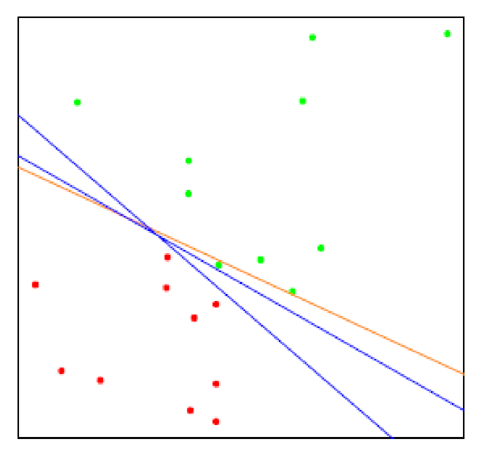


Parallel & Scalable Machine Learning

Introduction to Machine Learning Algorithms

Prof. Dr. – Ing. Morris Riedel

Adjunct Associated Professor

School of Engineering and Natural Sciences, University of Iceland

Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 10

Theory of Generalization

February 27th, 2019

Juelich Supercomputing Centre, Juelich, Germany

PRACE PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE TRAINING PORTAL

UNIVERSITY OF ICELAND

SCHOOL OF ENGINEERING AND NATURAL SCIENCES

FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

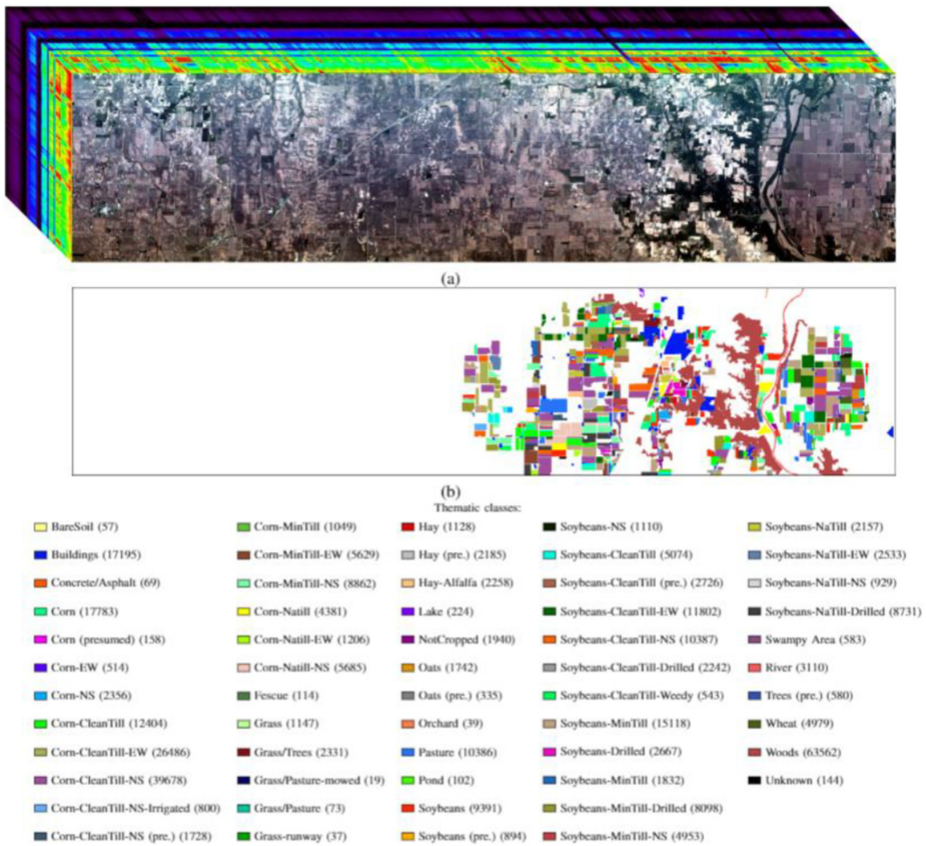
JÜLICH Forschungszentrum

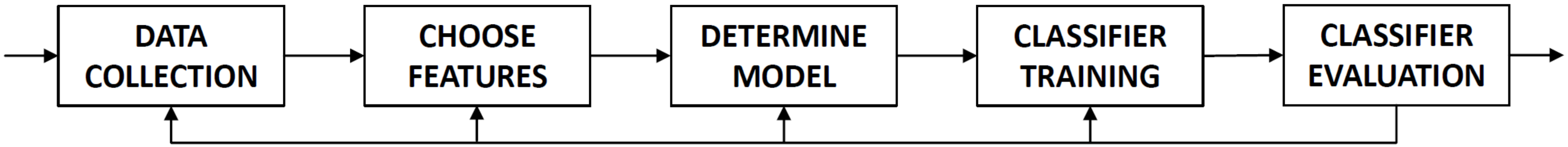
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DEEP Projects

Review of Lecture 9 – Data Preparation & Performance Evaluation

- Real Datasets are challenging
  - High number of classes
  - High dimensional datasets
  - Unbalanced class problems
- Machine Learning
  - Not just use dataset with any kind of algorithms (e.g. ANNs)
  - Instead substantial feature selection & engineering before
  - How to choose a model given the data amount we have?



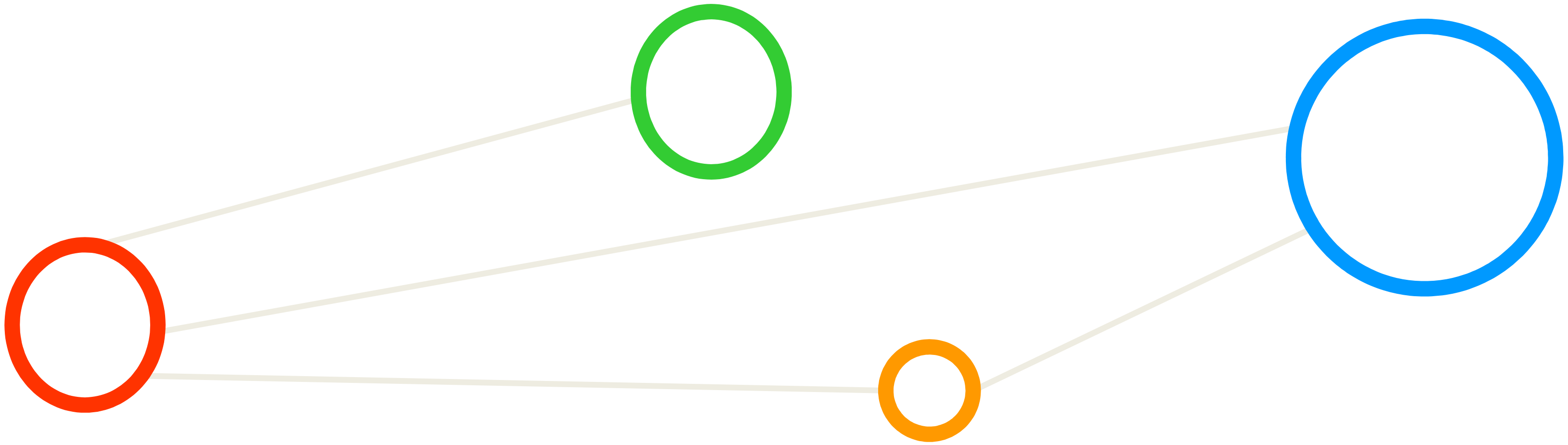


Outline of the Course

Theoretical Lectures

Practical Lectures

Outline

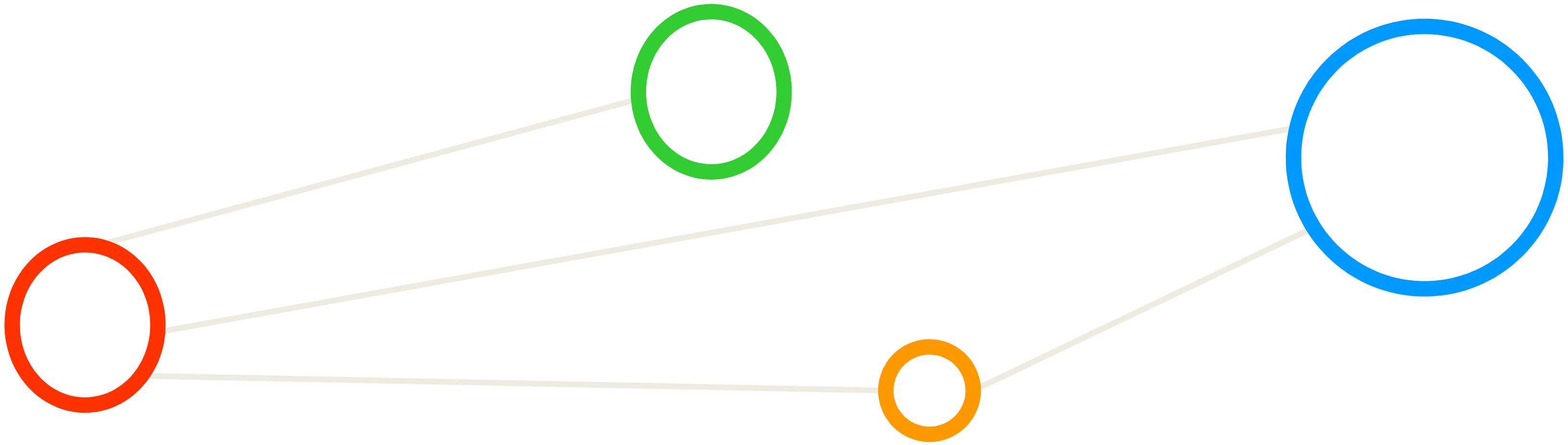


Outline

- Generalization in Supervised Learning
  - Formalization of Machine Learning
  - Mathematical Building Blocks & Linear Model Example
  - Feasibility of Learning & Degrees of Freedom
  - Hypothesis Set & Final Hypothesis
  - Learning Models & Validation Dependencies

- Learning Theory Basics
  - Union Bound & Problematic Factor M
  - Theory of Generalization
  - Linear Perceptron Example in Context
  - Model Complexity & VC Dimension
  - Problem of Overfitting

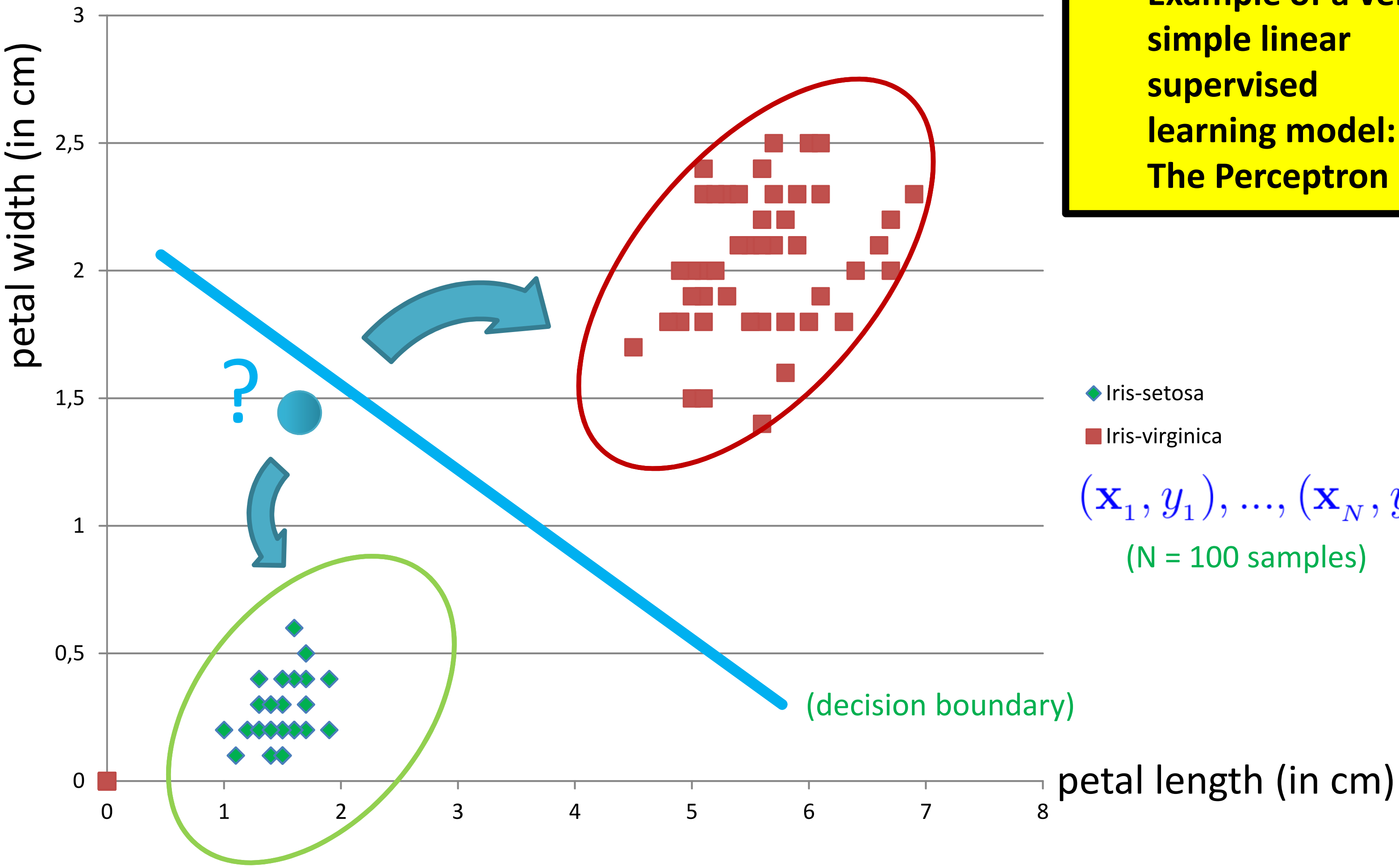
Generalization in Supervised Learning



AUDIENCE QUESTION

What means generalization and why it is important?

