CE9010: Introduction to Data Science

Lecture 7: Developing Data Science Projects

Semester 2 2017/18

Xavier Bresson

School of Computer Science and Engineering
Data Science and AI Research Centre
Nanyang Technological University (NTU), Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- Conclusion

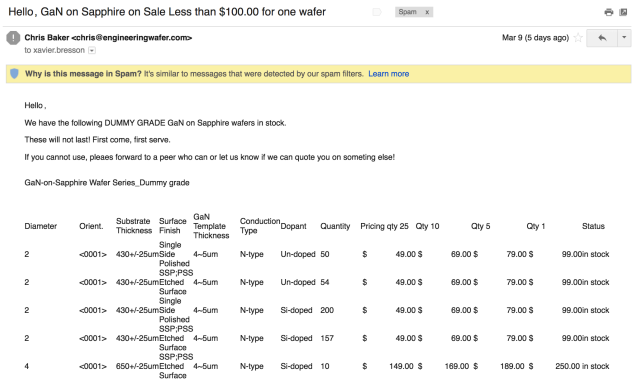
Outline

- **Strategy for developing a data analysis system**
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- Conclusion

Strategy

- How to develop a data analysis system?
  - Guidelines and best practices

    Example: How to build a spam classifier to distinguish between spam and non-spam emails?



Hello, GaN on Sapphire on Sale Less than $100.00 for one wafer

Spam x

Chris Baker <chris@engineeringwafer.com> Mar 9 (5 days ago)

to xavier.bresson

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

Hello,

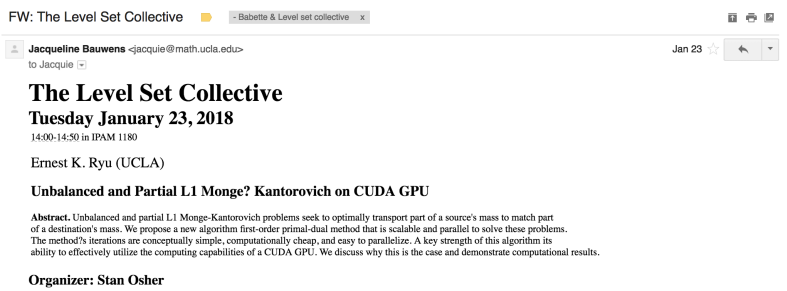
We have the following DUMMY GRADE GaN on Sapphire wafers in stock.

These will not last! First come, first serve.

If you cannot use, pleaes forward to a peer who can or let us know if we can quote you on someting else!

GaN-on-Sapphire Wafer Series_Dummy grade

Diameter	Orient.	Substrate Thickness	Surface Finish	GaN Template Thickness	Conduction Type	Dopant	Quantity	Pricing qty 25	Qty 10	Qty 5	Qty 1	Status
2	<0001>	430+/-25um	Single Side Polished SSP;PSS	4–5um	N-type	Un-doped	50	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Etched Surface Single	4–5um	N-type	Un-doped	54	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Side Polished SSP;PSS	4–5um	N-type	Si-doped	200	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Etched Surface SSP;PSS	4–5um	N-type	Si-doped	157	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
4	<0001>	650+/-25um	Etched Surface	4–5um	N-type	Si-doped	10	$ 149.00	$ 169.00	$ 189.00	$ 250.00	in stock



FW: The Level Set Collective - Babette & Level set collective x

Jacqueline Bauwens <jacquie@math.ucla.edu> Jan 23

to Jacquie

The Level Set Collective

Tuesday January 23, 2018

14:00-14:50 in IPAM 1180

Ernest K. Ryu (UCLA)

Unbalanced and Partial L1 Monge? Kantorovich on CUDA GPU

Abstract. Unbalanced and partial L1 Monge-Kantorovich problems seek to optimally transport part of a source's mass to match part of a destination's mass. We propose a new algorithm first-order primal-dual method that is scalable and parallel to solve these problems. The method?s iterations are conceptually simple, computationally cheap, and easy to parallelize. A key strength of this algorithm its ability to effectively utilize the computing capabilities of a CUDA GPU. We discuss why this is the case and demonstrate computational results.

Organizer: Stan Osher

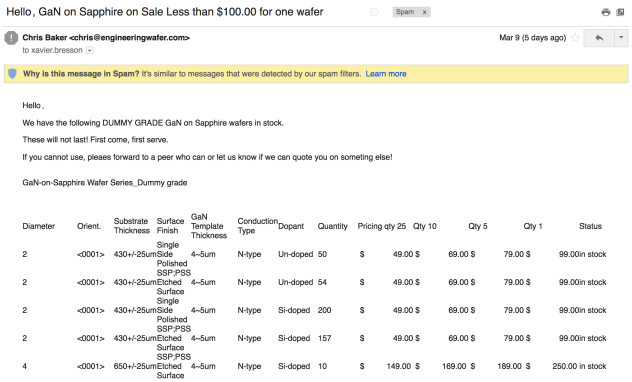
- Step 1: Identify the task and the class of learning techniques.

  For this example, we build a classifier using supervised learning.

Data representation

- Step 2: What data representation (features/variables)?
  - Develop hand-crafted features (domain expertise).
  - Learn the features from the data (end-to-end systems).

For this example, the features of the email can be hand-crafted with 100 words indicating of spams and non-spams like: deal, buy, discount,... for spams and my name, chit-chat words, etc for non-spams.



Hello, GaN on Sapphire on Sale Less than $100.00 for one wafer

Chris Baker <chris@engineeringwafer.com> Mar 9 (5 days ago)

to xavier.bresson

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

Hello ,

We have the following DUMMY GRADE GaN on Sapphire wafers in stock.

These will not last! First come, first serve.

If you cannot use, pleaes forward to a peer who can or let us know if we can quote you on someting else!

GaN-on-Sapphire Wafer Series_Dummy grade

Diameter	Orient.	Substrate Thickness	Surface Finish	GaN Template Thickness	Conduction Type	Dopant	Quantity	Pricing qty 25	Qty 10	Qty 5	Qty 1	Status
2	<0001>	430+/-25um	Single Side Polished SSP-PSS	4~5um	N-type	Un-doped	50	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Etched Surface	4~5um	N-type	Un-doped	54	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Single Side Polished SSP-PSS	4~5um	N-type	Si-doped	200	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
2	<0001>	430+/-25um	Etched Surface	4~5um	N-type	Si-doped	157	$ 49.00	$ 69.00	$ 79.00	$ 99.00	in stock
4	<0001>	650+/-25um	Etched Surface	4~5um	N-type	Si-doped	10	$ 149.00	$ 169.00	$ 189.00	$ 250.00	in stock

Email (data)

Representation of an email $\Rightarrow$

$$x = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{xavier} \\ \text{deal} \\ \text{discount} \\ \cdots \end{matrix} \quad \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \begin{matrix} \text{100 elements} \\ x \in \mathbb{R}^{100} \\ \dim(x)=100 \end{matrix}$$

$$x_{(j)} = \begin{cases} \text{nb of words } j \text{ appearing in the email} \\ 0 \text{ otherwise} \end{cases}$$

Design optimal classifier

- Step 3: How to make the spam classifier with the lowest possible error?
  - Collect (lots of) training data.
  - Select a smaller set of hand-crafted features.
  - Get additional hand-crafted features.
  - Increase learning capacity with polynomial features (x^2,x^3,...), neural networks.
  - Increase/decrease regularization

- How to select the number of features, of polynomial terms, the regularization constant, other hyper-parameters of the learning systems? Cross-validation technique.

