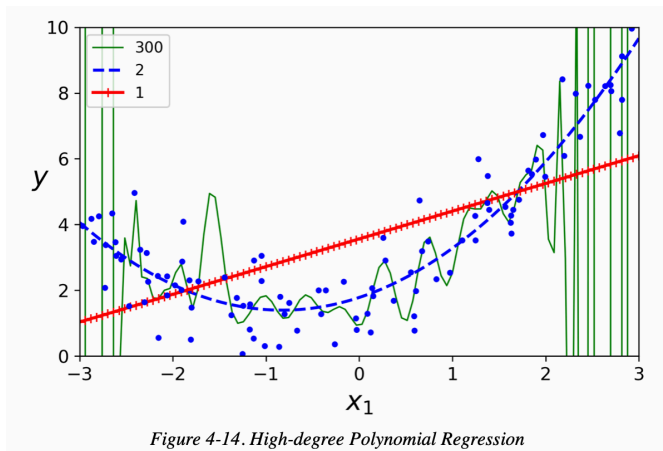
- ❖ With only one (1) or two (2) examples, the model perfectly “fits” the **training set**.
- ❖ As the size of the data set grows, it becomes impossible to fit the training set since the randomly generated examples used to produce this example were generated using a quadratic function. Accordingly the RMSE reaches a plateau and stays there.
- ❖ With few examples, the model performs badly on the **validation set**. With few examples, it model generalizes poorly.
- ❖ As the size of the data set grows, the performance on the **validation set** improves. Eventually, the performance does not improve.
- ❖ “These learning curves are typical of a model that's underfitting. Both curves have reached a plateau; they are close and fairly high.” [2]

Polynomial of degree 10 - overfitting



Source: Géron 2019

Overfitting and underfitting



Source: Géron 2019

Learning curves - overfitting

- ❖ Here, the error on the **training data** is much lower.

Learning curves - overfitting

- ❖ Here, the error on the **training data** is much lower.
- ❖ There is a gap between the two curves, the model performs significantly better on the training data compared to the validation data.

Overfitting - deep nets - loss

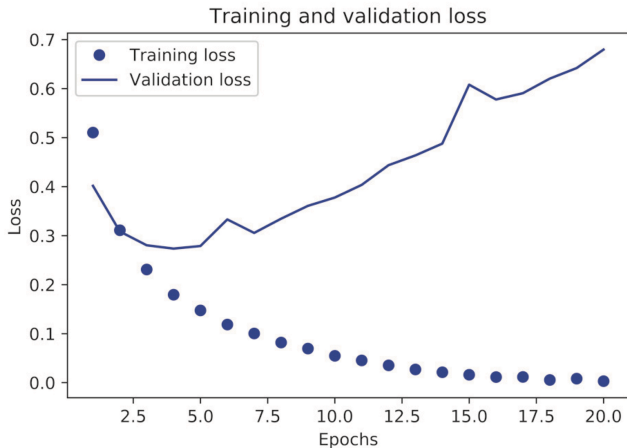


Figure 3.7 Training and validation loss

Source: Chollet 2018

Overfitting - deep nets - accuracy

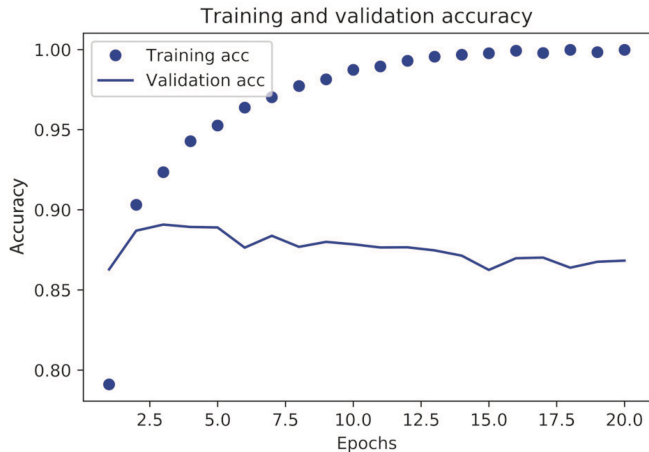


Figure 3.8 Training and validation accuracy

Source: Chollet 2018

Summary

❖ Underfitting:

- ❖ Your model is **too simple** (linear model).
- ❖ Uninformative features.
- ❖ **Poor** performance on both **training** and **validation** data.