Supervised Learning & Generalization for New/Unseen Data



- **Example of a very simple linear supervised learning model: The Perceptron**

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(N = 100 samples)

Learning Approaches – Supervised Learning – Formalization

- Each observation of the predictor measurement(s) has an associated response measurement:
  - Input $\mathbf{x} = x_1, ..., x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$

Training Examples

$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

- Goal: Fit a model that relates the response to the predictors
  - Prediction: Aims of accurately predicting the response for future observations
  - Inference: Aims to better understanding the relationship between the response and the predictors

- **Supervised learning approaches fits a model that related the response to the predictors**
- **Supervised learning approaches are used in classification algorithms such as SVMs**
- **Supervised learning works with data = [input, correct output]**

[1] An Introduction to Statistical Learning

Feasibility of Learning

- **Statistical Learning Theory deals with the problem of finding a predictive function based on data**

[2] Wikipedia on 'statistical learning theory'

- Theoretical framework underlying practical learning algorithms
  - E.g. Support Vector Machines (SVMs)
  - Best understood for 'Supervised Learning'

- Theoretical background used to solve 'A learning problem'
  - Inferring one 'target function' that maps between input and output
  - Learned function can be used to predict output from future input (fitting existing data is not enough)

Unknown Target Function

$f : X \rightarrow Y$

(ideal function)

Terminologies & Different Dataset Elements

- Target Function $f : X \to Y$
  - Ideal function that 'explains' the data we want to learn
- Labelled Dataset (samples)
  - 'in-sample' data given to us: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Learning vs. Memorizing
  - The goal is to create a system that works well 'out of sample'
  - In other words we want to classify 'future data' (ouf of sample) correct
- Dataset Part One: Training set
  - Used for training a machine learning algorithms
  - Result after using a training set: a trained system
- Dataset Part Two: Test set
  - Used for testing whether the trained system might work well
  - Result after using a test set: accuracy of the trained model

Exercises – Explore Testing on Training Dataset

- **Learning exercise to understand better the theory of generalization – don‘t do this in practice!**


```
# model evaluation
score = model.evaluate(X_train, Y_train, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

MNIST Data – Testing on Training Dataset – Solution

- Memorizing vs. Generalization

