Machine Learning for Networking
ML4N

Luca Vassio
Gabriele Ciravegna
Zhihao Wang
Tailai Song

Model selection and validation

- Overfitting vs. underfitting
- Train on trainset, then validate on validation set
- Choose model with minimum validation error on the hypothesis
- Compute test error for chosen hypothesis

Performance metrics

Loss vs Metric

- Losses are for machines
  - Minimize ERM
  - Tuning parameters (**w**) of a model
- Metrics are for humans
  - Measure of quality
  - Choosing hyper-parameters of a model (not directly optimized by ERM)

Loss vs Metric

- **0/1 loss** → misclassified samples
- **Logistic loss** → easy to optimize

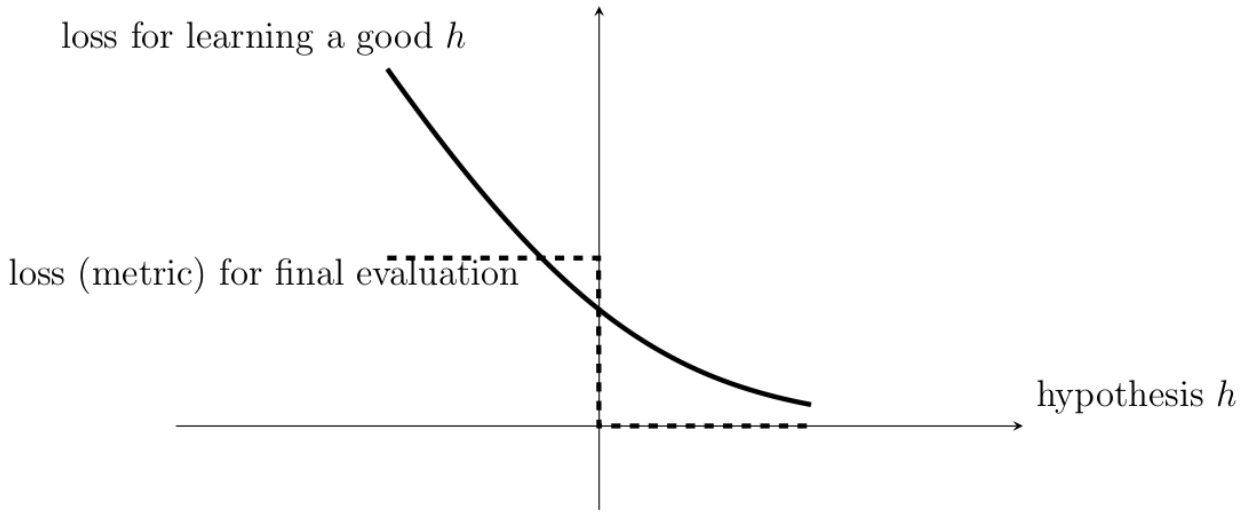


Figure 2.13: Two different loss functions for a given data point and varying hypothesis h. One of these loss functions (solid curve) is used to learn a good hypothesis by minimizing the loss. The other loss function (dashed curve) is used to evaluate the performance of the learnt hypothesis. The loss function used for this final performance evaluation is sometimes referred to as a metric.

Evaluation of supervised ML

- Efficiency
  - model building time
  - prediction time
- Scalability
  - training set size
  - attribute number
- Robustness
  - noise, missing data
- Interpretability
  - model interpretability
  - model compactness

Evaluation of regression techniques

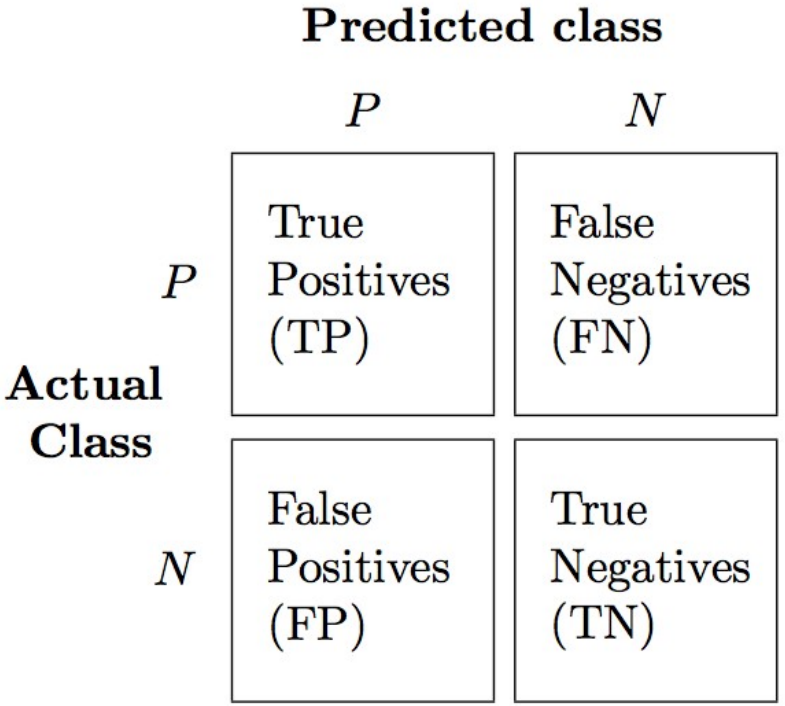
- Quality of the prediction
  - Mean squared error
  - Mean absolute error
  - ...