Outline

- Strategy for developing a data analysis system
- **Cross-validation for hyper-parameter estimation**
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- Conclusion

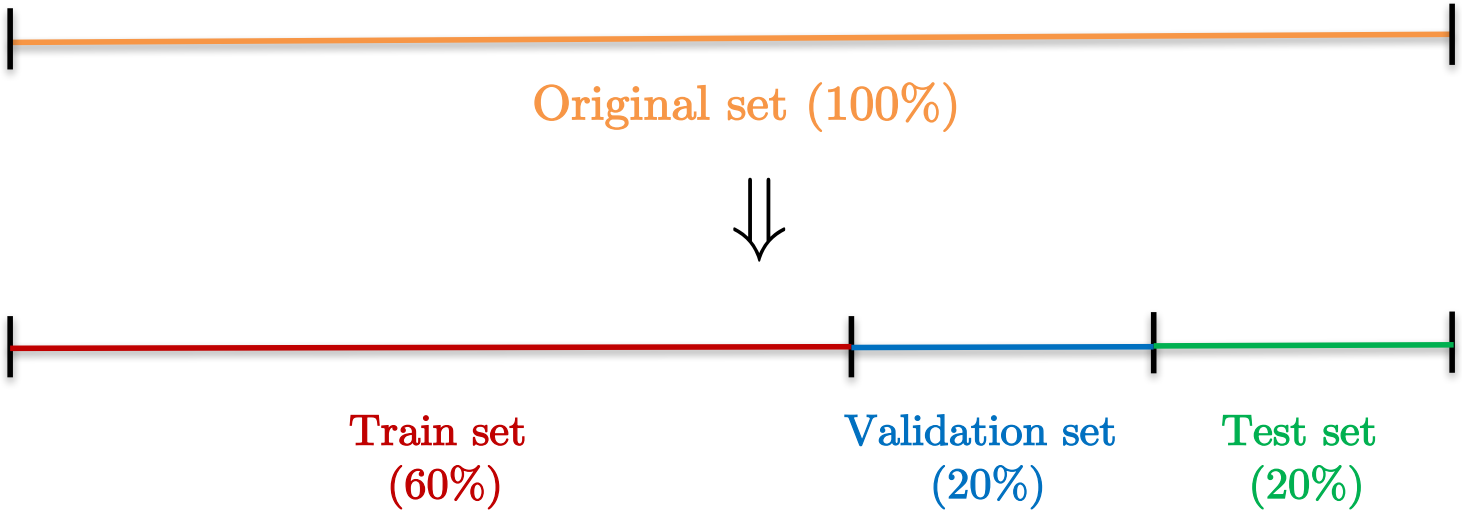
Cross-validation

- Terminology:
  - Parameters of learning algorithms are the variables that can be computed by any gradient-based techniques (gradient descent). Example: Parameters w in supervised regression and classification.
  - Hyper-parameters of learning algorithms are the variable that cannot be computed by gradient-based techniques (but by cross-validation). Examples: Degree d of polynomial terms, regularization constant λ, number of hidden layers and neurons in neural networks.
  - Hyper-parameter selections are also called model selection problems.
- Cross-validation is a technique for estimating the values of the hyper-parameters of the learning system.

Train/validation/test sets

- We split the original set into 3 sets:
  - A train set
  - A validation set (a.k.a. cross-validation set)
  - A test set

- Example:



Train/validation/test sets

- We define 3 losses (for each dataset):

  Remember the loss is a measure of fitness between the predictive model and the data. If the fit is almost perfect then the loss is close to zero.

nb of training data

$$L_{\text{Train}}(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(p_w(x_i), y_i)$$

nb of validation data

$$L_{\text{Val}}(w) = \frac{1}{n_v} \sum_{i=1}^{n_v} \ell(p_w(x_i^v), y_i^v)$$

nb of test data

$$L_{\text{Test}}(w) = \frac{1}{n_t} \sum_{i=1}^{n_t} \ell(p_w(x_i^t), y_i^t)$$

Train set (60%)	Validation set (20%)	Test set (20%)
$(x_1, y_1), ..., (x_n, y_n)$	$(x_1^v, y_1^v), ..., (x_n^v, y_n^v)$	$(x_1^t, y_1^t), ..., (x_n^t, y_n^t)$

Why the validation set?

- The validation test is essential to avoid the poor generalization problem (bad predictions for new data).

- If the validation test is not used to evaluate the prediction performances on the hyper-parameters, then the learning model will over-fit the test data and will not be able to generalize to new data.

- Example: Let us consider the selection of the polynomial degree d for the prediction function:

polynomial degree d

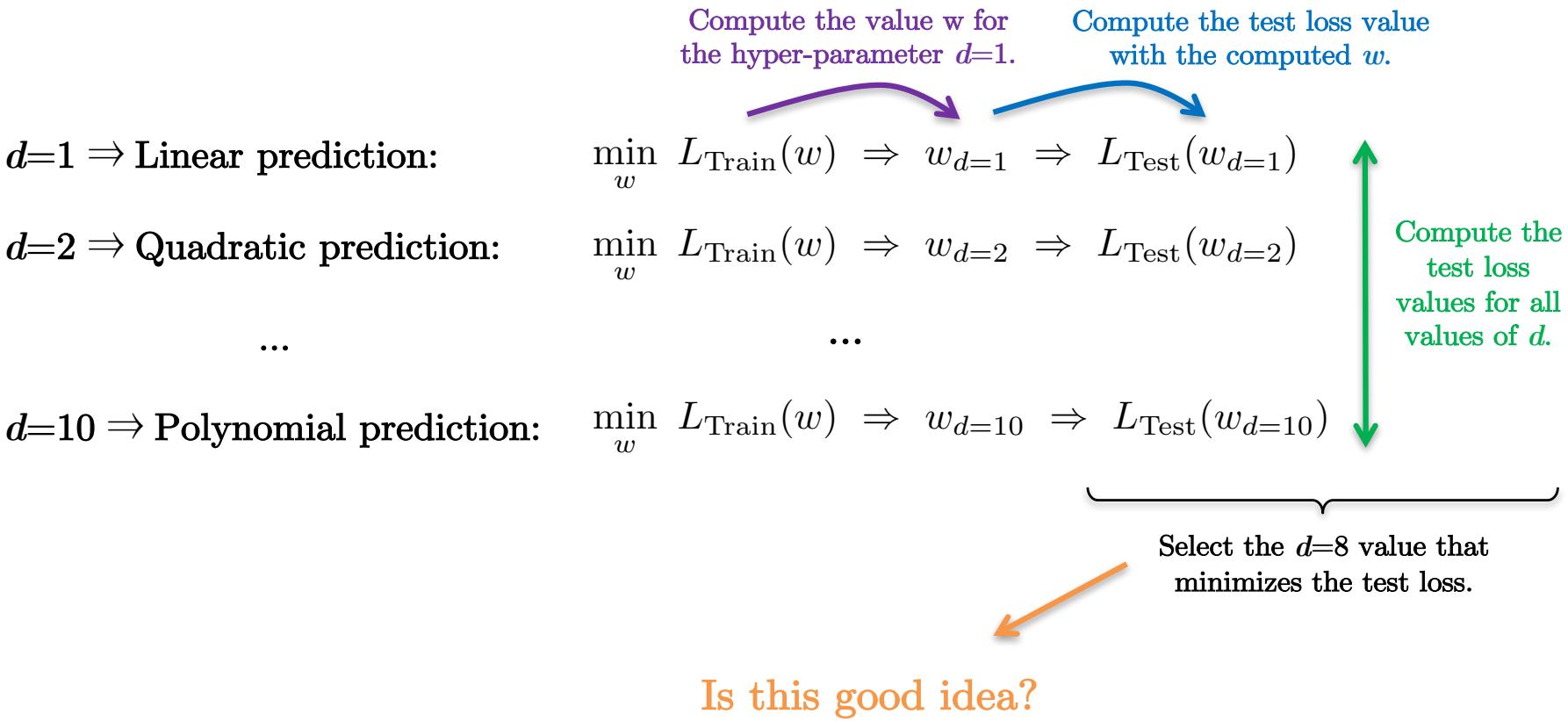
$$p_w(x) = \sigma(w_d^T x)$$

$$w_d^T x = w_0 + w_1 x_{(1)} + w_2 x_{(2)} + w_3 x_{(1)}^2 + w_4 x_{(2)}^2 + \ldots + w_d x_{(1)}^d + w_{d+1} x_{(2)}^d$$