```
10112/60000 [====>........................] - ETA: 1s
11552/60000 [====>........................] - ETA: 1s
12960/60000 [=====>.......................] - ETA: 1s
14400/60000 [======>......................] - ETA: 1s
15840/60000 [======>......................] - ETA: 1s
17280/60000 [=======>.....................] - ETA: 1s
18752/60000 [========>....................] - ETA: 1s
20192/60000 [=========>...................] - ETA: 1s
21632/60000 [=========>...................] - ETA: 1s
23072/60000 [==========>..................] - ETA: 1s
24512/60000 [===========>.................] - ETA: 1s
25952/60000 [===========>.................] - ETA: 1s
27392/60000 [============>................] - ETA: 1s
28832/60000 [=============>...............] - ETA: 1s
30272/60000 [==============>..............] - ETA: 1s
31712/60000 [==============>..............] - ETA: 0s
33152/60000 [===============>.............] - ETA: 0s
34592/60000 [================>............] - ETA: 0s
36032/60000 [=================>...........] - ETA: 0s
37472/60000 [=================>...........] - ETA: 0s
38912/60000 [==================>..........] - ETA: 0s
40352/60000 [===================>.........] - ETA: 0s
41792/60000 [===================>.........] - ETA: 0s
43232/60000 [====================>........] - ETA: 0s
44672/60000 [=====================>.......] - ETA: 0s
46080/60000 [======================>......] - ETA: 0s
47520/60000 [======================>......] - ETA: 0s
48768/60000 [=======================>.....] - ETA: 0s
50208/60000 [========================>....] - ETA: 0s
51648/60000 [========================>....] - ETA: 0s
53088/60000 [=========================>...] - ETA: 0s
54528/60000 [==========================>..] - ETA: 0s
55968/60000 [==========================>..] - ETA: 0s
57408/60000 [===========================>..] - ETA: 0s
58848/60000 [============================>.] - ETA: 0s
60000/60000 [==============================] - 2s 35us/step
Jsing TensorFlow backend.
est score: 0.035654142725043254
est accuracy: 0.9916
```

Mathematical Building Blocks (1)

Unknown Target Function

$f : X \rightarrow Y$

(ideal function)

Elements we not exactly (need to) know

Training Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

Elements we must and/or should have and that might raise huge demands for storage

Elements that we derive from our skillset and that can be computationally intensive

Elements that we derive from our skillset

Mathematical Building Blocks (1) – Our Linear Example

Unknown Target Function

$f : X \to Y$

(ideal function)

Training Examples

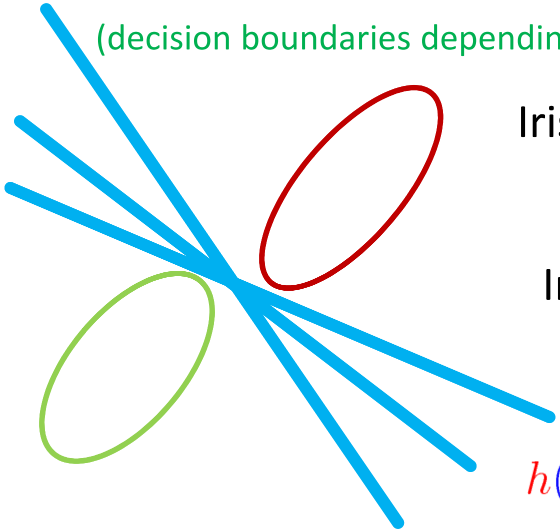
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

1. Some pattern exists
2. No exact mathematical formula (i.e. target function)
3. Data exists

(if we would know the exact target function we dont need machine learning, it would not make sense)

(decision boundaries depending on f)



Iris-virginica if $\sum_{i=1}^{d} w_i x_i > threshold$

Iris-setosa if $\sum_{i=1}^{d} w_i x_i < threshold$

(w_i and threshold are still unknown to us)

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$

(we search a function similiar like a target function)

Feasibility of Learning – Hypothesis Set & Final Hypothesis

- The ‘ideal function‘ will remain unknown in learning
  - Impossible to know and learn from data
  - If known a straightforward implementation would be better than learning
  - E.g. hidden features/attributes of data not known or not part of data

Unknown Target Function
$f : X \to Y$

- But ‘(function) approximation‘ of the target function is possible
  - Use training examples to learn and approximate it
  - Hypothesis set $\mathcal{H}$ consists of m different hypothesis (candidate functions)

$$\mathcal{H} = \{h_1, ..., h_m\};$$

‘select one function‘
that best approximates

$$g : X \to Y$$

Hypothesis Set
$\mathcal{H} = \{h\}; \;\; g \in \mathcal{H}$

Final Hypothesis
$g \approx f$

Mathematical Building Blocks (2)

Unknown Target Function

$f : X \to Y$

(ideal function)

Elements we not exactly (need to) know

Training Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

Elements we must and/or should have and that might raise huge demands for storage

Final Hypothesis

$g \approx f$

Elements that we derive from our skillset and that can be computationally intensive

Hypothesis Set

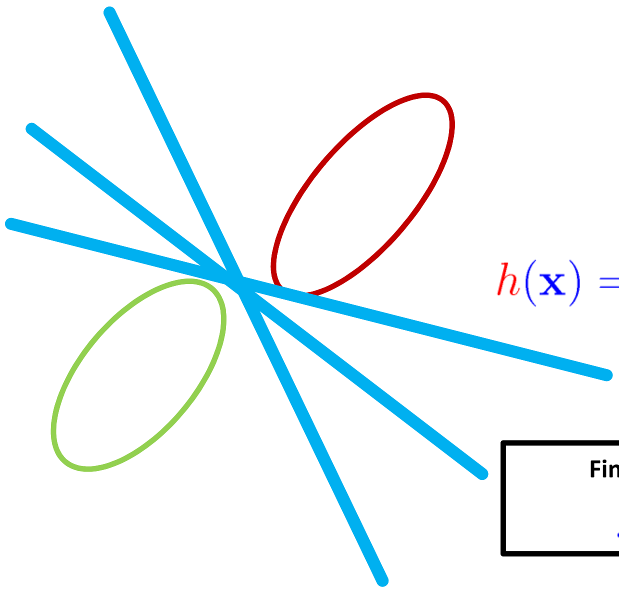
$\mathcal{H} = \{h\}; \;\; g \in \mathcal{H}$

(set of candidate formulas)

Elements that we derive from our skillset

Mathematical Building Blocks (2) – Our Linear Example

(decision boundaries depending on f)



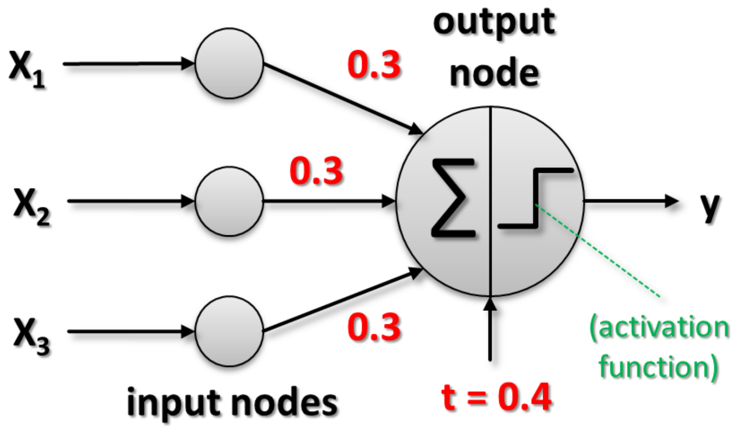
$$\mathcal{H} = \{h_1, \ldots, h_m\};$$

(we search a function similiar like a target function)

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$

Final Hypothesis
$g \approx f$

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$



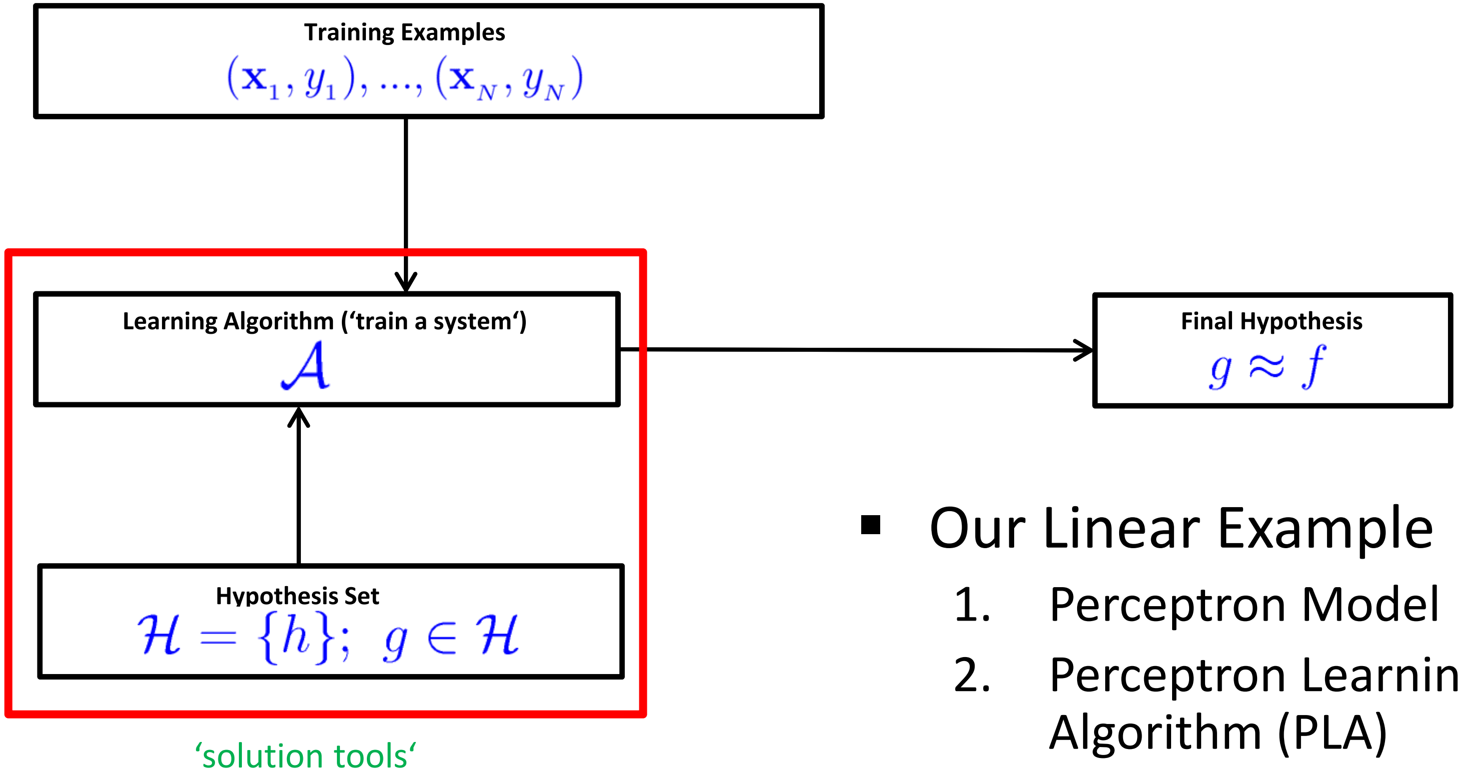
(trained perceptron model and our selected final hypothesis)

Hypothesis Set
$\mathcal{H} = \{h\}; \;\; g \in \mathcal{H}$

(Perceptron model – linear model)

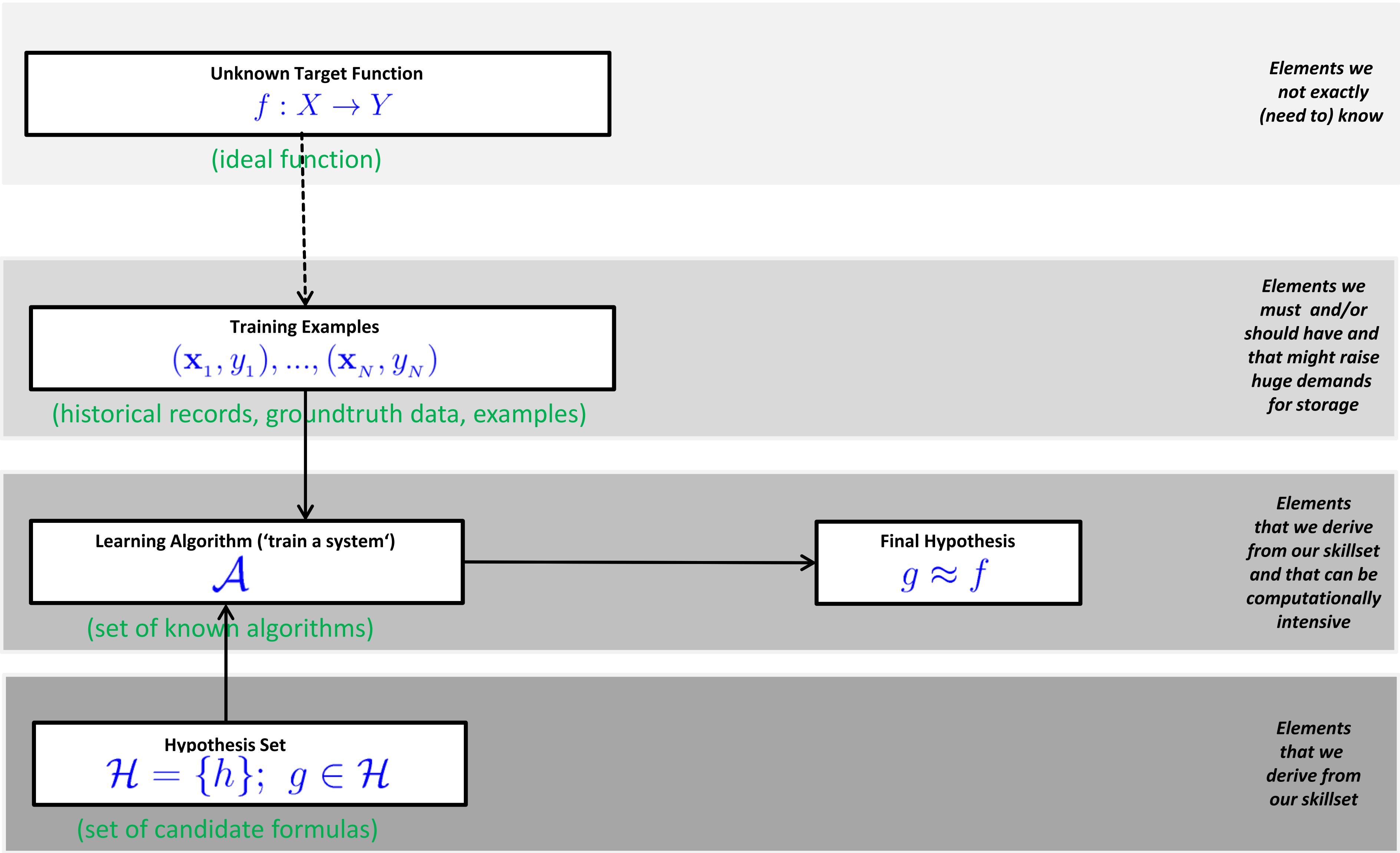
The Learning Model: Hypothesis Set & Learning Algorithm

- The solution tools – the learning model:
  1. Hypothesis set $\mathcal{H}$ - a set of candidate formulas /models
  2. Learning Algorithm $\mathcal{A}$ - ‘train a system‘ with known algorithms

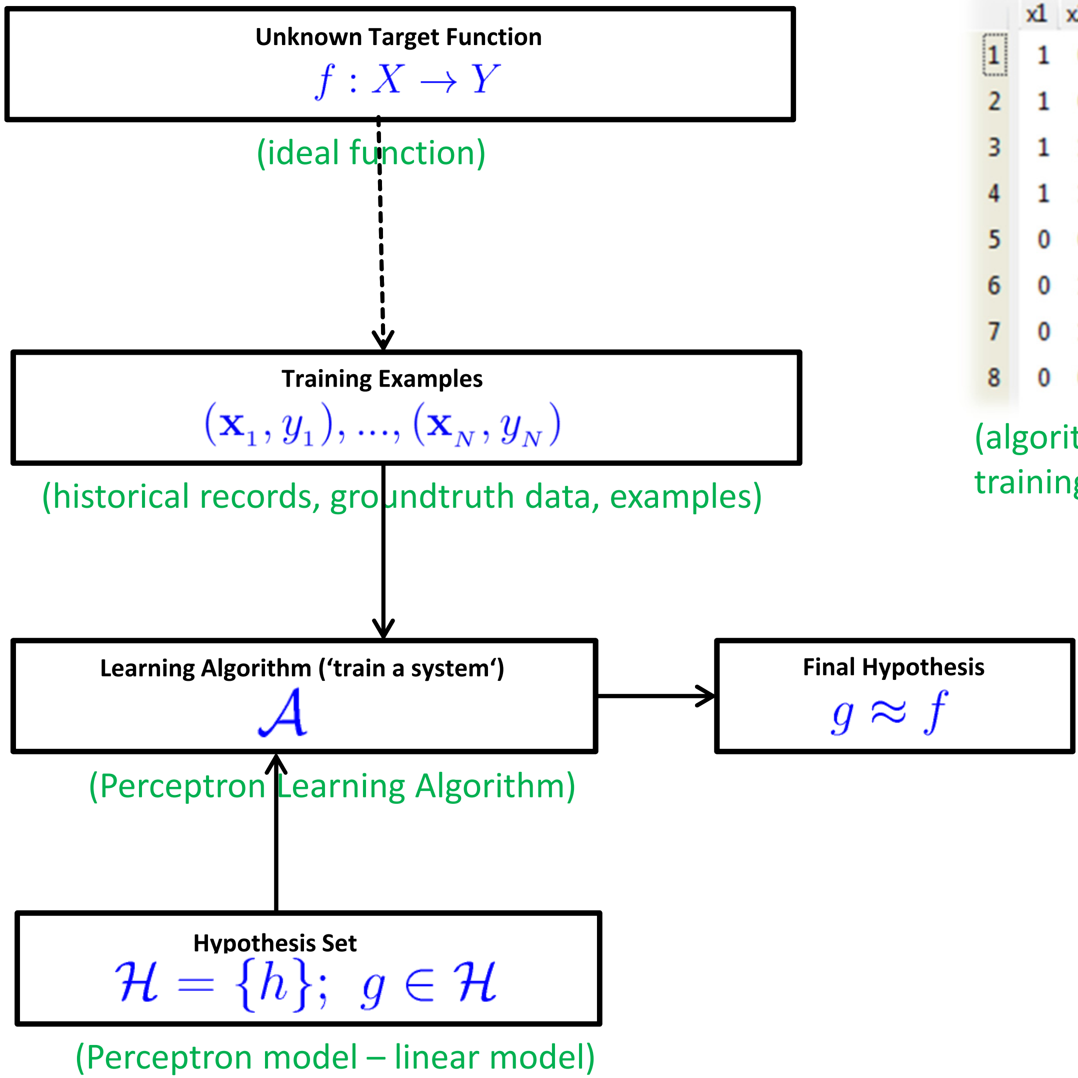


- Our Linear Example
  1. Perceptron Model
  2. Perceptron Learning Algorithm (PLA)

Mathematical Building Blocks (3)



Mathematical Building Blocks (3) – Our Linear Example



	x1	x2	x3	y
1	1	0	0	-1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1
5	0	0	1	-1
6	0	1	0	-1
7	0	1	1	1
8	0	0	0	-1

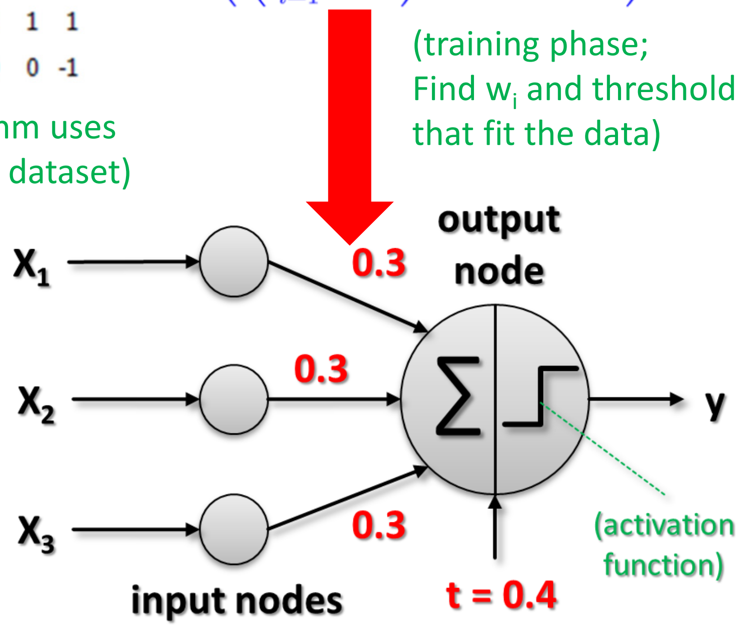
(algorithm uses training dataset)

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

(training data)

$$sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right)$$