Evaluation of classification techniques

- Quality of the prediction
  - Confusion matrix
  - Accuracy
  - Precision, Recall, F-measure (per class, or average)
  - ROC curve
  - ...

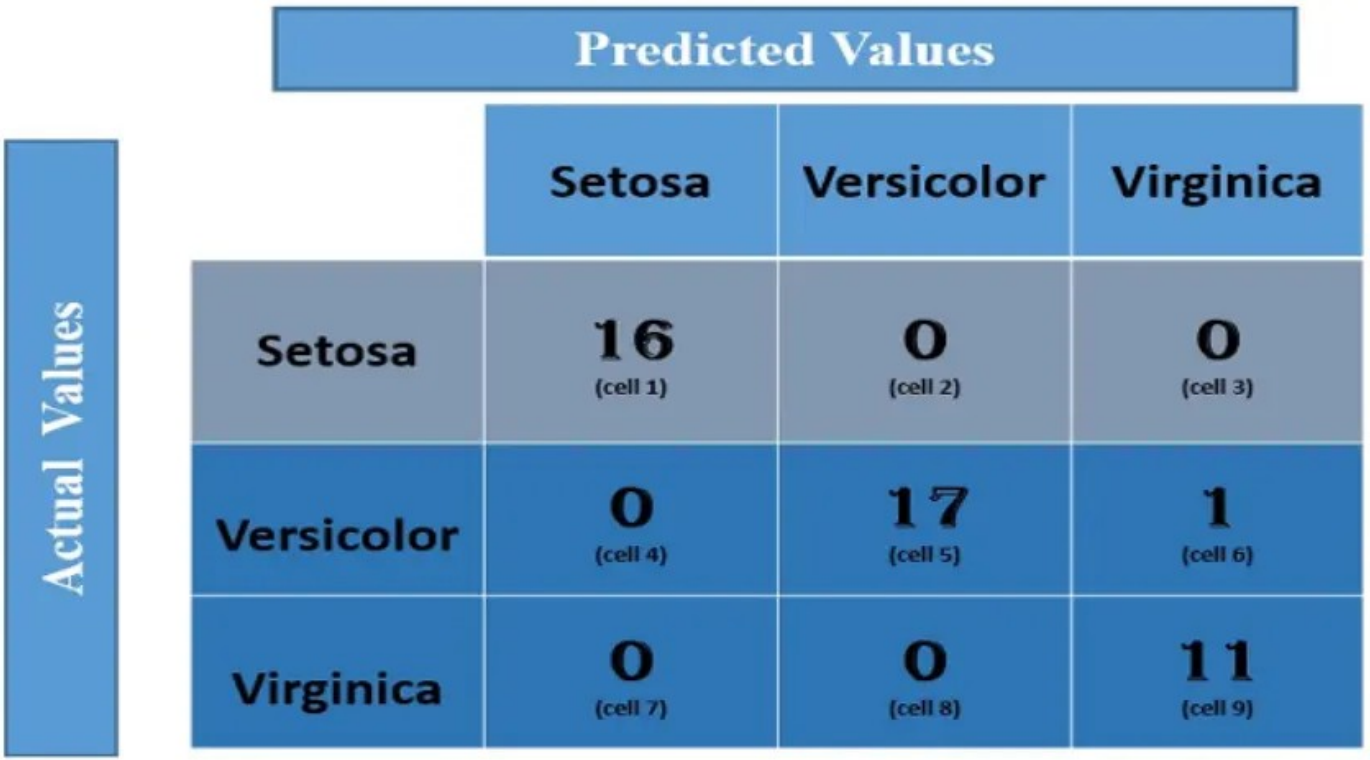
Confusion matrix

- For binary classifiers



Confusion matrix

- For multi-class classifiers



Actual Values	Predicted Values: Setosa	Predicted Values: Versicolor	Predicted Values: Virginica
Setosa	16 (cell 1)	0 (cell 2)	0 (cell 3)
Versicolor	0 (cell 4)	17 (cell 5)	1 (cell 6)
Virginica	0 (cell 7)	0 (cell 8)	11 (cell 9)

Accuracy

- Most widely-used metric for model evaluation

$$\text{Accuracy} = \frac{\text{Number of correctly classified objects}}{\text{Number of classified objects}}$$

- For binary classifiers

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy

- Accuracy is related to **0/1 loss**

$$\text{Accuracy} \approx p(y = \widehat{y})$$

$$(1/m) \sum_{i=1}^{m} L\left((\mathbf{x}^{(i)}, y^{(i)}), h\right) \approx p(y \neq \hat{y})$$

$$\text{Accuracy} = 1 - (1/m) \sum_{i=1}^{m} L\left((\mathbf{x}^{(i)}, y^{(i)}), h\right).$$

Accuracy

- Its value might be misleading
- Consider a binary problem $y \in \{P, N\}$
  - Cardinality of class P = 9900
  - Cardinality of class N = 100
- Model h(**x**): class P for all **x**
- Model predicts everything to be class P
- Accuracy is 9900/10000 = 99.0 %
- Model has 99% accuracy but does not detect any class N object