d=1 $\Rightarrow$ Linear prediction
d=2 $\Rightarrow$ Quadratic prediction
...
d=10 $\Rightarrow$ Polynomial prediction

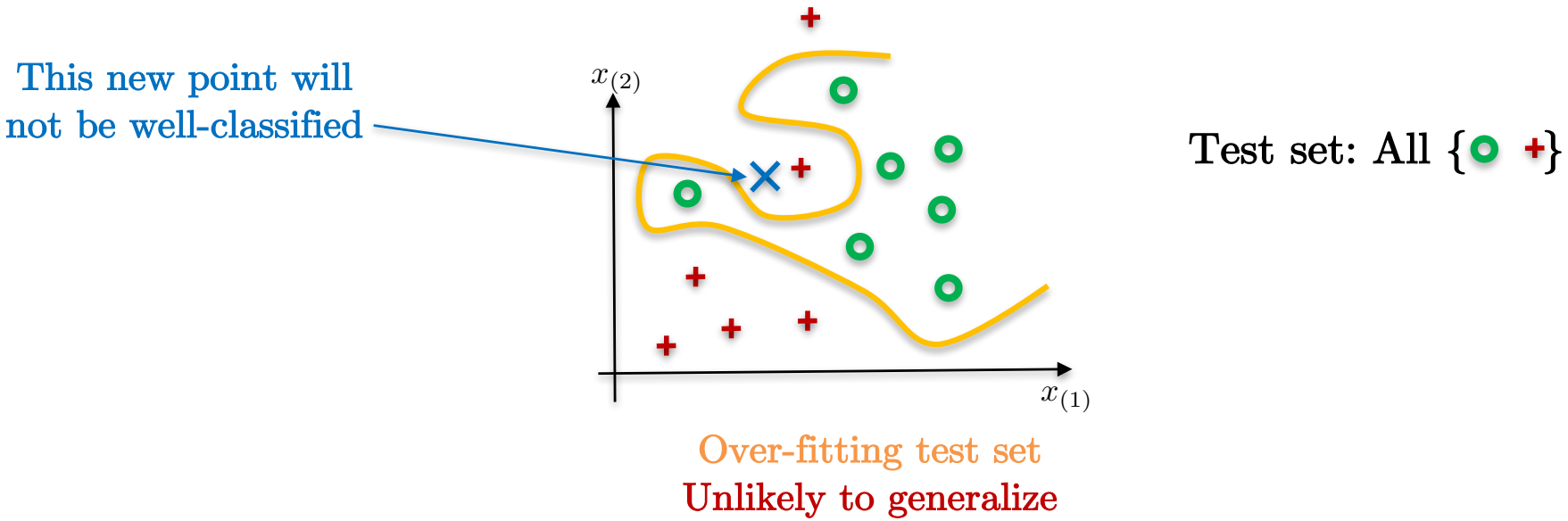
Direct approach

- Suppose we do not have an evaluation set but only the train and test sets to evaluate d. Then, we may want to select d by learning the predictive function with different values d and select d that minimizes the test loss value:



Bad approach

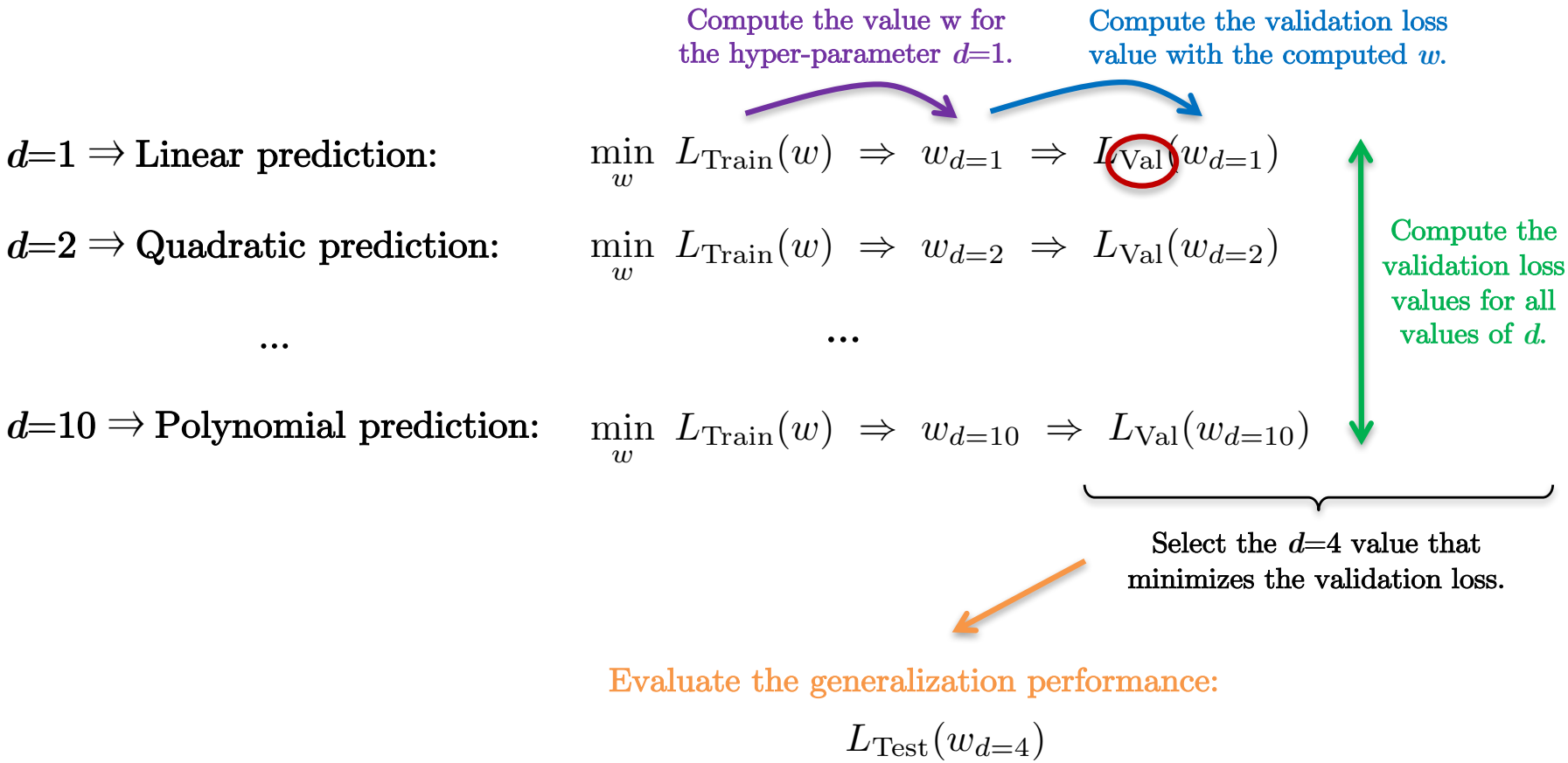
- It is a bad solution to select the value of the hyper-parameter d that minimizes the test loss because we select the d that over-fits the test set $\Rightarrow$ The generalization performance may be poor.