(training phase; Find w_i and threshold that fit the data)



(trained perceptron model and our selected final hypothesis)

Different Models – Hypothesis Set & ‘Degrees of Freedom‘

Hypothesis Set
$\mathcal{H} = \{h\};\ g \in \mathcal{H}$

$$\mathcal{H} = \{h_1, \ldots, h_m\};$$

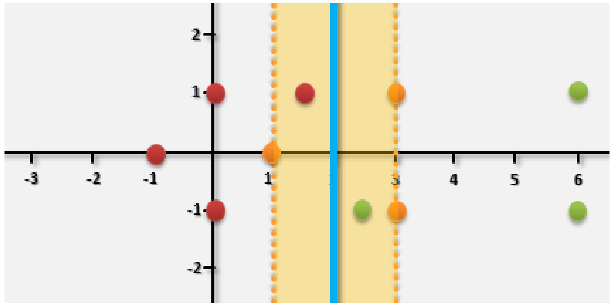
(all candidate functions derived from models and their parameters)

- **Choosing from various model approaches $h_1, \ldots, h_m$ is a different hypothesis**
- **Additionally a change in model parameters of $h_1, \ldots, h_m$ means a different hypothesis too**

‘select one function‘ that best approximates

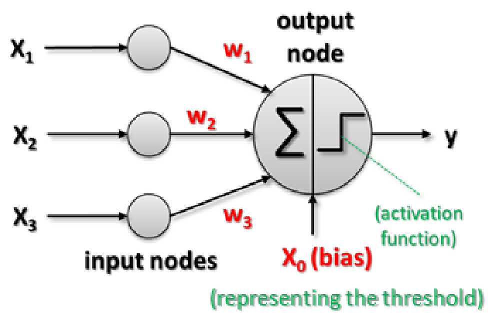
Final Hypothesis
$g \approx f$

h_1



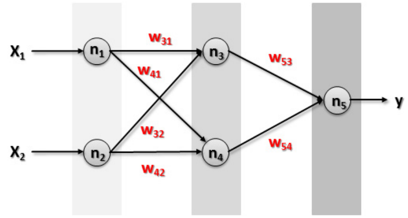
(e.g. support vector machine model)

h_2



(e.g. linear perceptron model)

h_m



(e.g. artificial neural network model)

Validation Technique – Model Selection Process – Revisited

- **Model selection is choosing (a) different types of models or (b) parameter values inside models**
- **Model selection takes advantage of the validation error in order to decide → 'pick the best'**

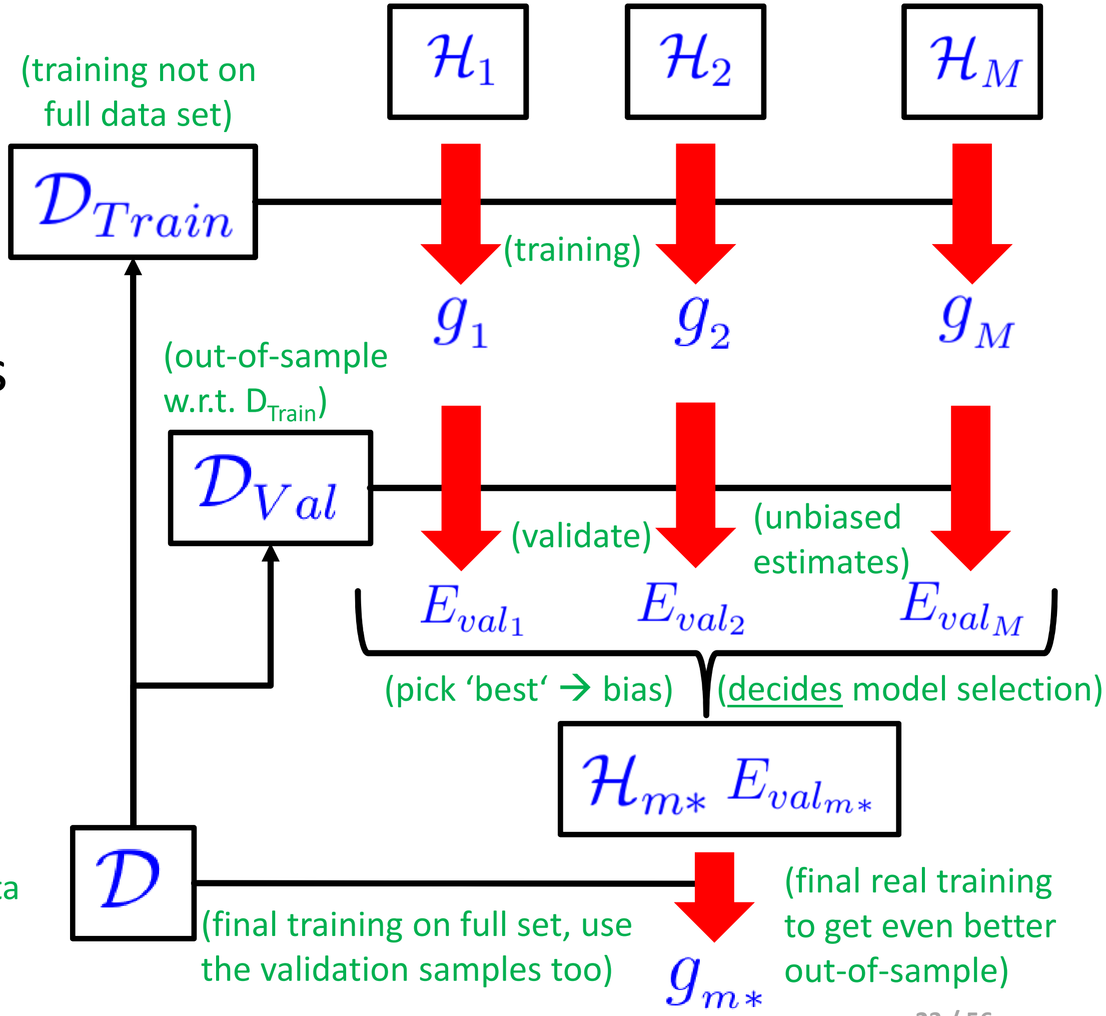
Hypothesis Set
$\mathcal{H} = \{h\}; \;\; g \in \mathcal{H}$

(set of candidate formulas across models)

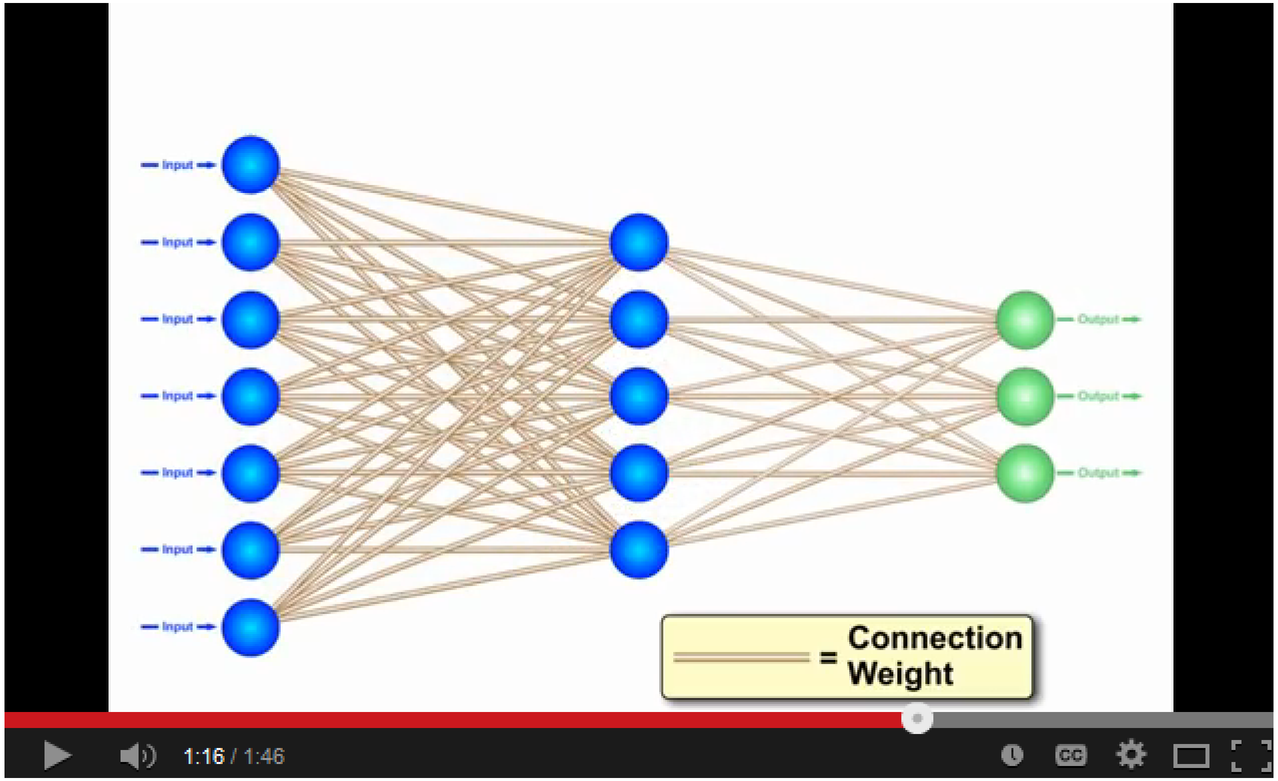
- Many different models
  Use validation error to perform select decisions
- Careful consideration:
  - 'Picked means decided' hypothesis has already bias (→ contamination)
  - Using $\mathcal{D}_{Val}$ M times

Final Hypothesis
$g_{m*} \approx f$

(test this on unseen data good, but depends on availability in practice)

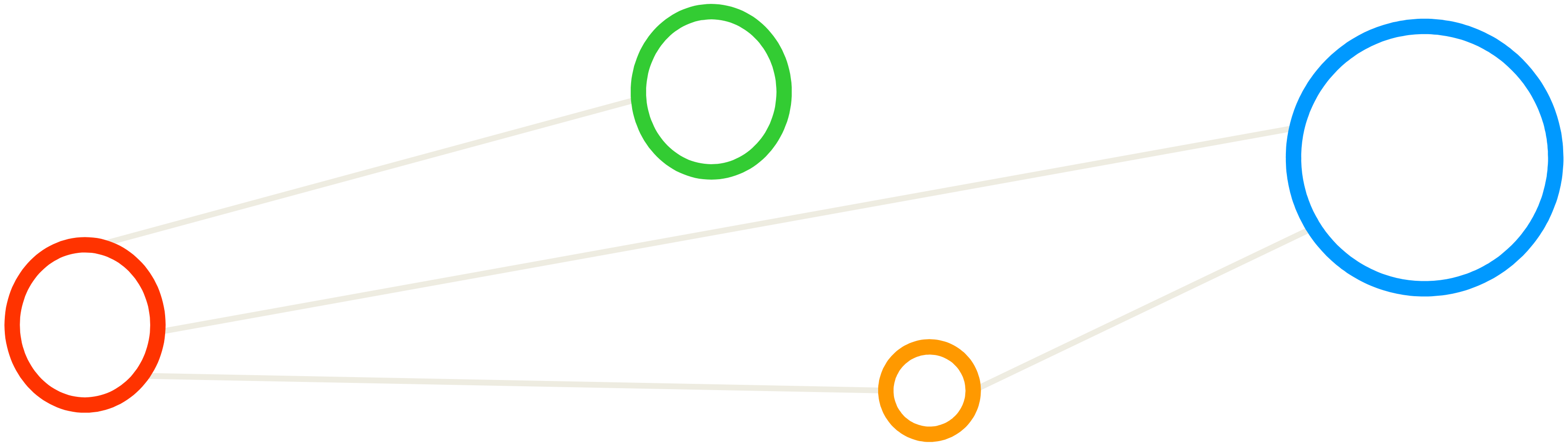


[Video] Towards Multi-Layer Perceptrons



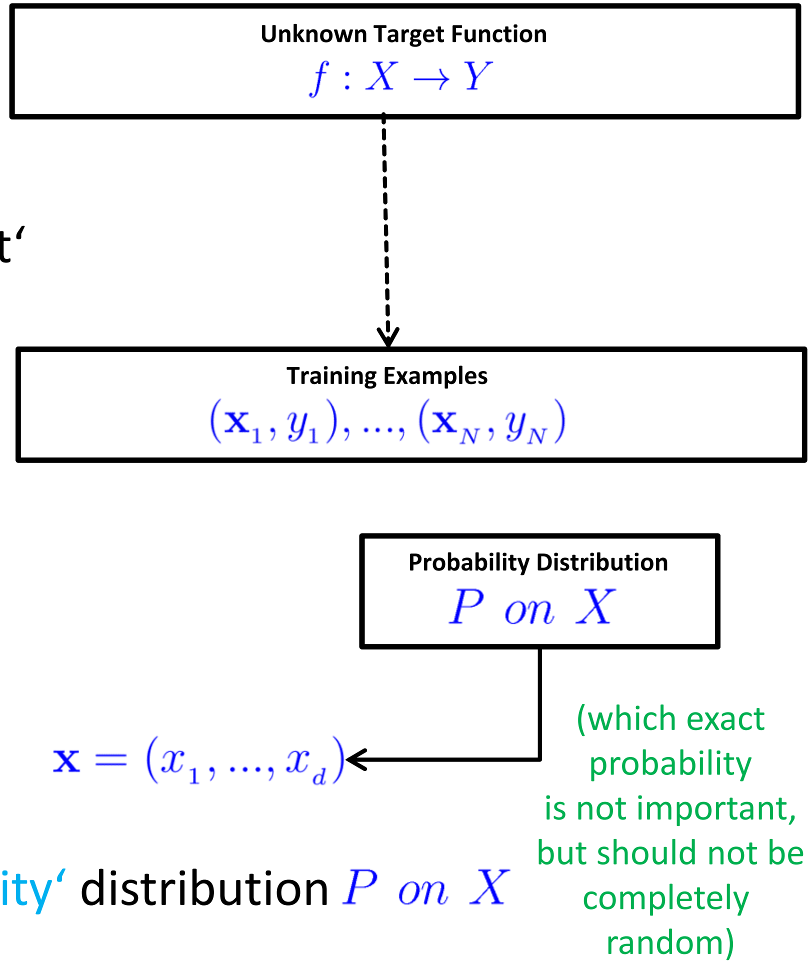
[3] YouTube Video, Neural Networks – A Simple Explanation

Learning Theory Basics



Feasibility of Learning – Probability Distribution

- Predict output from future input
  (fitting existing data is not enough)
  - In-sample '1000 points' fit well
  - Possible: Out-of-sample >= '1001 point' doesn't fit very well
  - Learning 'any target function' is not feasible (can be anything)
- Assumptions about 'future input'
  - Statement is possible to define about the data outside the in-sample data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
  - All samples (also future ones) are derived from same 'unknown probability' distribution $P \;\; on \;\; X$



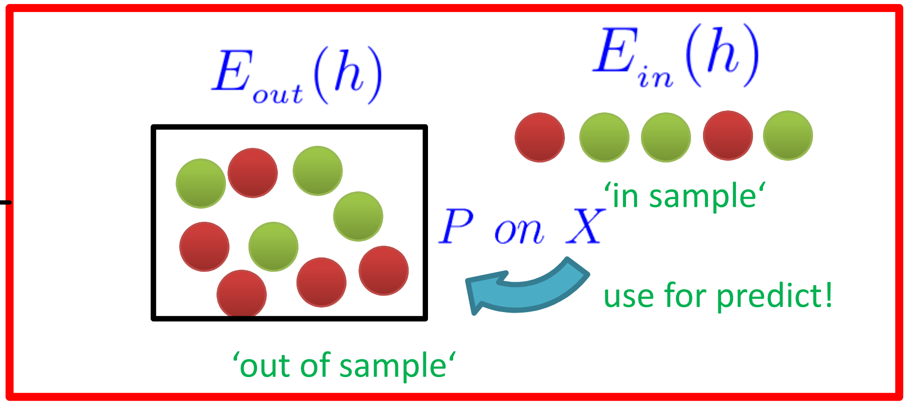
- **Statistical Learning Theory assumes an unknown probability distribution over the input space X**

Feasibility of Learning – In Sample vs. Out of Sample

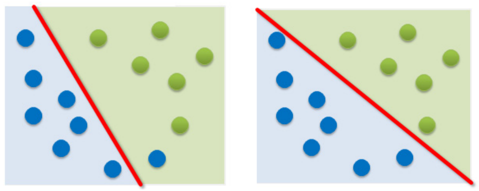
- Given ‘unknown‘ probability $P \ on \ X$
  - Given large sample N for $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$
  - There is a probability of ‘picking one point or another‘
  - ‘Error on in sample‘ is known quantity (using labelled data): $E_{in}(h)$
  - ‘Error on out of sample‘ is unknown quantity: $E_{out}(h)$
  - In-sample frequency is likely close to out-of-sample frequency

E_{in} tracks E_{out}

depend on which hypothesis h out of M different ones



$\mathcal{H} = \{h_1, ..., h_m\};$



$$E_{in}(h) \approx E_{out}(h)$$

use $E_{in}(h)$ as a proxy thus the other way around in learning

$$E_{out}(h) \approx E_{in}(h)$$

- **Statistical Learning Theory part that enables that learning is feasible in a probabilistic sense (P on X)**

Feasibility of Learning – Union Bound & Factor M

- **The union bound means that (for any countable set of m 'events') the probability that at least one of the events happens is not greater that the sum of the probabilities of the m individual 'events'**

- Assuming no overlaps in hypothesis set
  - Apply mathematical rule 'union bound'
  - (Note the usage of g instead of h, we need to visit all)

Final Hypothesis
$g \approx f$

Think if E_{in} deviates from E_{out} with more than tolerance ϵ it is a 'bad event' in order to apply union bound

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ \ <= \Pr\ [\ |\ E_{in}(h_1) - E_{out}(h_1)\ | > \epsilon$$

$$\text{or}\ \ |\ E_{in}(h_2) - E_{out}(h_2)\ | > \epsilon\ \ \ldots$$

$$\text{or}\ \ |\ E_{in}(h_M) - E_{out}(h_M)\ | > \epsilon\ \ ]$$

'visiting M different hypothesis'

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ \ <= \sum_{m=1}^{M} \Pr\ [\ |\ E_{in}(h_m) - E_{out}(h_m)\ | > \epsilon\ ]$$

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ \ <= \sum_{m=1}^{M} 2e^{-2\epsilon^2 N}$$

fixed quantity for each hypothesis obtained from Hoeffdings Inequality

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ \ <= 2Me^{-2\epsilon^2 N}$$

problematic: if M is too big we loose the link between the in-sample and out-of-sample

Feasibility of Learning – Modified Hoeffding's Inequality

- Errors in-sample $E_{in}(g)$ track errors out-of-sample $E_{out}(g)$
  - Statement is made being 'Probably Approximately Correct (PAC)'
  - Given M as number of hypothesis of hypothesis set $\mathcal{H}$
  - 'Tolerance parameter' in learning ϵ
  - Mathematically established via 'modified Hoeffdings Inequality': (original Hoeffdings Inequality doesn't apply to multiple hypothesis)

[4] Valiant, 'A Theory of the Learnable', 1984

'Approximately' 'Probably'

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

'Probability that E_{in} deviates from E_{out} by more than the tolerance Є is a small quantity depending on M and N'

- Theoretical 'Big Data' Impact → more N → better learning
  - The more samples N the more reliable will track $E_{in}(g)$ $E_{out}(g)$ well
  - (But: the 'quality of samples' also matter, not only the number of samples)

- **Statistical Learning Theory part describing the Probably Approximately Correct (PAC) learning**

Exercises – Explore Train on Test & Test on Train

- **Learning exercise to understand better the theory of generalization – don‘t do this in practice!**