Accuracy

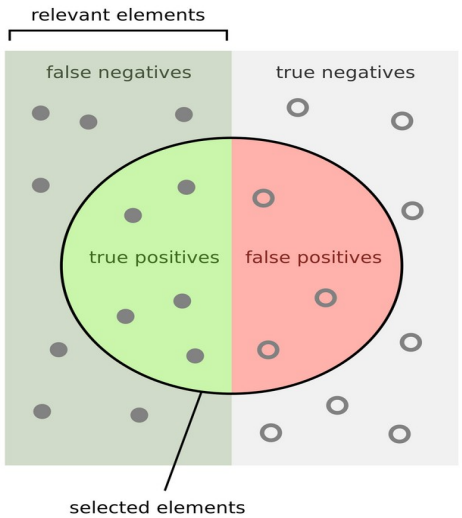
- Not appropriate for different class relevance/importance
- Misclassification of objects of a given class is more important
  - e.g., ill patients erroneously assigned to the healthy patients class

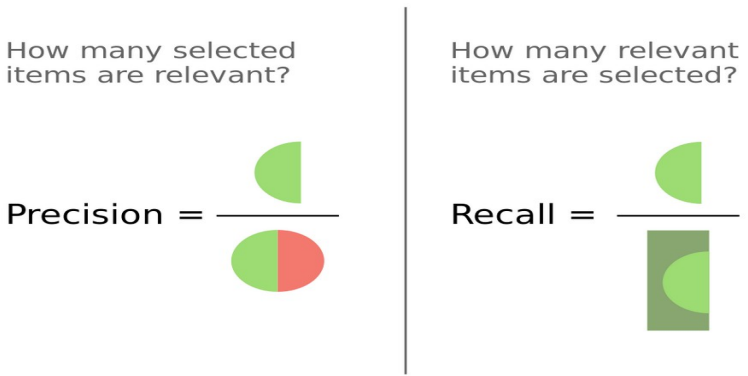
Class specific measures

- Evaluate separately for each class c

$$\text{Precision(c)} \quad p = \frac{\text{Number of objects correctly assigned to c}}{\text{Number of objects assigned to c}}$$

$$\text{Recall(c)} \quad r = \frac{\text{Number of objects correctly assigned to c}}{\text{Number of objects belonging to c}}$$





Class specific measures

- Evaluate separately for each class c

$$\text{Precision(c)} \quad p = \frac{\text{Number of objects correctly assigned to c}}{\text{Number of objects assigned to c}}$$

$$\text{Recall(c)} \quad r = \frac{\text{Number of objects correctly assigned to c}}{\text{Number of objects belonging to c}}$$

F-measure or F1-score: harmonic mean of the precision and recall

$$\text{F-measure}(c) F = \frac{2rp}{r+p}$$

Class specific measures

- For a binary classification problem
- For the positive class (c=positive)
  - Precision
  - Recall (True Positive Rate)
  - F-measure (F1-score)

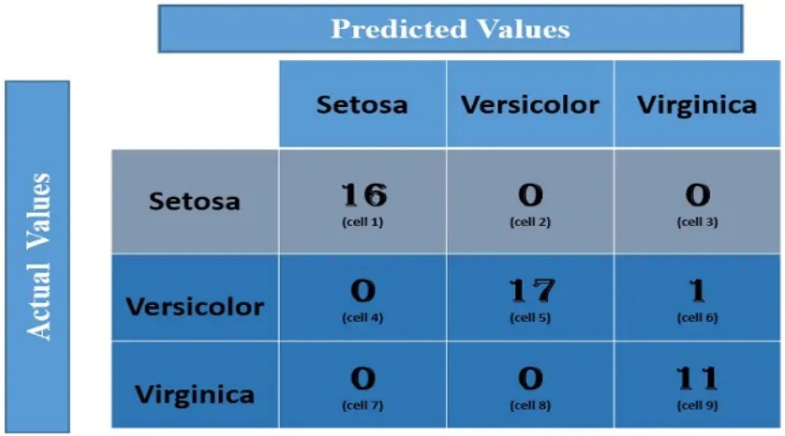
$$p = \frac{TP}{TP + FP}$$

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

$$F = \frac{2rp}{r + p} = \frac{2TP}{2TP + FP + FN}$$

Class-specific measure

- For multi-class classifiers



We can use the same formula as for the binary case

Calculate the TP, TN, FP, and FN values for each class from confusion matrix:

- **TP**: The actual value and predicted value should be the same. For Setosa class, the value of cell 1 (16)
- **FN**: The sum of values of corresponding row except for the TP value. For Setosa, the value of cell 2 (0) and 3 (0)
- **FP**: The sum of values of the corresponding column except for the TP value. For Setosa, cell 4 (0) and cell 7 (0)
- **TN**: The sum of values of all columns and rows except the values of that class that we are calculating the values for. For Setosa, cell 5 (17), cell 6 (1), cell 8 (0) and cell 9 (11)

ROC curve

- ROC (Receiver Operating Characteristic) developed for analyzing noisy signals and finding the trade-off between positive hits and false alarms
- ROC curve plots
  - TPR, True Positive Rate (same as Recall)

    TPR = TP/(TP+FN)