Never use the test set!
(very common error)

Cross-validation approach

- Use the validation set as a "test set" to select the hyper-parameter values (the real test set stays untouched):

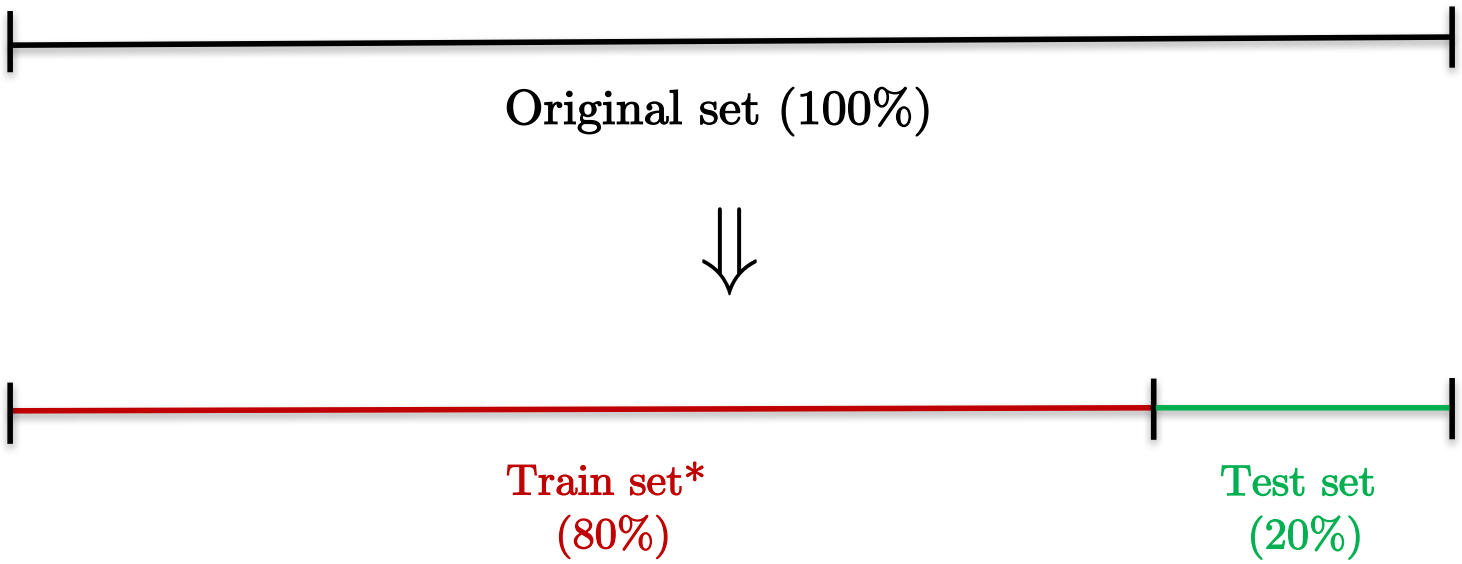


As the test data has not been used – it can be used to evaluate the generalization performance of the learning system.

k-fold cross validation

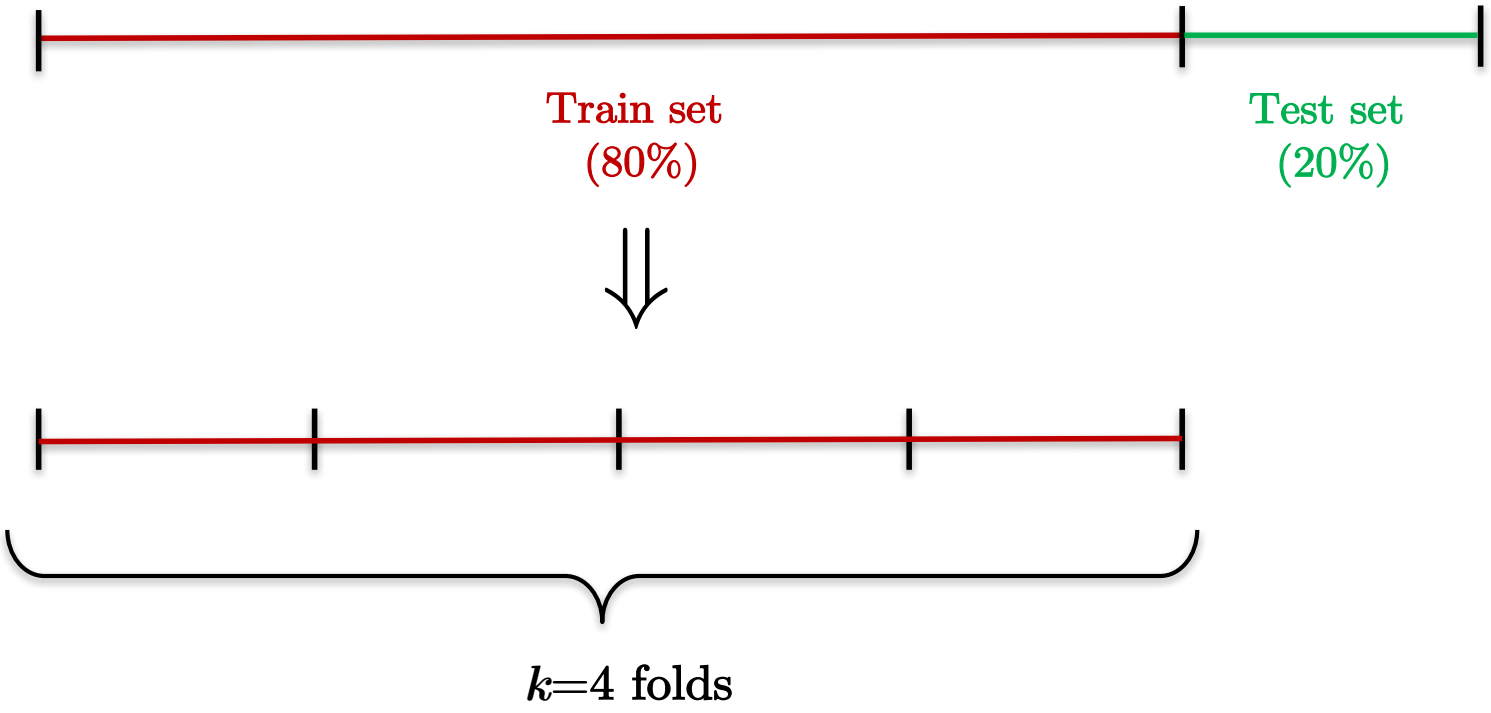
- The k-fold cross validation allows to evaluate the generalization performance on all the training data*.