❖ Overfitting:

- ❖ Your model is **too complex** (tall decision tree, deep and wide neural networks. . .).
- ❖ **Too many features** given the number of examples available
- ❖ **Excellent** performance on the **training set**, but **poor** performance on the **validation set**.

k sets! Cross-validation

What if I have a small number of examples?

- ✚ The data is **divided into k sets**.

k sets! Cross-validation

What if I have a small number of examples?

- ✚ The data is **divided into k sets**.
- ✚ Each set is called a **fold**;

k sets! Cross-validation

What if I have a small number of examples?

- ✚ The data is **divided into k sets**.
 - ✚ Each set is called a **fold**;
 - ✚ We talk about **3-fold, 5-fold, 10-fold** cross-validation;

k sets! Cross-validation

What if I have a small number of examples?

- ✚ The data is **divided into k sets**.
 - ✚ Each set is called a **fold**;
 - ✚ We talk about **3-fold**, **5-fold**, **10-fold** cross-validation;
 - ✚ The special case where $k = N$ is called **leave-one-out**.

k sets! Cross-validation

What if I have a small number of examples?

- ❖ The data is **divided into k sets**.
 - ❖ Each set is called a **fold**;
 - ❖ We talk about **3-fold**, **5-fold**, **10-fold** cross-validation;
 - ❖ The special case where $k = N$ is called **leave-one-out**.
- ❖ The training/validation procedure is **ran k times**.

k sets! Cross-validation

What if I have a small number of examples?

- ❖ The data is **divided into k sets**.
 - ❖ Each set is called a **fold**;
 - ❖ We talk about **3-fold**, **5-fold**, **10-fold** cross-validation;
 - ❖ The special case where $k = N$ is called **leave-one-out**.
- ❖ The training/validation procedure is **ran k times**.
 - ❖ Each time, one of the k sets is used for validation, whereas the rest of the data is used for training.

k sets! Cross-validation

What if I have a small number of examples?

- ❖ The data is **divided into k sets**.
 - ❖ Each set is called a **fold**;
 - ❖ We talk about **3-fold**, **5-fold**, **10-fold** cross-validation;
 - ❖ The special case where $k = N$ is called **leave-one-out**.
- ❖ The training/validation procedure is **ran k times**.
 - ❖ Each time, one of the k sets is used for validation, whereas the rest of the data is used for training.
- ❖ At the end, one calculates the **mean** and **standard deviation** for the metrics of interest: cost/loss function, precision/recall, etc.

k sets! Cross-validation

What if I have a small number of examples?

- ❖ The data is **divided into k sets**.
 - ❖ Each set is called a **fold**;
 - ❖ We talk about **3-fold**, **5-fold**, **10-fold** cross-validation;
 - ❖ The special case where $k = N$ is called **leave-one-out**.
- ❖ The training/validation procedure is **ran k times**.
 - ❖ Each time, one of the k sets is used for validation, whereas the rest of the data is used for training.
- ❖ At the end, one calculates the **mean** and **standard deviation** for the metrics of interest: cost/loss function, precision/recall, etc.
- ❖ Open the door to **hypothesis testing**.

TO DO 2020

- ✦ Insert a discussion about **hypothesis testing**

sklearn.model_selection.cross_val_score

```
from sklearn.model_selection import cross_val_score

lin_scores = cross_val_score(lin_reg, X, y, cv=10)

print("Scores:", lin_scores)
print("Mean:", lin_scores.mean())
print("Standard deviation:", lin_scores.std())
```

```
Scores: [66782.73843989 66960.118071 70347.95244419 74739.57052552
        68031.13388938 71193.84183426 64969.63056405 68281.61137997
        71552.91566558 67665.10082067]
Mean: 69052.46136345083
Standard deviation: 2731.674001798348
```

sklearn.model_selection.cross_val_score

```
from sklearn.model_selection import cross_val_score

tree_rmse_scores = cross_val_score(tree_reg, X, y, cv=10)

print("Scores:", tree_rmse_scores)
print("Mean:", tree_rmse_scores.mean())
print("Standard deviation:", tree_rmse_scores.std())
```

```
Scores: [70194.33680785 66855.16363941 72432.58244769 70758.73896782
         71115.88230639 75585.14172901 70262.86139133 70273.6325285
         75366.87952553 71231.65726027]
Mean: 71407.68766037929
Standard deviation: 2439.4345041191004
```


Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.
 - ❖ A **grid search** is a better approach.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.
 - ❖ A **grid search** is a better approach.
 - ❖ Systematically **enumerate** all the possible combinations of hyperparameters.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.
 - ❖ A **grid search** is a better approach.
 - ❖ Systematically **enumerate** all the possible combinations of hyperparameters.
 - ❖ For **each combination**, train a model on the **training set** and test its performance on the **validation set**.

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.
 - ❖ A **grid search** is a better approach.
 - ❖ Systematically **enumerate** all the possible combinations of hyperparameters.
 - ❖ For **each combination**, train a model on the **training set** and test its performance on the **validation set**.
 - ❖ Initially, you try **powers** of two (2) or ten (10).

Grid search

- ❖ Most learning algorithms have **many hyperparameters** that need tuning.
 - ❖ In fact, this is one of the **major disadvantages** of machine learning algorithms.
- ❖ Often, people manually explore various combinations.
 - ❖ A **grid search** is a better approach.
 - ❖ Systematically **enumerate** all the possible combinations of hyperparameters.
 - ❖ For **each combination**, train a model on the **training set** and test its performance on the **validation set**.
 - ❖ Initially, you try **powers** of two (2) or ten (10).
 - ❖ If time allows, conduct another grid search with values close to the optimal values found in the previous iteration.

sklearn.model_selection.GridSearchCV

See: Géron 2019 §2

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30],
     'max_features': [2, 4, 6, 8]}
]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5)

grid_search.fit(X_train, y_train)

grid_search.best_params_
```

```
{'max_features': 8, 'n_estimators': 30}
```

Randomized search

- ❖ What if the **number of combinations is large** (many hyperparameters, many values for each)
- ❖ Scikit-Learn provides **RandomizedSearchCV**
 - ❖ The user can either supply a list of values for each hyperparameters or a probability distribution (a method for sampling values)
 - ❖ The user also specifies the number of iterations, that is the number of combinations to try.
 - ❖ Makes the execution time more predictable.

7-Steps workflow

Workflow [Deep Learning with Python]

- ❖ Defining the problem and assembling the data
- ❖ Choosing a measure of success
- ❖ Choosing an evaluation protocol
- ❖ Preparing the data
- ❖ Developing an initial model
- ❖ Developing a model that overfits
- ❖ Regularization and hyper parameter tuning

Source: [4]

Prologue

Summary

- ✦ We discussed of the roles of the **training**, **validation**, and **test** sets

Summary

- ❖ We discussed of the roles of the **training**, **validation**, and **test** sets
- ❖ We also talked about **cross-validation**: *k*-folds and leave-one-out

Summary

- ❖ We discussed of the roles of the **training**, **validation**, and **test** sets
- ❖ We also talked about **cross-validation**: k -folds and leave-one-out
- ❖ Underfitting and overfitting are important concepts





Summary

- ❖ We discussed of the roles of the **training**, **validation**, and **test** sets
- ❖ We also talked about **cross-validation**: k -folds and leave-one-out
- ❖ Underfitting and overfitting are important concepts
- ❖ We looked at grid search and randomized search

Next module

- ❖ **Training** - gradient descent

References

-  Nathalie Japkowicz and Mohak Shah.
Evaluating Learning Algorithms: a classification perspective.
Cambridge University Press, Cambridge, 2011.
-  Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.
-  Andriy Burkov.
The Hundred-Page Machine Learning Book.
Andriy Burkov, 2019.
-  François Chollet.
Deep learning with Python.
Manning Publications, 2017.



Marcel Turcotte

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science (EECS)**
University of Ottawa