    On the **y-axis**
  - FPR, False Positive Rate

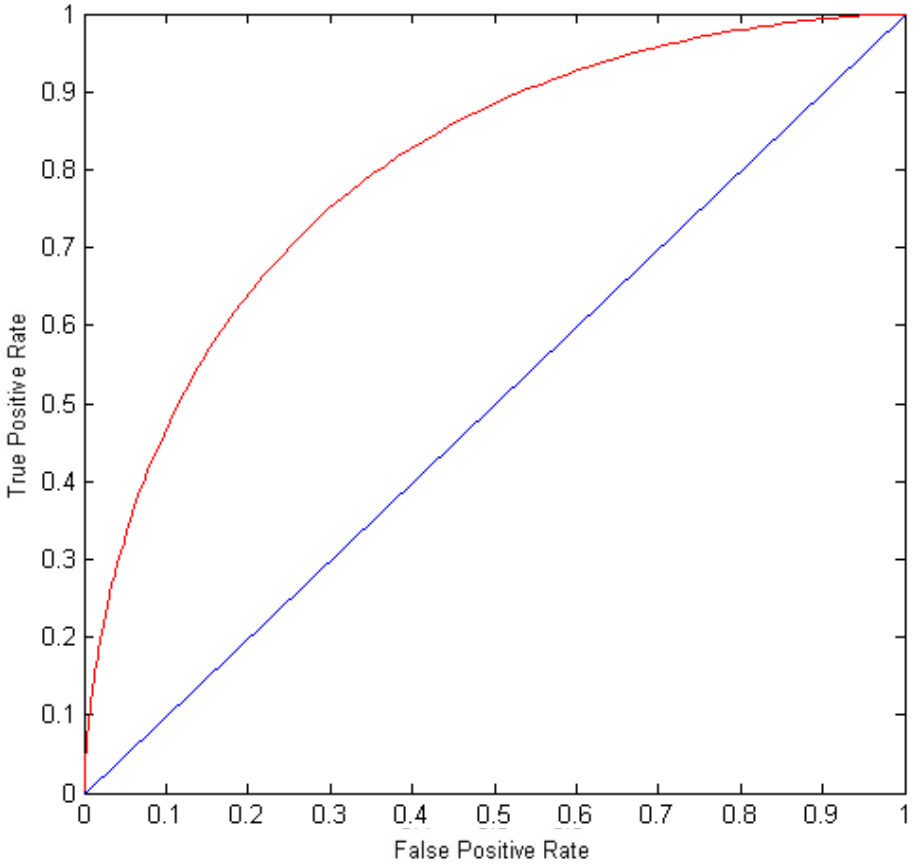
    FPR = FP/(FP + TN)

    On the **x-axis**

ROC curve

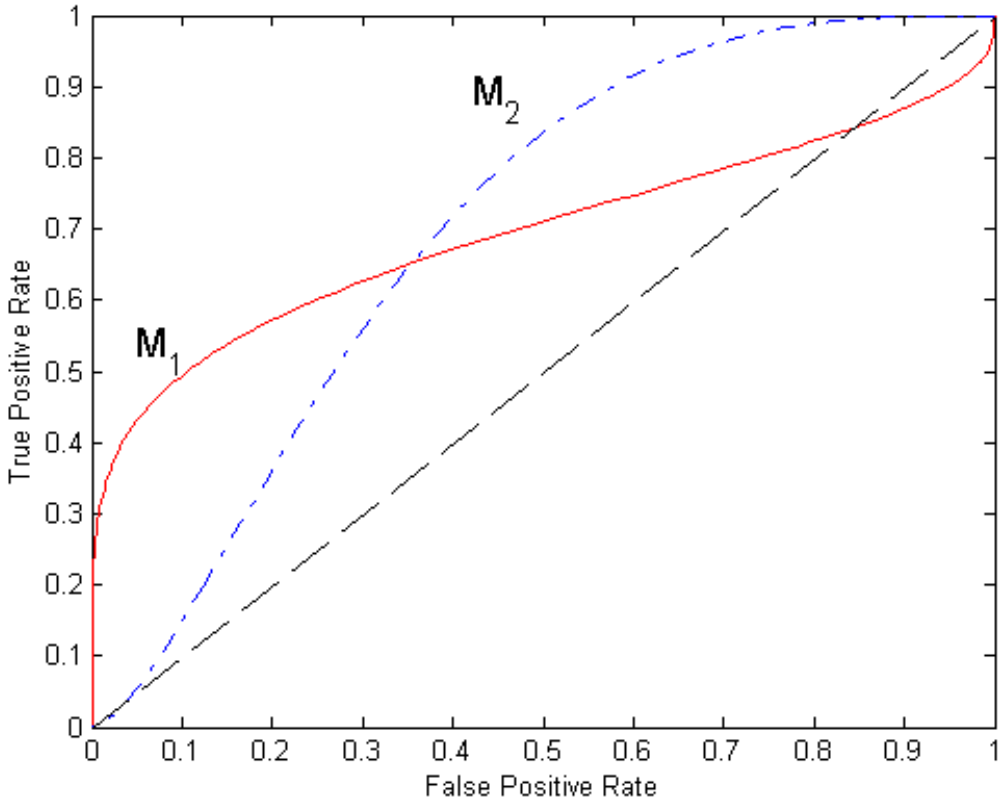
(FPR, TPR)

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal
- Diagonal line
  - Random guessing (if two balanced classes)
- Below diagonal line
  - prediction is opposite of the true class