- Step 1: Split the original set into 2 datasets:
  - A training set
  - A test set



k-fold cross validation

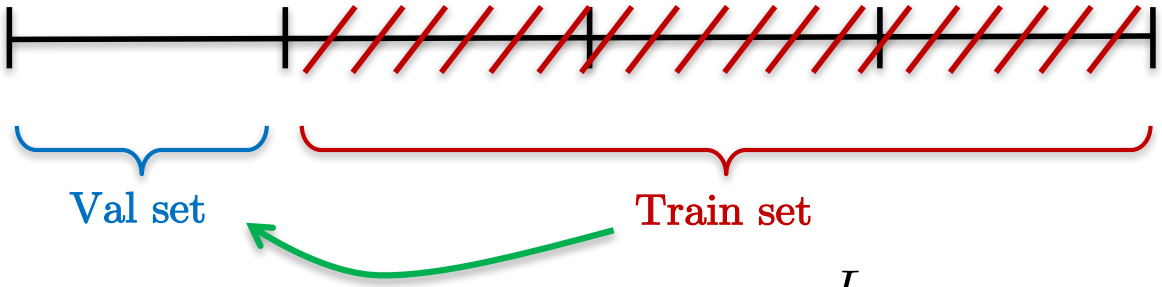
- Step 2: Split the new training set into k=4 folds/sub-sets:



k-fold cross validation

- Step 3: Loop over the k folds and define the evaluation set as one of the k fold and the rest of the data as the train set. Then, learn the parameters of the learning system using the training set and evaluate the hyper-parameter value on the evaluation set.

Fold 1:

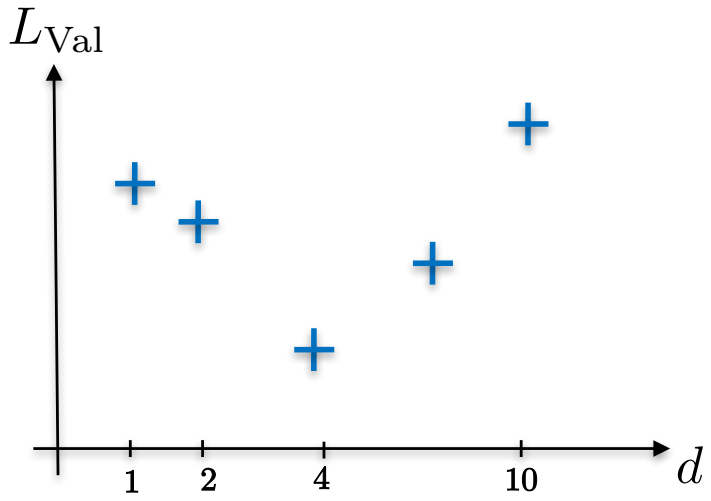


$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=1} \;\Rightarrow\; L_{\text{Val}}(w_{d=1})$$

$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=2} \;\Rightarrow\; L_{\text{Val}}(w_{d=2})$$

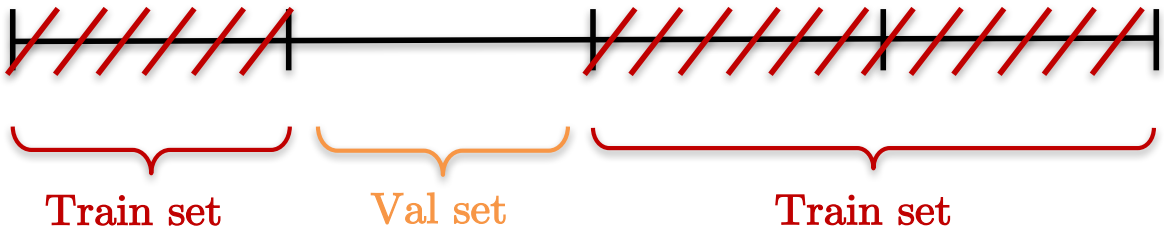
$$\dots$$

$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=10} \;\Rightarrow\; L_{\text{Val}}(w_{d=10})$$



k-fold cross validation

Fold 2:

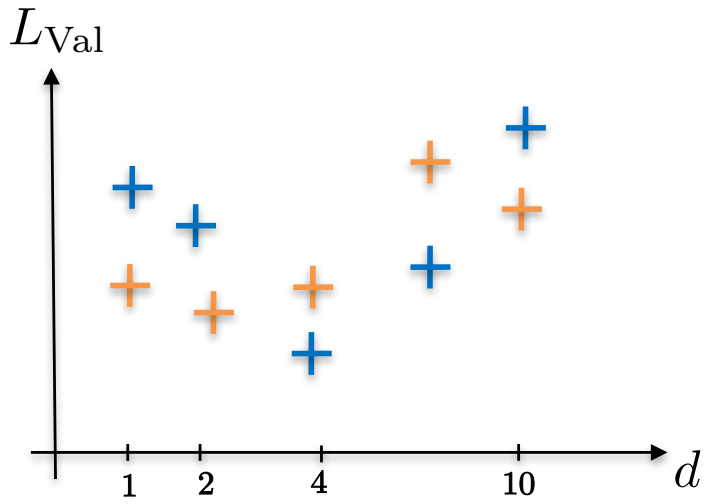


$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=1} \;\Rightarrow\; L_{\text{Val}}(w_{d=1})$$

$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=2} \;\Rightarrow\; L_{\text{Val}}(w_{d=2})$$

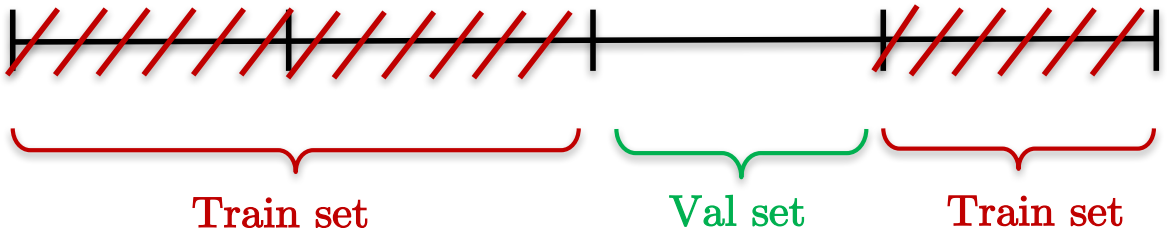
$$\ldots$$

$$\min_w \; L_{\text{Train}}(w) \;\Rightarrow\; w_{d=10} \;\Rightarrow\; L_{\text{Val}}(w_{d=10})$$

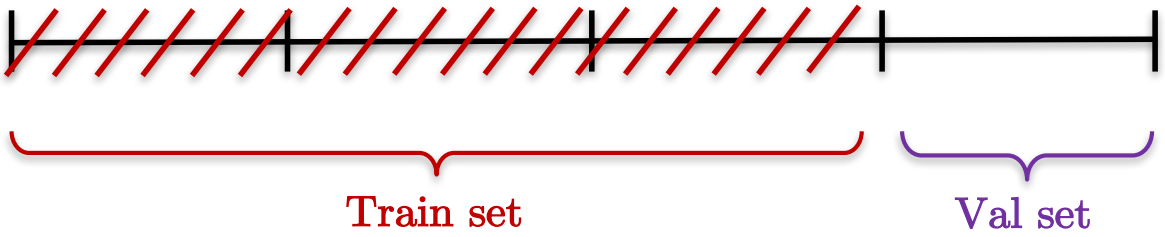


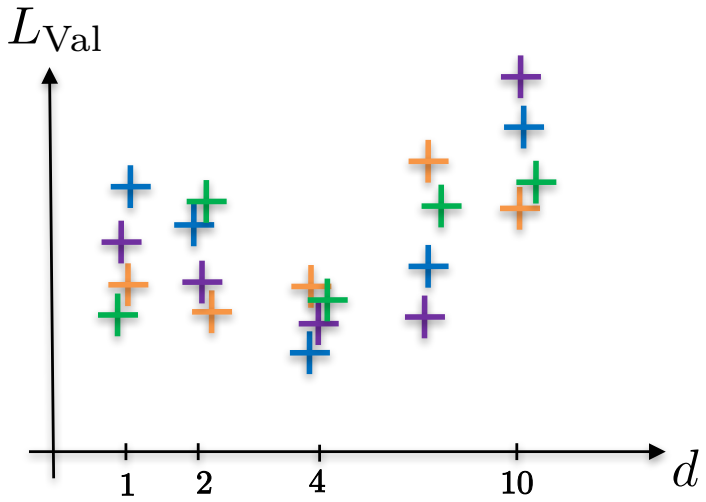
k-fold cross validation

Fold 3:



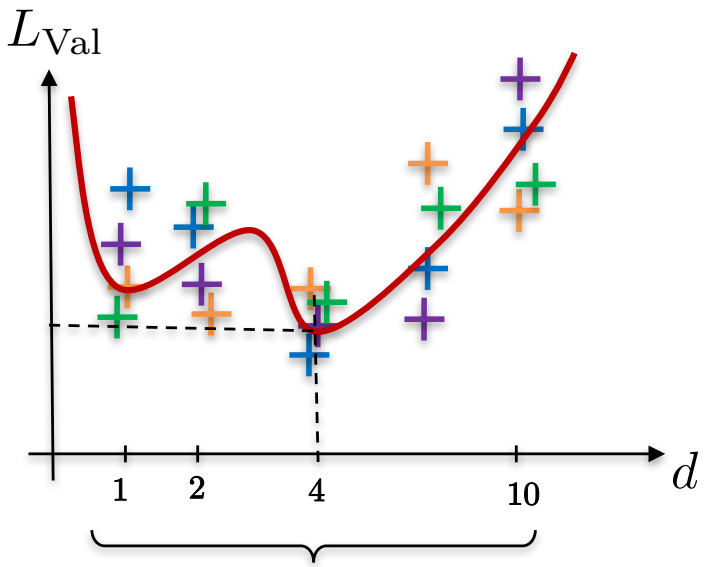
Fold 4:



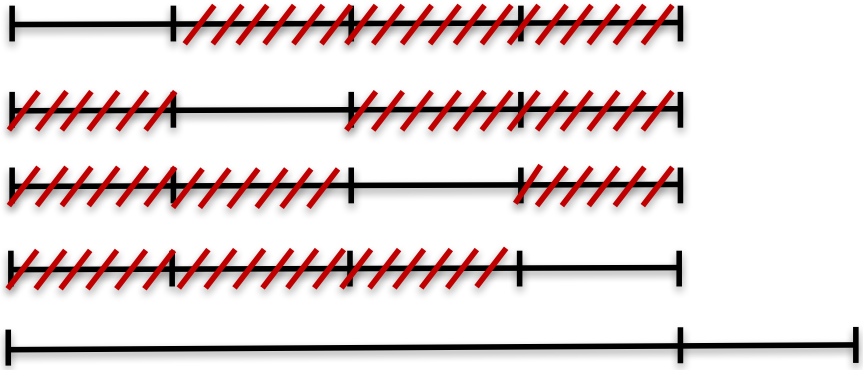


k-fold cross validation

- Step 4: Compute the mean value of the evaluation loss over the all k folds. Select the d value that minimizes the evaluation loss.



Select the d=4 value that minimizes the mean validation loss.



Train set (80%)

Test set (20%)

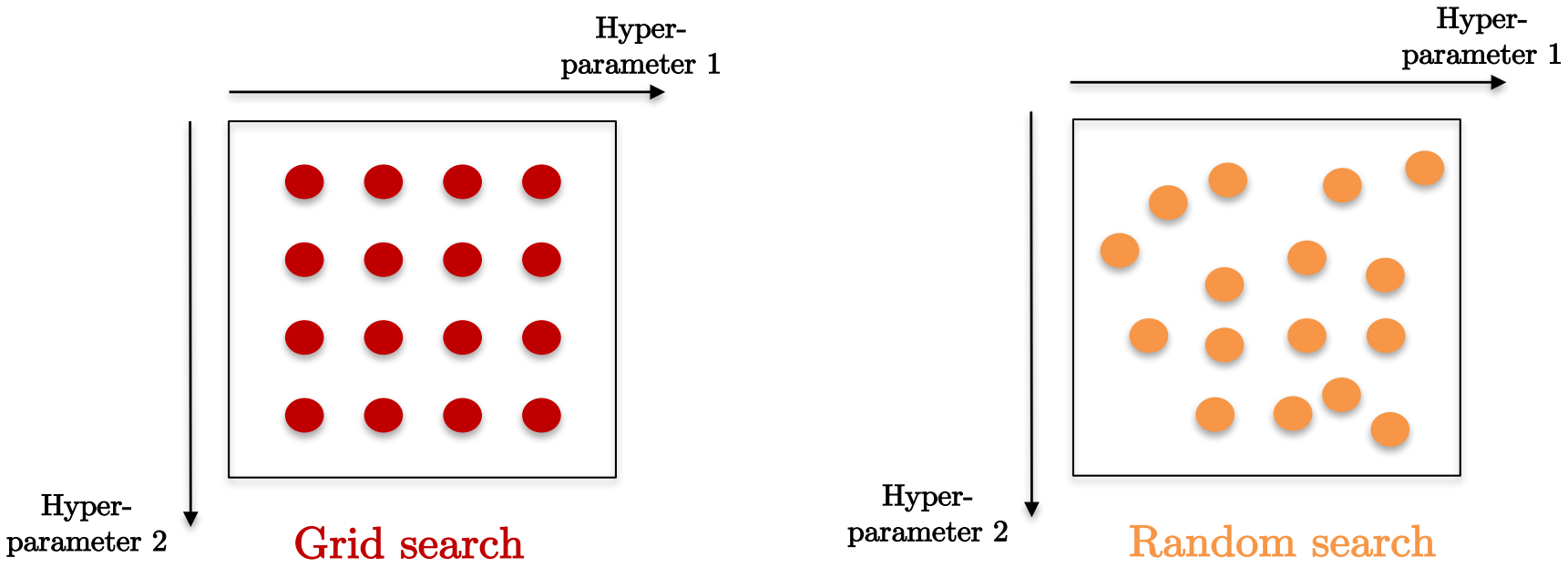
Evaluate the generalization performance:

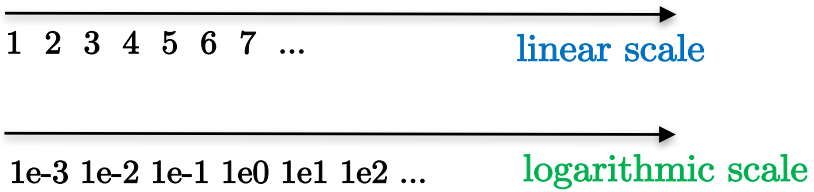
$L_{\text{Test}}(w_{d=4})$

- Generic technique: k-fold cross validation can be applied to estimate any hyper-parameter (s.a. regularization constant, neural network architectures with number of hidden layers, neurons).