```
# model training
history = model.fit(X_test, Y_test, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE, validation_split = VAL_SPLIT)

# model evaluation
score = model.evaluate(X_train, Y_train, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

MNIST Data – Testing on Training Dataset (20 Epochs)

- Testing Dataset 10000 samples now training
- Training Data 60000 samples now testing
- Number N affects training performance (was ~98%, Epochs constant)

```
26176/60000 [============>.................] - ETA: 1s
27584/60000 [============>.................] - ETA: 1s
28960/60000 [=============>................] - ETA: 1s
30336/60000 [==============>...............] - ETA: 1s
31712/60000 [==============>...............] - ETA: 1s
33056/60000 [===============>..............] - ETA: 0s
34432/60000 [================>.............] - ETA: 0s
35808/60000 [================>.............] - ETA: 0s
37184/60000 [=================>............] - ETA: 0s
38528/60000 [==================>...........] - ETA: 0s
39936/60000 [==================>...........] - ETA: 0s
41344/60000 [===================>..........] - ETA: 0s
42784/60000 [====================>.........] - ETA: 0s
44192/60000 [=====================>........] - ETA: 0s
45600/60000 [=====================>........] - ETA: 0s
47008/60000 [======================>.......] - ETA: 0s
48416/60000 [=======================>......] - ETA: 0s
49824/60000 [=======================>......] - ETA: 0s
51232/60000 [========================>.....] - ETA: 0s
52608/60000 [=========================>....] - ETA: 0s
54016/60000 [==========================>...] - ETA: 0s
55424/60000 [==========================>...] - ETA: 0s
56864/60000 [===========================>..] - ETA: 0s
58272/60000 [============================>.] - ETA: 0s
59680/60000 [============================>.] - ETA: 0s
60000/60000 [==============================] - 2s 36us/step
Using TensorFlow backend.
Test score: 0.2172071209657685
Test accuracy: 0.94875
```

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

MNIST Data – Testing on Training Dataset (200 Epochs)

- Testing Dataset 10000 samples now training
- Training Data 60000 samples now testing
- Number N affects training performance (was ~98%, Epochs 200)

```
10048/60000 [====>.........................] - ETA: 1s
11456/60000 [====>.........................] - ETA: 1s
12864/60000 [=====>........................] - ETA: 1s
14272/60000 [======>.......................] - ETA: 1s
15680/60000 [======>.......................] - ETA: 1s
17088/60000 [=======>......................] - ETA: 1s
18496/60000 [========>.....................] - ETA: 1s
19904/60000 [========>.....................] - ETA: 1s
21312/60000 [=========>....................] - ETA: 1s
22752/60000 [==========>...................] - ETA: 1s
24192/60000 [===========>..................] - ETA: 1s
25632/60000 [===========>..................] - ETA: 1s
27072/60000 [============>.................] - ETA: 1s
28512/60000 [=============>................] - ETA: 1s
29952/60000 [=============>................] - ETA: 1s
31392/60000 [==============>...............] - ETA: 1s
32832/60000 [===============>..............] - ETA: 0s
34272/60000 [================>.............] - ETA: 0s
35712/60000 [================>.............] - ETA: 0s
37152/60000 [=================>............] - ETA: 0s
38592/60000 [==================>...........] - ETA: 0s
40032/60000 [===================>..........] - ETA: 0s
41472/60000 [===================>..........] - ETA: 0s
42912/60000 [====================>.........] - ETA: 0s
44352/60000 [=====================>........] - ETA: 0s
45792/60000 [=====================>........] - ETA: 0s
47232/60000 [======================>.......] - ETA: 0s
48672/60000 [=======================>......] - ETA: 0s
50112/60000 [========================>.....] - ETA: 0s
51552/60000 [========================>.....] - ETA: 0s
53024/60000 [=========================>....] - ETA: 0s
54464/60000 [==========================>...] - ETA: 0s
55904/60000 [==========================>...] - ETA: 0s
57344/60000 [===========================>..] - ETA: 0s
58816/60000 [============================>.] - ETA: 0s
60000/60000 [==============================] - 2s 35us/step
Using TensorFlow backend.
Test score: 0.39456051169445205
Test accuracy: 0.95465
```

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

MNIST Data – Testing on Training Dataset (400 Epochs)

- Testing Dataset 10000 samples now training
- Training Data 60000 samples now testing
- Number N affects training performance (was ~98%, Epochs 400)

```
15872/60000 [======>.......................] - ETA: 1s
17312/60000 [=======>......................] - ETA: 1s
18752/60000 [========>.....................] - ETA: 1s
20160/60000 [=========>....................] - ETA: 1s
21536/60000 [=========>....................] - ETA: 1s
22976/60000 [==========>...................] - ETA: 1s
24384/60000 [===========>..................] - ETA: 1s
25792/60000 [===========>..................] - ETA: 1s
27232/60000 [============>.................] - ETA: 1s
28672/60000 [=============>................] - ETA: 1s
30112/60000 [==============>...............] - ETA: 1s
31552/60000 [==============>...............] - ETA: 1s
32960/60000 [===============>..............] - ETA: 0s
34400/60000 [================>.............] - ETA: 0s
35808/60000 [================>.............] - ETA: 0s
37248/60000 [=================>............] - ETA: 0s
38656/60000 [==================>...........] - ETA: 0s
40096/60000 [===================>..........] - ETA: 0s
41536/60000 [===================>..........] - ETA: 0s
42944/60000 [====================>.........] - ETA: 0s
44384/60000 [=====================>........] - ETA: 0s
45824/60000 [=====================>........] - ETA: 0s
47232/60000 [======================>.......] - ETA: 0s
48672/60000 [=======================>......] - ETA: 0s
50112/60000 [========================>.....] - ETA: 0s
51552/60000 [========================>.....] - ETA: 0s
52960/60000 [=========================>....] - ETA: 0s
54368/60000 [==========================>...] - ETA: 0s
55808/60000 [==========================>...] - ETA: 0s
57216/60000 [===========================>..] - ETA: 0s
58624/60000 [============================>.] - ETA: 0s
60000/60000 [==============================] - 2s 35us/step
Using TensorFlow backend.
Test score: 0.46486433204362193
Test accuracy: 0.9536333333333333
```