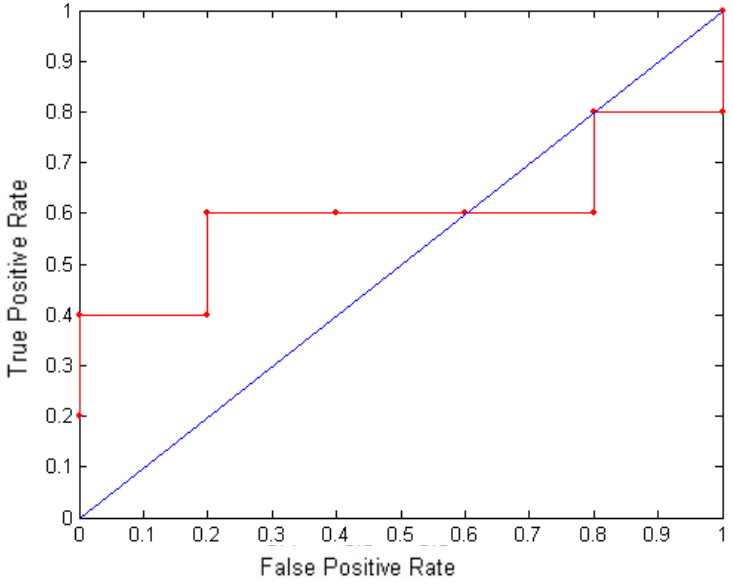
ROC curve

- No model consistently outperforms the other
  - M1 is better for small FPR
  - M2 is better for large FPR
- Area under ROC curve
  - Ideal area = 1.0
  - Random guess area = 0.5



How to build ROC curve

- Use classifier that produces posterior probability P(+|x) for each sample x
- Order the samples according to P(+|x)
- Apply threshold at each unique value of P(+|x)
- Compute TPR and FPR at each threshold



10 samples

Class	+	-	+	-	-	-	+	-	+	+	
P(+|x)	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Metrics in python

3.3. Metrics and scoring: quantifying the quality of predictions

There are 3 different APIs for evaluating the quality of a model's predictions:

- **Estimator score method**: Estimators have a `score` method providing a default evaluation criterion for the problem they are designed to solve. This is not discussed on this page, but in each estimator's documentation.
- **Scoring parameter**: Model-evaluation tools using cross-validation (such as `model_selection.cross_val_score` and `model_selection.GridSearchCV`) rely on an internal *scoring* strategy. This is discussed in the section The scoring parameter: defining model evaluation rules.
- **Metric functions**: The `sklearn.metrics` module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

https://scikit-learn.org/stable/modules/model_evaluation.html

Use a different loss for Train and Val

- We can use **different loss for training and validation**
- This enables the **comparison of different ML methods**
- Logistic regression uses **log loss** to learn hypothesis $h_1(x)$
- SVM uses **hinge loss** to learn hypothesis $h_2(x)$
- Compare h_1, h_2 by their **average 0/1 loss (accuracy)** on validation set

Use a different loss for Train and Val

Compare Logistic regression and SVM in Python:

```
# split dataset into training and validation set
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)

# learn hypothesis h1(x) by minimizing its average logistic loss on the training set
logreg= LogisticRegression(random_state=0).fit(X_train, y_train)
# compute validation error of learnt hypothesis h1(x) using average 0/1 loss (accuracy)
val_error_logreg = logreg.score(X_val, y_val)

svmclf = svm.SVC()
#learn hypothesis h2(x) by minimizing its average hinge loss on the training set
svmclf.fit(X_train, y_train)
# compute validation error of learnt hypothesis h2(x) using average 0/1 loss (accuracy)
val_error_svm = svmclf.score(X_val, y_val)

print("accuracy of logistic regression on validation set:",val_error_logreg)
print("accuracy of logistic regression on validation set:",val_error_svm)
```


https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html
https://scikit-learn.org/dev/modules/generated/sklearn.svm.SVC.html

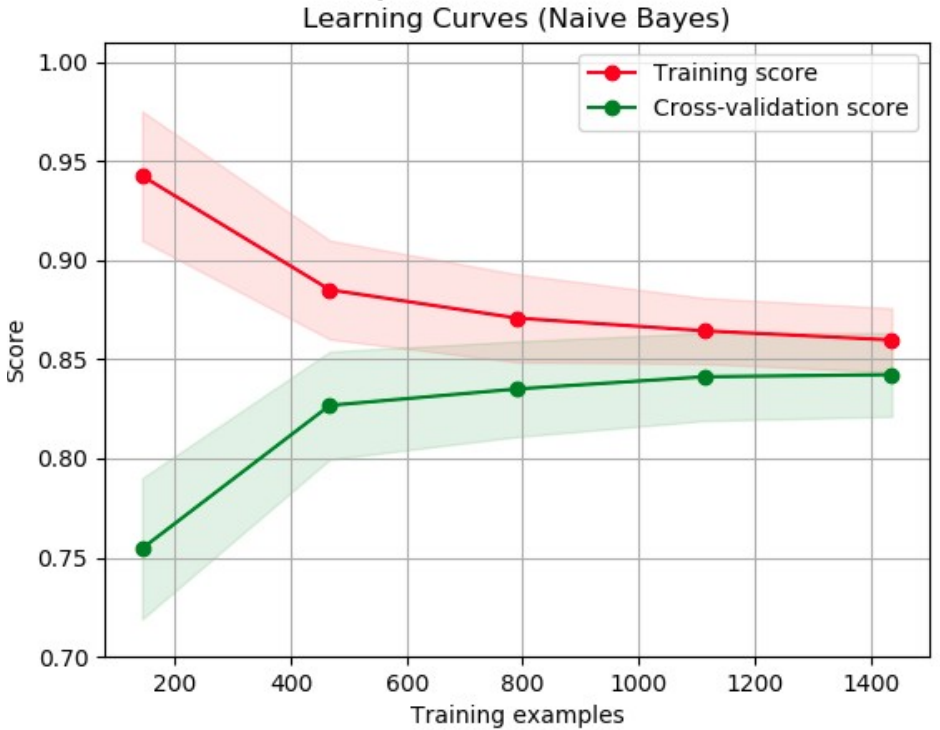
Learning curve for amount of data

- Learning curve shows how a **metric (or loss) changes** with varying **training sample size**
- It is a tool to find out how much we benefit (on **validation**) from adding more training data
- Requires a **sampling schedule** for creating learning curve
- Possible effects of too small sample size:
  - Bias in the estimate (poor performance)
  - Variance of estimate (results dependent on small changes)

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Learning curve for amount of data

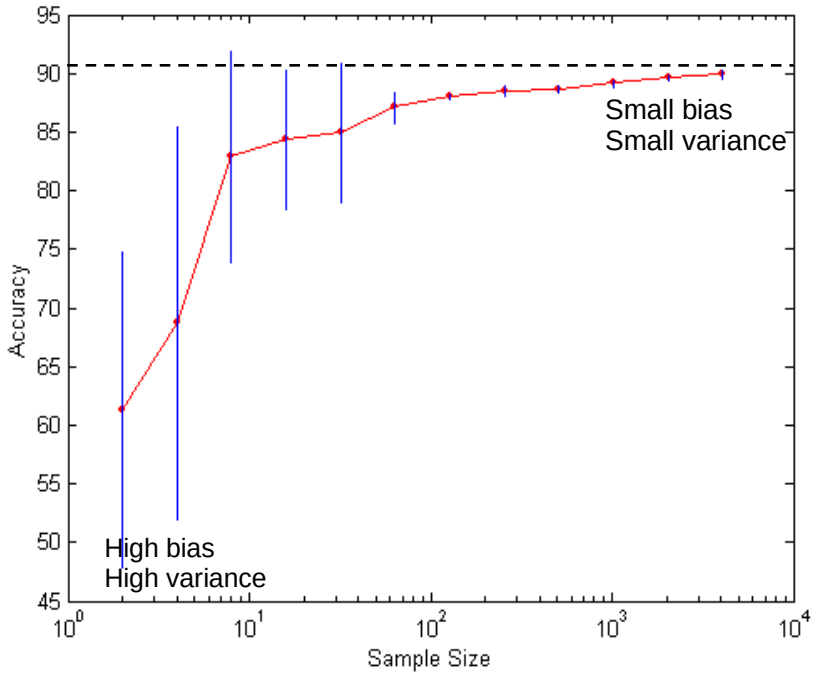
- Learning curve shows how a **metric (or loss) changes** on training and validation with varying **training sample size**



https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html#sklearn.model_selection.learning_curve

Learning curve for amount of data

Usually, on **validation** (size of validation is constant):



Hyper-parameter tuning

Hyper-parameter tuning

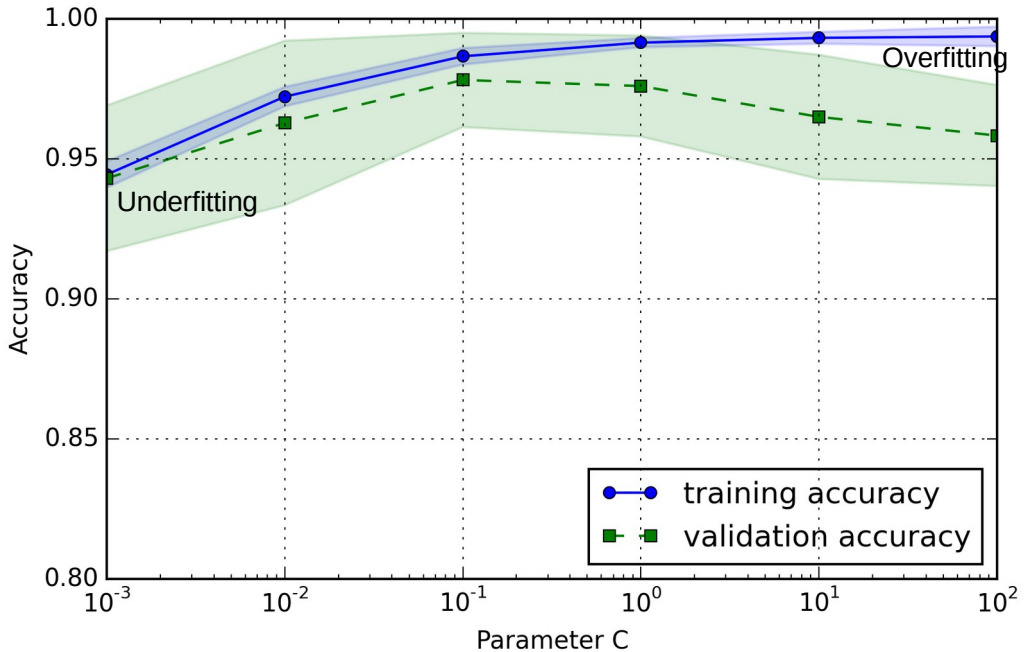
- **Hyper-parameters** are parameters that are **not directly learnt** within estimators
- Hyper-parameters of the model defines the model space H
- In scikit-learn they are passed as arguments to the constructor of the estimator classes
- Search the hyper-parameter space for the best (cross) validation score (loss or metric)