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ > \epsilon\ ]\ \ <=\ \ 2Me^{-2\epsilon^2 N}$$

MNIST Data – Testing on Training Dataset (800 Epochs)

- Testing Dataset 10000 samples now training
- Training Data 60000 samples now testing
- Number N affects training performance (was ~98%, Epochs 800)

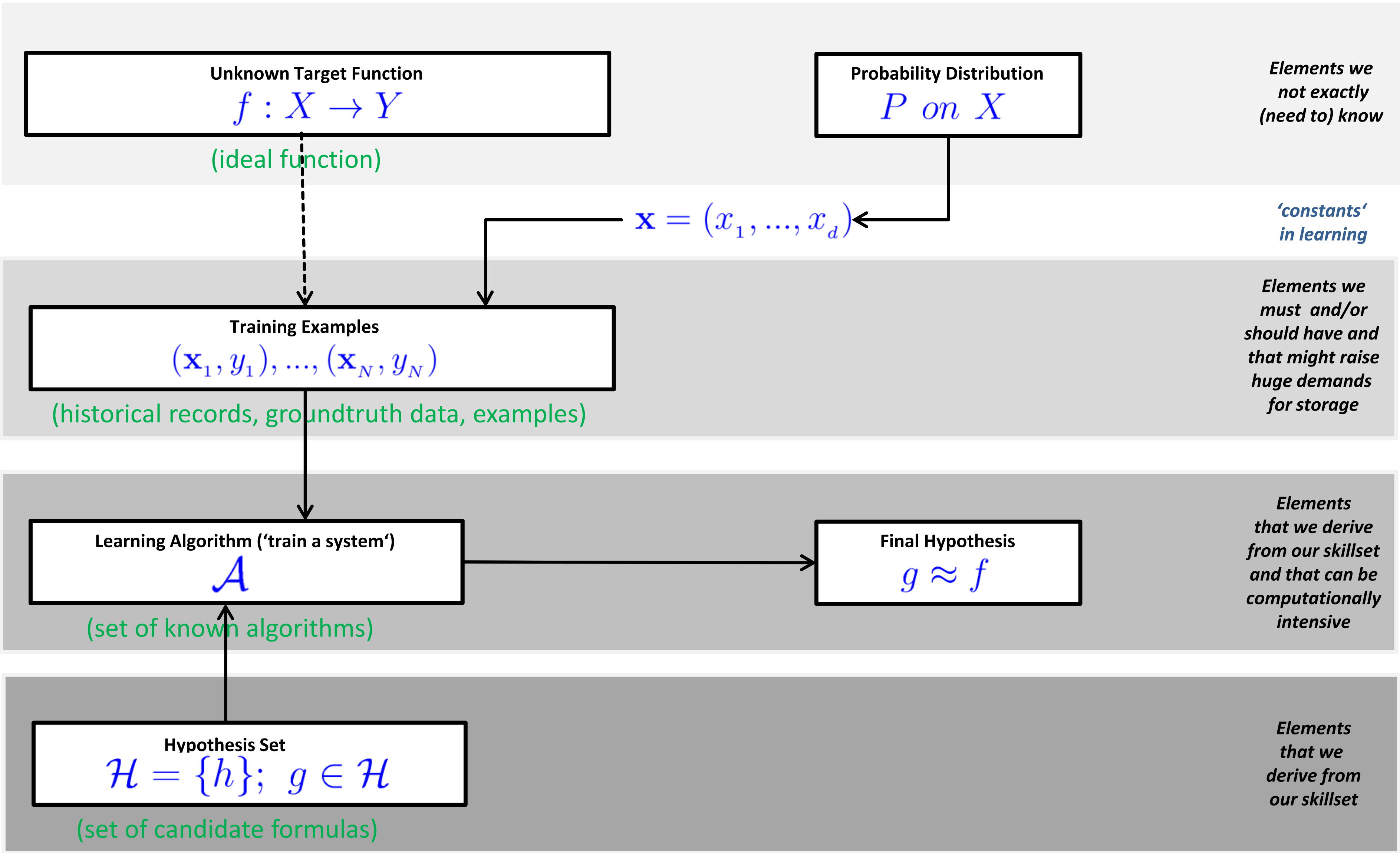
```
Epoch 800/800

 128/8000 [............................] - ETA: 0s - loss: 0.0126 - acc: 0.9922
2048/8000 [======>.....................] - ETA: 0s - loss: 0.0106 - acc: 0.9980
3968/8000 [=============>..............] - ETA: 0s - loss: 0.0148 - acc: 0.9972
5888/8000 [====================>.......] - ETA: 0s - loss: 0.0140 - acc: 0.9976
7808/8000 [============================>.] - ETA: 0s - loss: 0.0111 - acc: 0.9980
8000/8000 [==============================] - 0s 30us/step - loss: 0.0108 - acc: 0.9980 - val_loss: 0.2866 - val_acc: 0.9725
```

```
19712/60000 [========>.....................] - ETA: 1s
21088/60000 [=========>....................] - ETA: 1s
22464/60000 [==========>...................] - ETA: 1s
23840/60000 [==========>...................] - ETA: 1s
25216/60000 [===========>..................] - ETA: 1s
26624/60000 [============>.................] - ETA: 1s
28032/60000 [=============>................] - ETA: 1s
29440/60000 [=============>................] - ETA: 1s
30848/60000 [==============>...............] - ETA: 1s
32256/60000 [===============>..............] - ETA: 1s
33632/60000 [===============>..............] - ETA: 0s
35008/60000 [================>.............] - ETA: 0s
36416/60000 [=================>............] - ETA: 0s
37824/60000 [=================>............] - ETA: 0s
39200/60000 [==================>...........] - ETA: 0s
40608/60000 [===================>..........] - ETA: 0s
41984/60000 [===================>..........] - ETA: 0s
43392/60000 [====================>.........] - ETA: 0s
44768/60000 [=====================>........] - ETA: 0s
46144/60000 [======================>.......] - ETA: 0s
47520/60000 [======================>.......] - ETA: 0s
48928/60000 [=======================>......] - ETA: 0s
50336/60000 [========================>.....] - ETA: 0s
51744/60000 [========================>.....] - ETA: 0s
53120/60000 [=========================>....] - ETA: 0s
54528/60000 [==========================>...] - ETA: 0s
55936/60000 [==========================>...] - ETA: 0s
57312/60000 [===========================>..] - ETA: 0s
58720/60000 [============================>.] - ETA: 0s
Using TensorFlow backend.
Test score: 0.5250241714548276
Test accuracy: 0.9528833333333333
```

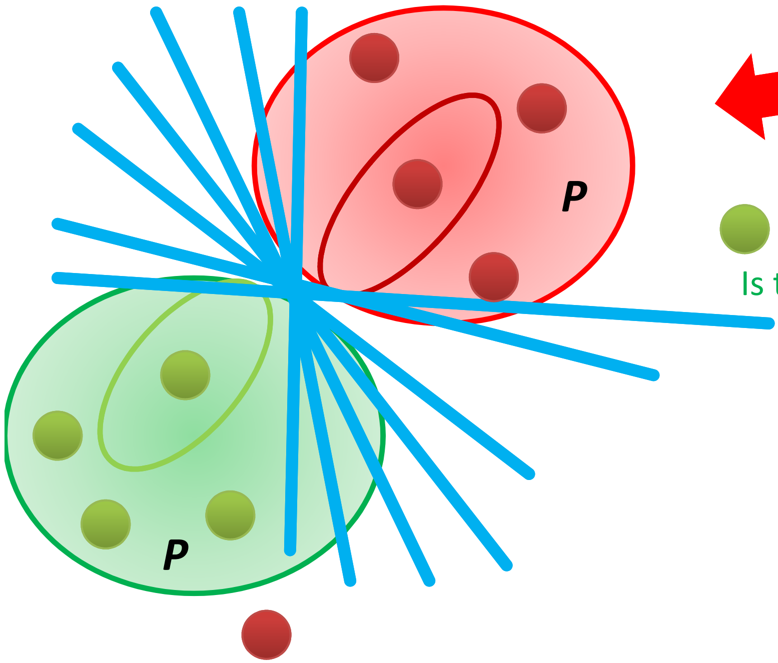
$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

Mathematical Building Blocks (4)



Mathematical Building Blocks (4) – Our Linear Example

(infinite M decision boundaries depending on f)



Probability Distribution
$P \ on \ X$

$\mathbf{x} = (x_1, \ldots, x_d)$

Is this point very likely from the same distribution or just noise?

We assume future points are taken from the same probability distribution as those that we have in our training examples

Training Examples
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Is this point very likely from the same distribution or just noise?

(we help here with the assumption for the samples)

(we do not solve the M problem here)

$$\Pr \left[\ | \ E_{in}(g) - E_{out}(g) \ | > \epsilon \ \right] \ <= \ 2Me^{-2\epsilon^2 N}$$

(counter example would be for instance a random number generator, impossible to learn this!)

Statistical Learning Theory – Error Measure & Noisy Targets

- Question: How can we learn a function from (noisy) data?
- ‘Error measures‘ to quantify our progress, the goal is: $h \approx f$
  - Often user-defined, if not often ‘squared error‘:

$$e(h(\mathbf{x}), f(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$$

Error Measure
α

  - E.g. ‘point-wise error measure‘

(e.g. think movie rated now and in 10 years from now)

- ‘(Noisy) Target function‘ is not a (deterministic) function
  - Getting with ‘same x in‘ the ‘same y out‘ is not always given in practice
  - Problem: ‘Noise‘ in the data that hinders us from learning
  - Idea: Use a ‘target distribution‘ instead of ‘target function‘
  - E.g. credit approval (yes/no)

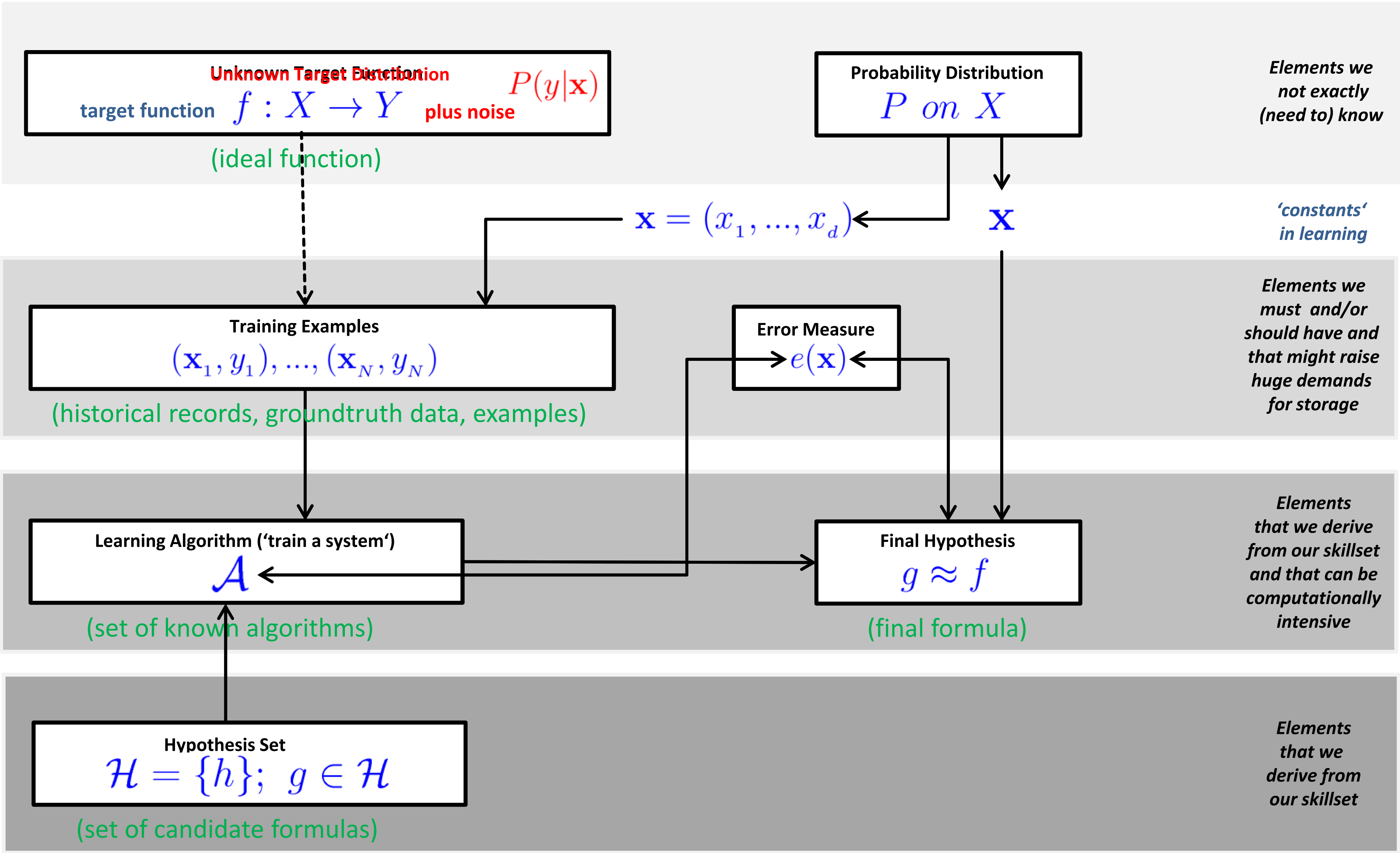
Unknown Target Distribution $P(y|\mathbf{x})$

target function $f : X \rightarrow Y$ plus noise

(ideal function)