Validation curve

- **Influence of a single hyper-parameter** on the **training** score and the **validation** score
- Score can be metric or (inverse of) loss
- A different hypothesis is trained for different values of the hyper-parameter
- Find out whether the estimator is overfitting or underfitting for some hyperparameter values
  - If the training score and the validation score are both low, the estimator is **underfitting**
  - If the training score is high and the validation score is low, the estimator is **overfitting.**

Validation curve

- **Influence of a single hyper-parameter** on the **training** score and the **validation** score



https://scikit-learn.org/stable/modules/learning_curve.html

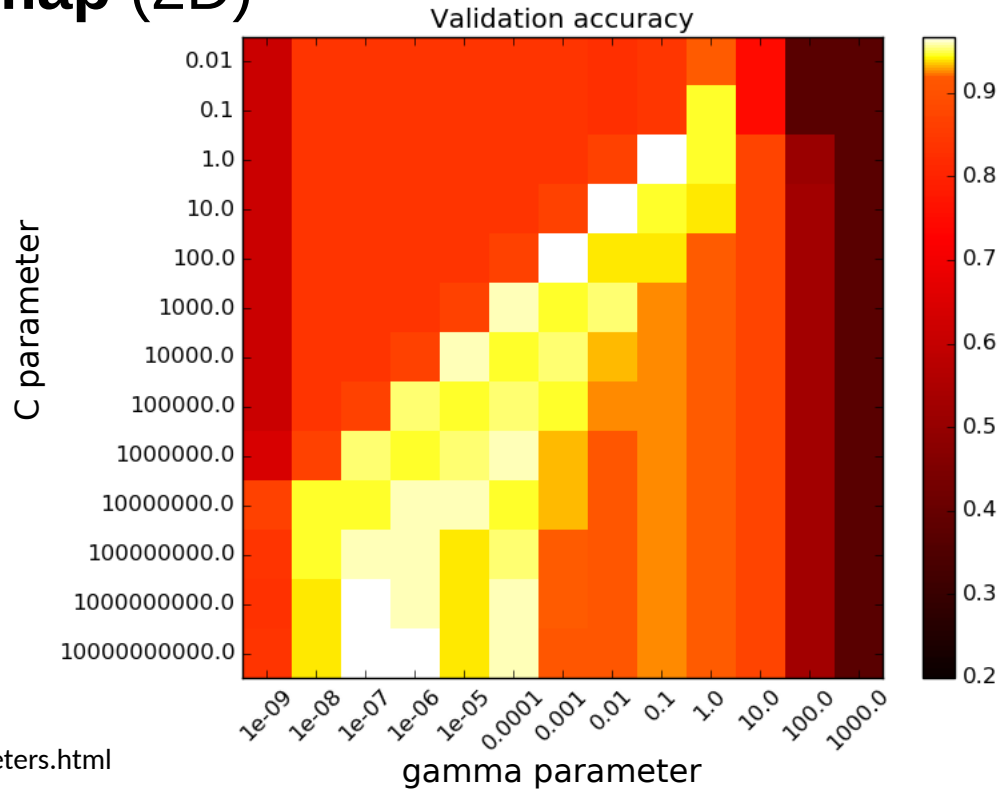
Grid search

- Exhaustively generates candidates from a **grid of hyper-parameter values**
- A different hypothesis trained for all possible combination of hyper-parameters values
- Can be done for many hyper-parameters (2 or more)

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Grid search

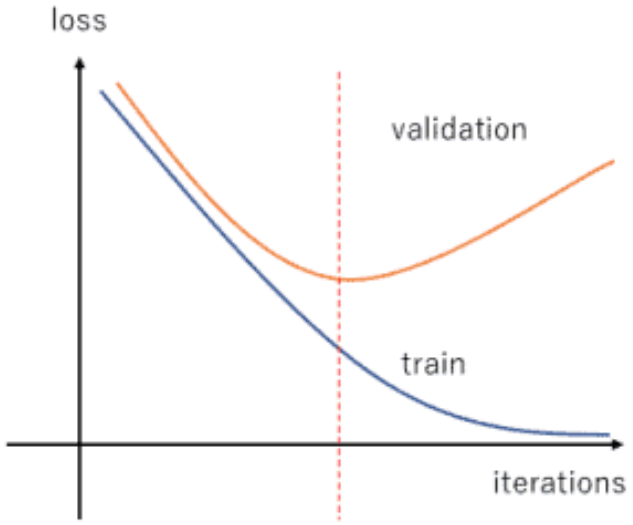
- In case of 2 hyper-parameters: **validation score** can be represented as a **heatmap** (2D)



http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Training/learning curve for number of iterations