- **Statistical Learning Theory refines the learning problem of learning an unknown target distribution**

Mathematical Building Blocks (5)



Mathematical Building Blocks (5) – Our Linear Example

- Iterative Method using (labelled) training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(one point at a time is picked)

1. Pick one misclassified training point where:

$$sign(\mathbf{w}^T \mathbf{x}_n) \neq y_n$$

Error Measure
α

2. Update the weight vector:

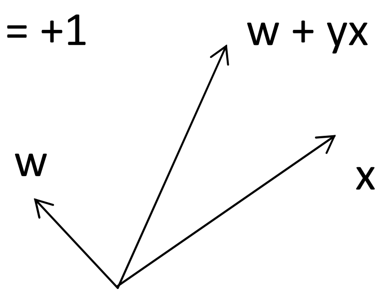
$$\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n$$

(y_n is either +1 or -1)

(a) adding a vector or
(b) subtracting a vector

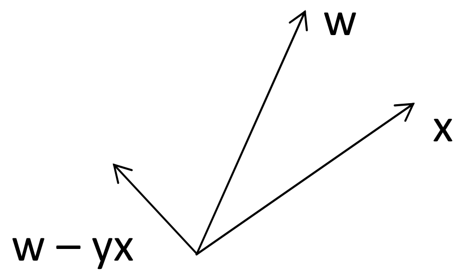
y = +1

(a)



y = -1

(b)



- Terminates when there are no misclassified points

Error Measure
α

(converges only with linearly seperable data)

Training and Testing – Influence on Learning

- Mathematical notations
  - Testing follows: $\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 2\quad e^{-2\epsilon^2 N}$
    (hypothesis clear)
  - Training follows: $\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$
    (hypothesis search) (e.g. student exam training on examples to get E_{in} ‚down', then test via exam)
- Practice on 'training examples'
  - Create two disjoint datasets
  - One used for training only
    (aka training set)
  - Another used for testing only
    (aka test set)

Training Examples

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

- Training & Testing are different phases in the learning process
  - Concrete number of samples in each set often influences learning

Theory of Generalization – Initial Generalization & Limits

- Learning is feasible in a probabilistic sense
  - Reported final hypothesis – using a 'generalization window' on $E_{out}(g)$
  - Expecting 'out of sample performance' tracks 'in sample performance'
  - Approach: $E_{in}(g)$ acts as a 'proxy' for $E_{out}(g)$

$$E_{out}(g) \approx E_{in}(g)$$

This is not full learning – rather 'good generalization' since the quantity $E_{out}(g)$ is an unknown quantity

- Reasoning
  - Above condition is not the final hypothesis condition:
  - More similiar like $E_{out}(g)$ approximates 0 (out of sample error is close to 0 if approximating f)
  - $E_{out}(g)$ measures how far away the value is from the 'target function'
  - Problematic because $E_{out}(g)$ is an unknown quantity (cannot be used…)
  - The learning process thus requires 'two general core building blocks'

Final Hypothesis

$$\boxed{g \approx f}$$

Theory of Generalization – Learning Process Reviewed

- 'Learning Well'
  - Two core building blocks that achieve $E_{out}(g)$ approximates 0
- First core building block
  - Theoretical result using Hoeffdings Inequality $E_{out}(g) \approx E_{in}(g)$
  - Using $E_{out}(g)$ directly is not possible – it is an unknown quantity
- Second core building block

  (try to get the 'in-sample' error lower)
  - Practical result using tools & techniques to get $E_{in}(g) \approx 0$
  - e.g. linear models with the Perceptron Learning Algorithm (PLA)
  - Using $E_{in}(g)$ is possible – it is a known quantity – 'so lets get it small'
  - Lessons learned from practice: in many situations 'close to 0' impossible
  - E.g. remote sensing images use case of land cover classification

- **Full learning means that we can make sure that $E_{out}(g)$ is close enough to $E_{in}(g)$ [from theory]**
- **Full learning means that we can make sure that $E_{in}(g)$ is small enough [from practical techniques]**

Complexity of the Hypothesis Set – Infinite Spaces Problem

$$\Pr \left[\; | \; E_{in}(g) - E_{out}(g) \; | > \epsilon \; \right] \; <= \; 2Me^{-2\epsilon^2 N}$$

theory helps to find a way to deal with infinite M hypothesis spaces

- Tradeoff & Review
  - Tradeoff between Є, M, and the 'complexity of the hypothesis space H'
  - Contribution of detailed learning theory is to 'understand factor M'
- M Elements of the hypothesis set $\mathcal{H}$ M elements in H here
  - Ok if N gets big, but problematic if M gets big → bound gets meaningless
  - E.g. classification models like perceptron, support vector machines, etc.
  - Challenge: those classification models have continous parameters
  - Consequence: those classification models have infinite hypothesis spaces
  - Aproach: despite their size, the models still have limited expressive power

- **Many elements of the hypothesis set H have continous parameter with infinite M hypothesis spaces**

Factor M from the Union Bound & Hypothesis Overlaps

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ] \ <= \Pr\ [\ |\ E_{in}(h_1) - E_{out}(h_1)\ | > \epsilon$$
$$\text{or}\ \ |\ E_{in}(h_2) - E_{out}(h_2)\ | > \epsilon\ \ldots$$
$$\text{or}\ \ |\ E_{in}(h_M) - E_{out}(h_M)\ | > \epsilon\ \ ]$$

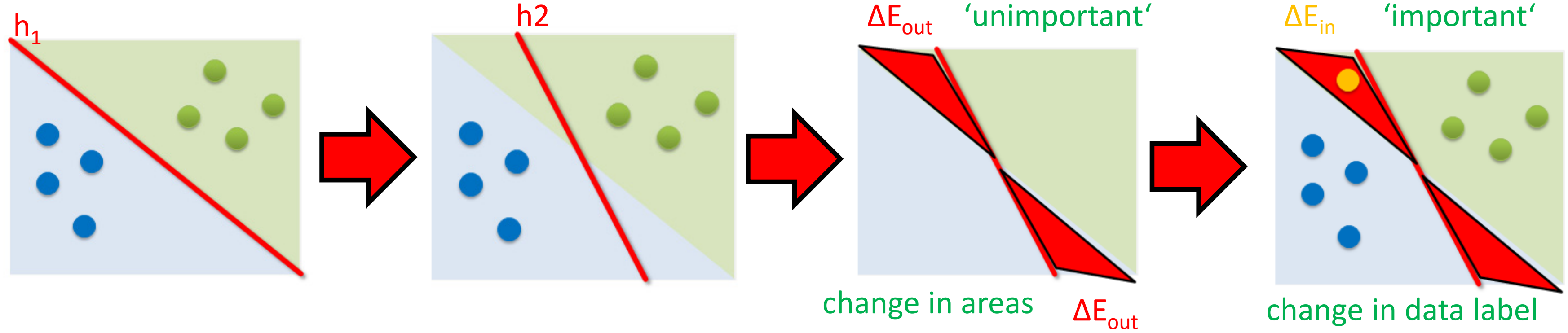
assumes no overlaps, all probabilities happen disjointly

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ] \ <= \ 2Me^{-2\epsilon^2 N}$$ takes no overlaps of M hypothesis into account

- Union bound is a 'poor bound', ignores correlation between h
  - Overlaps are common: the interest is shifted to data points changing label

$$|\ E_{in}(h_1) - E_{out}(h_1)\ | \approx \ |\ E_{in}(h_2) - E_{out}(h_2)\ |$$

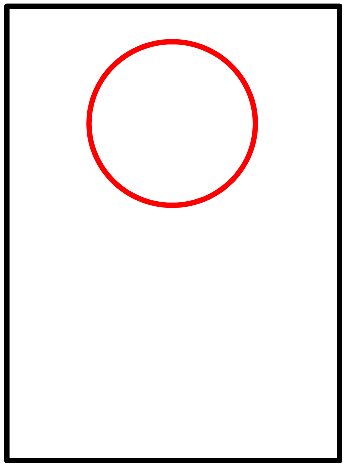
(at least very often, indicator to reduce M)



- **Statistical Learning Theory provides a quantity able to characterize the overlaps for a better bound**

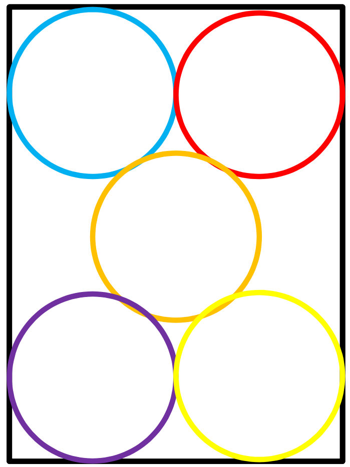
Replacing M & Large Overlaps

(Hoeffding Inequality)



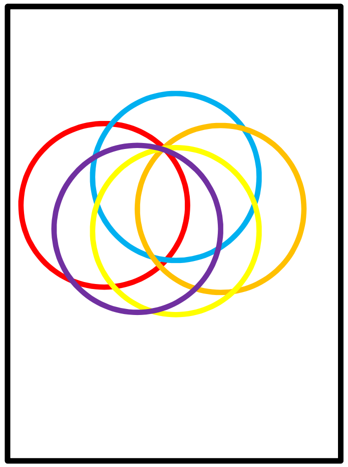
(valid for 1 hypothesis)

(Union Bound)



(valid for M hypothesis, worst case)

(towards Vapnik Chervonenkis Bound)



(valid for m (N) as growth function)

- Characterizing the overlaps is the idea of a 'growth function'
  - Number of dichotomies: Number of hypothesis but on finite number N of points $\mathbf{m}_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)|$
  - Much redundancy: Many hypothesis will reports the same dichotomies

- **The mathematical proofs that $m_H(N)$ can replace M is a key part of the theory of generalization**

Complexity of the Hypothesis Set – VC Inequality

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$

$$\mathbf{m}_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)|$$

- Vapnik-Chervonenkis (VC) Inequality
  - Result of mathematical proof when replacing M with growth function m
  - 2N of growth function to have another sample (2 x $E_{in}(h)$, no $E_{out}(h)$)

$$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ | > \epsilon\ ]\ <=\ 4m_{\mathcal{H}}(2N)e^{-1/8\epsilon^2 N}$$

(characterization of generalization)

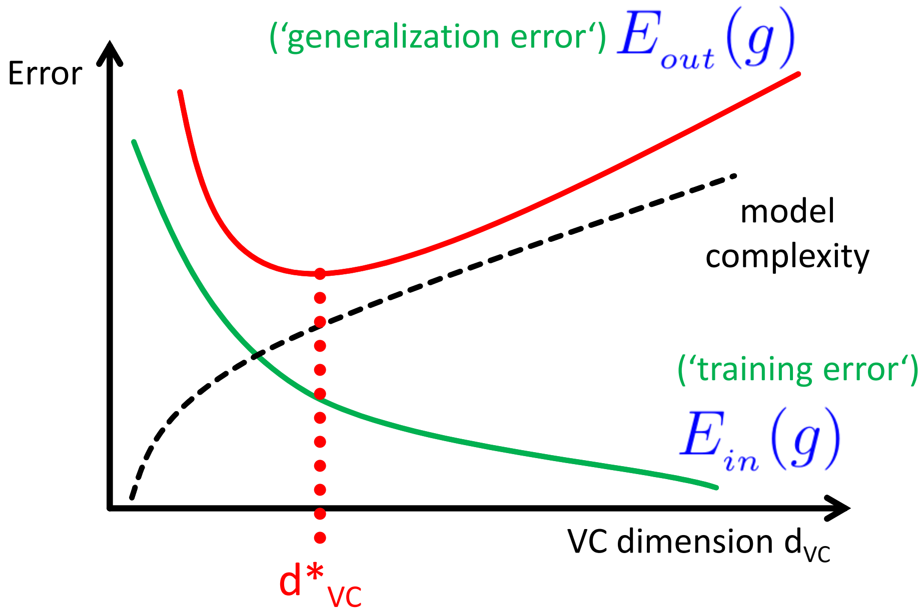
  - In Short – finally : We are able to learn and can generalize 'ouf-of-sample'