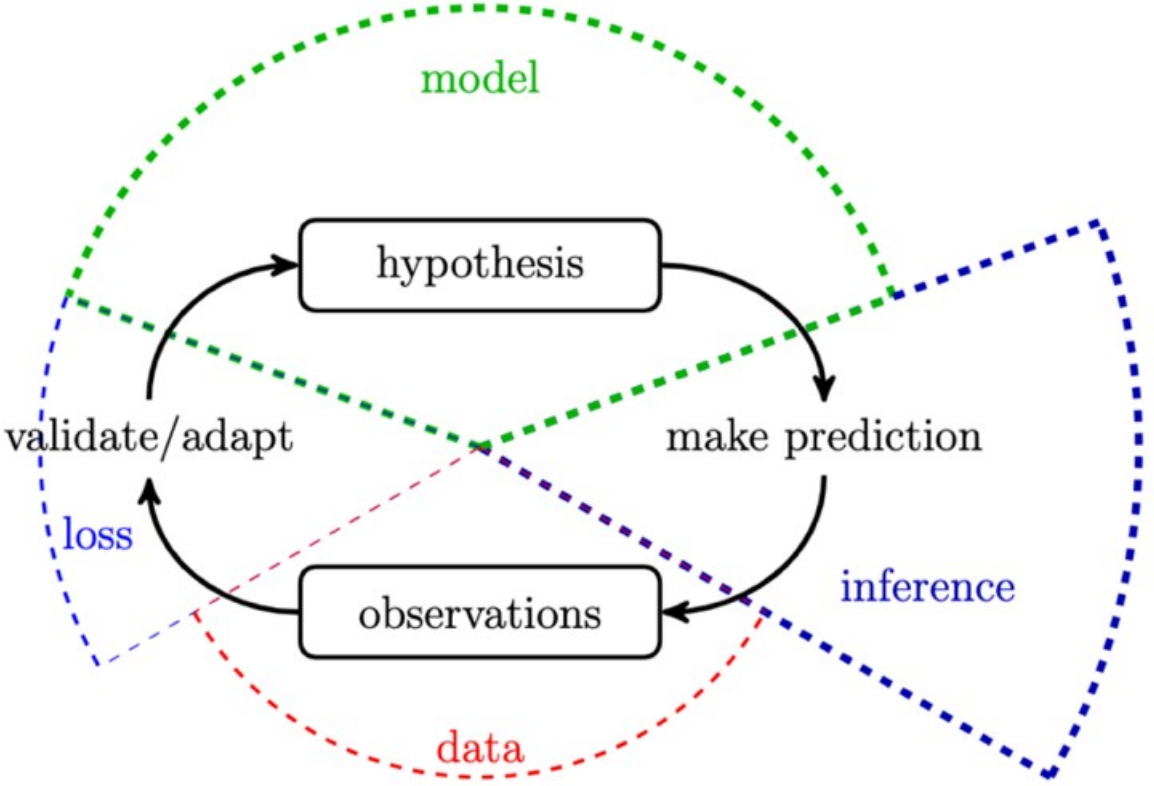
- **Many algorithms/methods to solve ERM** are based on **iterative improvements** of the empirical risk (see next lecture)
- At each iterations, a **different hypothesis** is proposed
- These iterations are often called **epochs**
- When to stop?
- Iterations/epochs can be seen as an **hyper-parameter of the algorithm/method** (not hyper-parameters of the model H itself)
- Score can be metric or (inverse of) loss
- Another hyper-parameter of the algorithm is the **learning rate**



Summary

- Performance evaluation through metrics
- Hyper-parameter tuning = model selection
- Hyper-parameter tuning with validation curve and grid search
- Monitor learning curve → changes increasing the data samples
- Monitor training curve → when to stop learning

ML process


model
hypothesis
validate/adapt
make prediction
loss
observations
inference
data

ML process

- Load/generate datasets
- Choose samples, features and labels
- Split the dataset in training, validation (or later cv) and testing
- Pre-processing techniques
- Define model and loss
- Perform ERM with an algorithm (learn)
- Evaluate the obtained performance (metrics)
- Tune the hyper-parameters (and check learning/training curves)
- Test the final hypothesis on test data
- Use it in the wild for inference

Any questions?

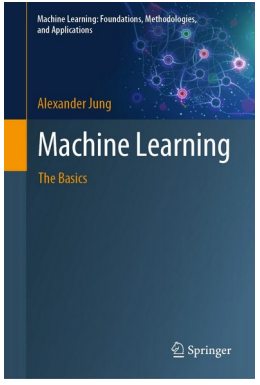
Self-assessment quiz

- Given the following data and prediction output, compute:
  - Accuracy
  - Precision and Recall (per class and average)
  - ROC curve for class A (given the posterior p(A|**x**))

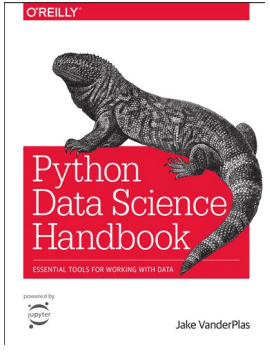
x	y	h(**x**)	p(A|**x**)
$\mathbf{x}_1$	A	A	0.9
$\mathbf{x}_2$	A	A	0.6
$\mathbf{x}_3$	C	B	0.2
$\mathbf{x}_4$	B	B	0.1
$\mathbf{x}_5$	C	C	0.2
$\mathbf{x}_6$	C	C	0.3
$\mathbf{x}_7$	A	A	0.7
$\mathbf{x}_8$	A	B	0.4
$\mathbf{x}_9$	B	A	0.8
$\mathbf{x}_{10}$	A	A	0.8
$\mathbf{x}_{11}$	A	C	0.3
$\mathbf{x}_{12}$	A	A	0.9
$\mathbf{x}_{13}$	B	B	0.1
$\mathbf{x}_{14}$	C	C	0.1

References: readings

- Chapter 2-6



- Chapter 5



- Further resources: https://scikit-learn.org/stable/model_selection.html
  https://scikit-learn.org/stable/modules/model_evaluation.html

Slide acknowledgments

- Alexander Jung – Aalto University
- Tania Cerquitelli and Elena Maria Baralis – Politecnico di Torino