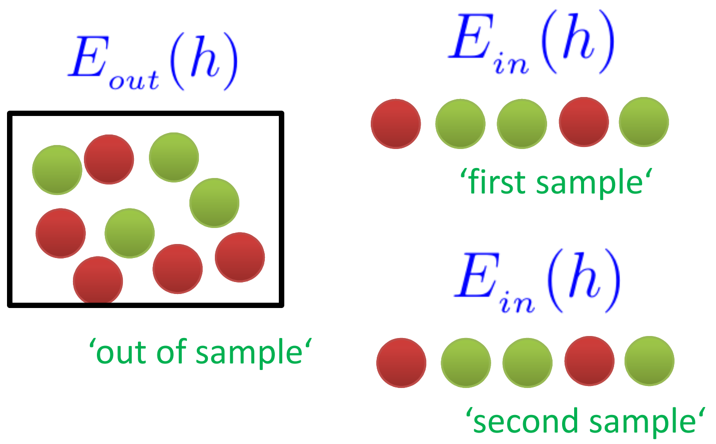
- **The Vapnik-Chervonenkis Inequality is the most important result in machine learning theory**
- **The mathematial proof brings us that M can be replaced by growth function (no infinity anymore)**
- **The growth function is dependent on the amount of data N that we have in a learning problem**

Complexity of the Hypothesis Set – VC Dimension

- Vapnik-Chervonenkis (VC) Dimension over instance space X
  - VC dimension gets a 'generalization bound' on all possible target functions

Issue: unknown to 'compute' – VC solved this using the growth function on different samples





idea: 'first sample' frequency close to 'second sample' frequency

- **Complexity of Hypothesis set H can be measured by the Vapnik-Chervonenkis (VC) Dimension d_{VC}**
- **Ignoring the model complexity d_{VC} leads to situations where $E_{in}(g)$ gets down and $E_{out}(g)$ gets up**

Different Models – Hypothesis Set & Model Capacity

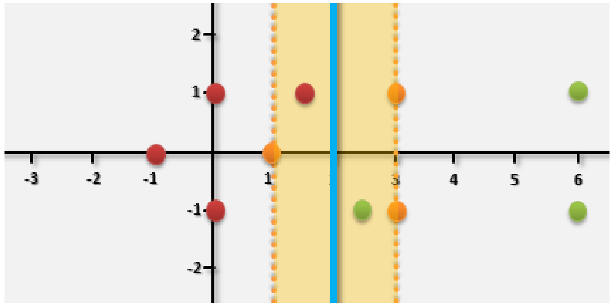
Hypothesis Set

$\mathcal{H} = \{h\}; \ g \in \mathcal{H}$

$$\mathcal{H} = \{h_1, ..., h_m\};$$

(all candidate functions
derived from models
and their parameters)

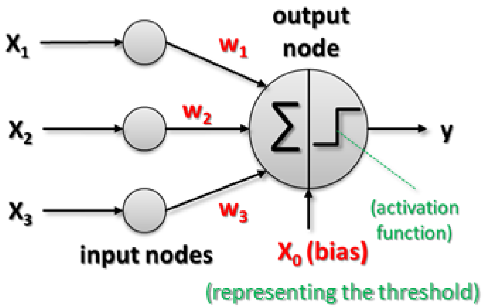
h_1



(e.g. support vector machine model)

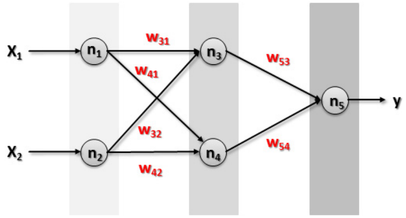
- **Choosing from various model approaches h_1, …, h_m is a different hypothesis**
- **Additionally a change in model parameters of h_1, …, h_m means a different hypothesis too**
- **The model capacity characterized by the VC Dimension helps in choosing models**
- **Occam‘s Razor rule of thumb: ‘simpler model better‘ in any learning problem, not too simple!**

h_2



(e.g. linear perceptron model)

h_m



(e.g. artificial neural network model)

‘select one function‘
that best approximates

Final Hypothesis

$g \approx f$

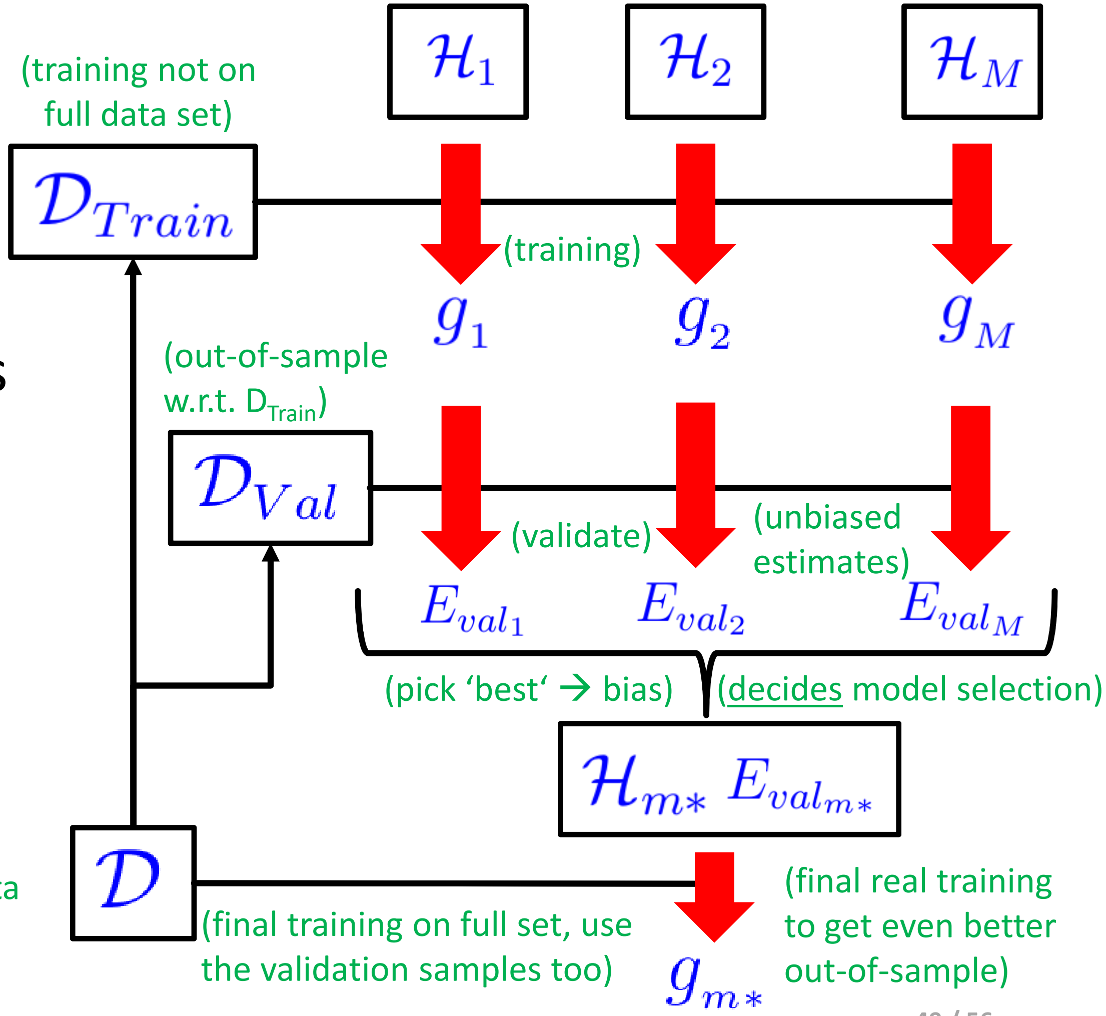
Validation Technique – Model Selection Process – Revisited

- **Model selection is choosing (a) different types of models or (b) parameter values inside models**
- **Model selection takes advantage of the validation error in order to decide → 'pick the best'**

Hypothesis Set
$\mathcal{H} = \{h\}; \;\; g \in \mathcal{H}$

(set of candidate formulas across models)

- Many different models
  Use validation error to perform select decisions
- Careful consideration:
  - 'Picked means decided' hypothesis has already bias (→ contamination)
  - Using $\mathcal{D}_{Val}$ M times



Final Hypothesis
$g_{m*} \approx f$

(test this on unseen data good, but depends on availability in practice)

AUDIENCE QUESTION

What could happen to Overfitting and we try to stop it?

Prevent Overfitting for better 'ouf-of-sample' generalization

[5] Stop Overfitting, YouTube

Appendix A – SSH Commands JURECA

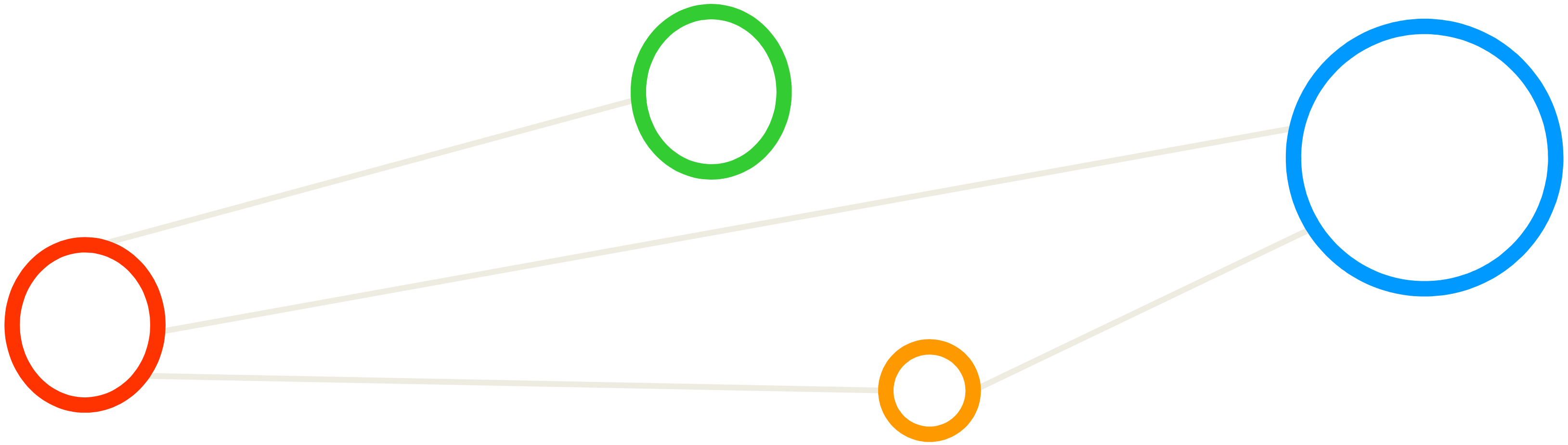
Appendix A – SSH Commands JURECA

- salloc --gres=gpu:4 --partition=gpus --nodes=1 --account=training1904 --time=00:30:00 --reservation=prace_ml_gpus_tue

- module --force purge;
  module use /usr/local/software/jureca/OtherStages
  module load Stages/Devel-2018b GCCcore/.7.3.0
  module load TensorFlow/1.12.0-GPU-Python-3.6.6
  module load Keras/2.2.4-GPU-Python-3.6.6

- srun python PYTHONSCRIPTNAME

Lecture Bibliography



Lecture Bibliography

- [1] An Introduction to Statistical Learning with Applications in R,
  Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [2] Wikipedia on 'Statistical Learning Theory',
  Online: http://en.wikipedia.org/wiki/Statistical_learning_theory
- [3] YouTube Video, 'Decision Trees',
  Online: http://www.youtube.com/watch?v=DCTUtPJn42s
- [4] Leslie G. Valiant, '*A Theory of the Learnable*', Communications of the ACM 27(11):1134–1142, 1984,
  Online: https://people.mpi-inf.mpg.de/~mehlhorn/SeminarEvolvability/ValiantLearnable.pdf
- [5] Udacity, 'Overfitting',
  Online: https://www.youtube.com/watch?v=CxAxRCv9WoA

- Acknowledgements and more Information: Yaser Abu-Mostafa, Caltech Lecture series, YouTube

Slides Available at http://www.morrisriedel.de/teaching