Quiz

- How do we select the hyper-parameter values to use?
  - Grid search (linear or logarithmic scales)
  - Random search



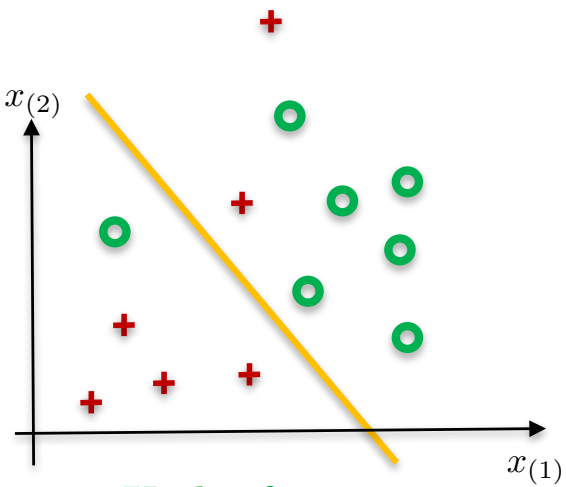


Outline

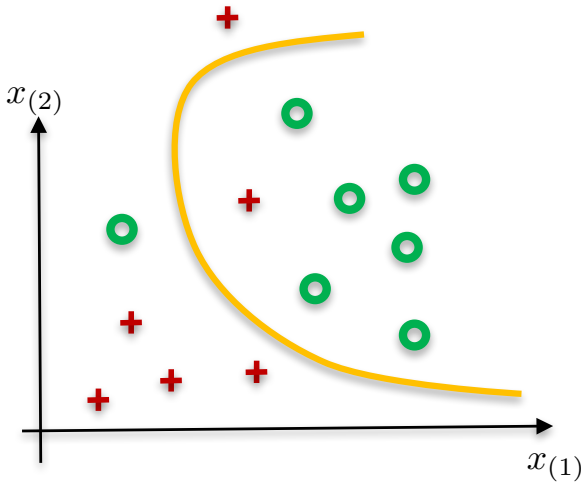
- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- **Learning curves for over- and under-fitting control**
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- Conclusion

Learning curves

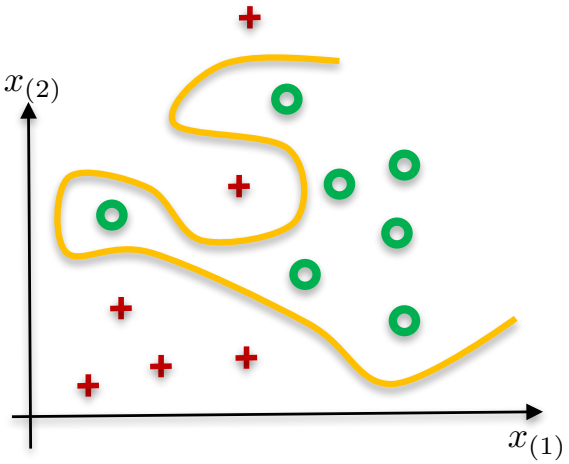
- Learning curves are the shapes of L w.r.t. hyper-parameters. They are useful to identify over-fitting and under-fitting problems, and therefore solve the generalization problem.
- Reminder: Over-fitting is due to high variance of the learning model and under-fitting is caused by high bias.



Under-fitting
High bias
Small variance
Unlikely to generalize



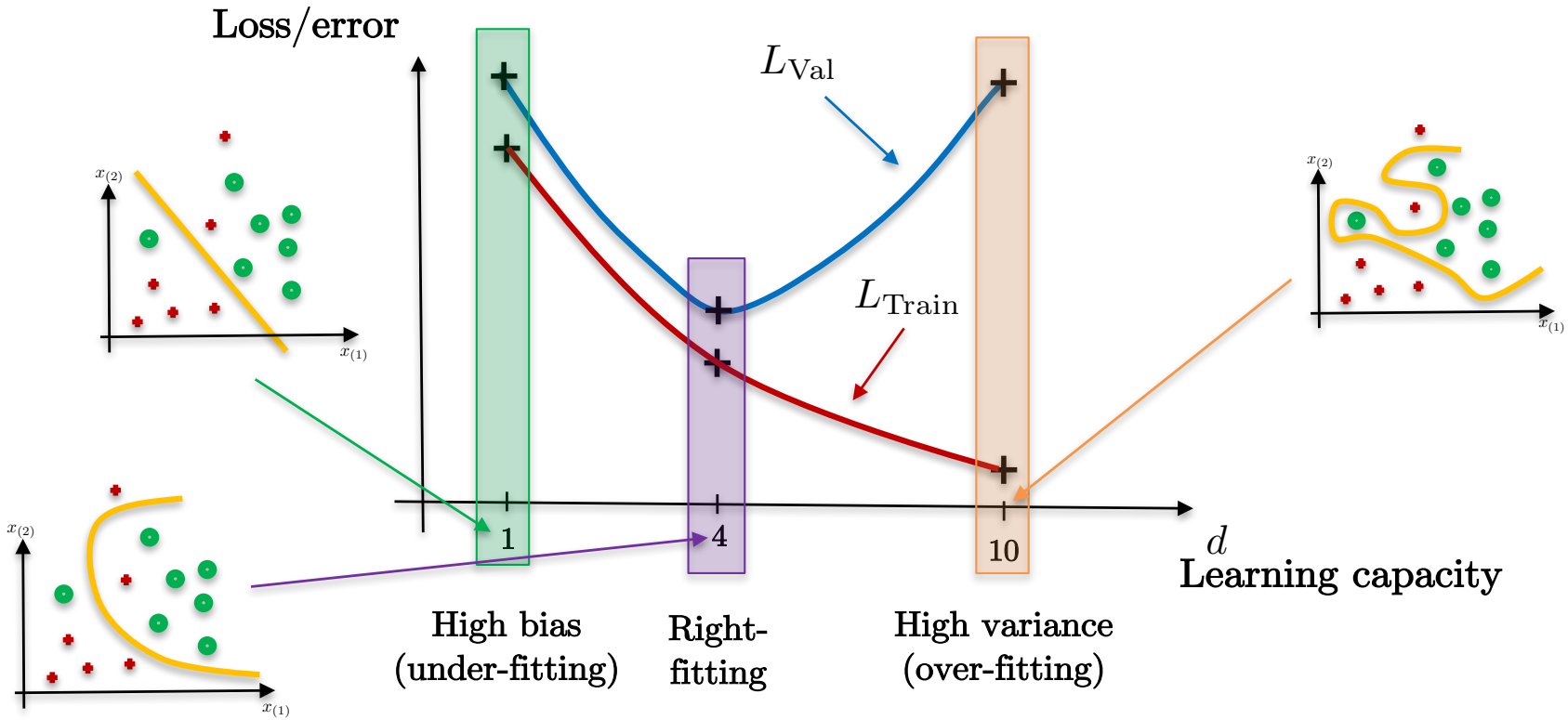
Right-fitting
Good balance between bias and variance
Good generalization



Over-fitting
High variance
Small bias
Unlikely to generalize

Learning curve for capacity

- Plot the losses w.r.t. polynomial degree d, which acts as the learning capacity of the system:



$L_{\text{Train}} \approx L_{\text{Val}} = \text{large}$

$L_{\text{Train}} \approx 0$

$L_{\text{Val}} = \text{large} \gg L_{\text{Train}}$

Learning curve for regularization

- We remind the regularized training loss L with the hyper-parameter λ, which acts as the regularization of the learning system:

$$L(w) = L_{\text{Train}}(w) + \lambda L_{\text{Reg}}(w)$$

$$\frac{1}{n}\sum_{i=1}^{n} \ell(p_w(x_i), y_i) + \frac{\lambda}{d}\sum_{j=1}^{d} w_j^2$$

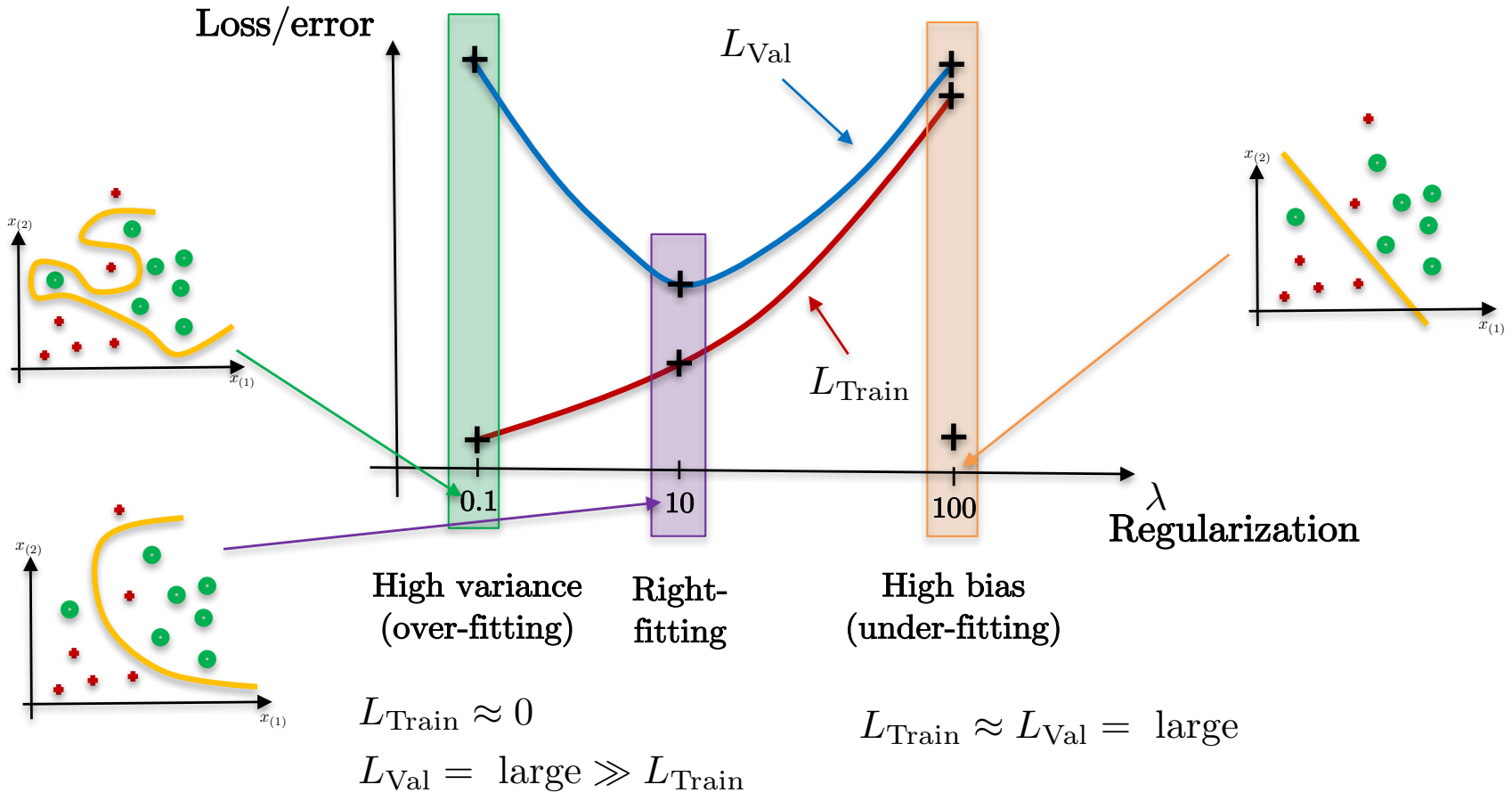
- We first minimize the regularized loss L (it gives us w) for different values of the hyper-parameter λ and then we plot the training and validation losses:

$$L_{\text{Train}}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(p_w(x_i), y_i)$$

$$L_{\text{Val}}(w) = \frac{1}{n_v}\sum_{i=1}^{n_v} \ell(p_w(x_i^v), y_i^v)$$

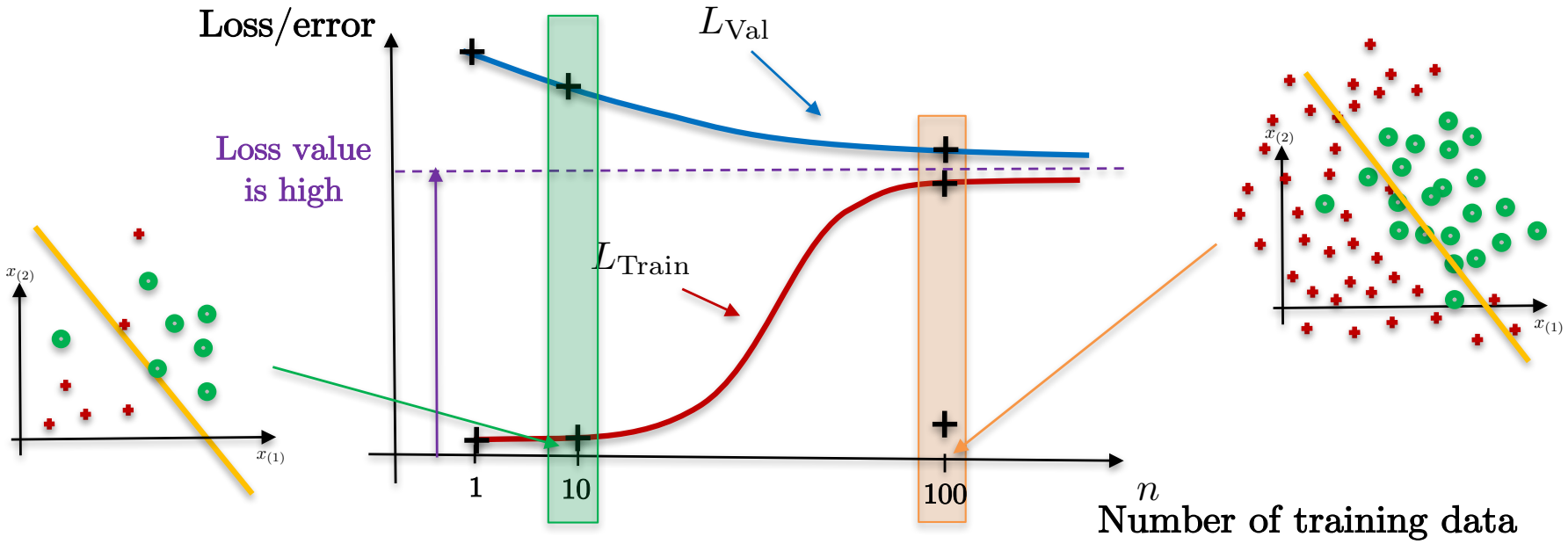
Learning curve for regularization

- Plot of the losses w.r.t. hyper-parameter λ:



Learning curve for training size

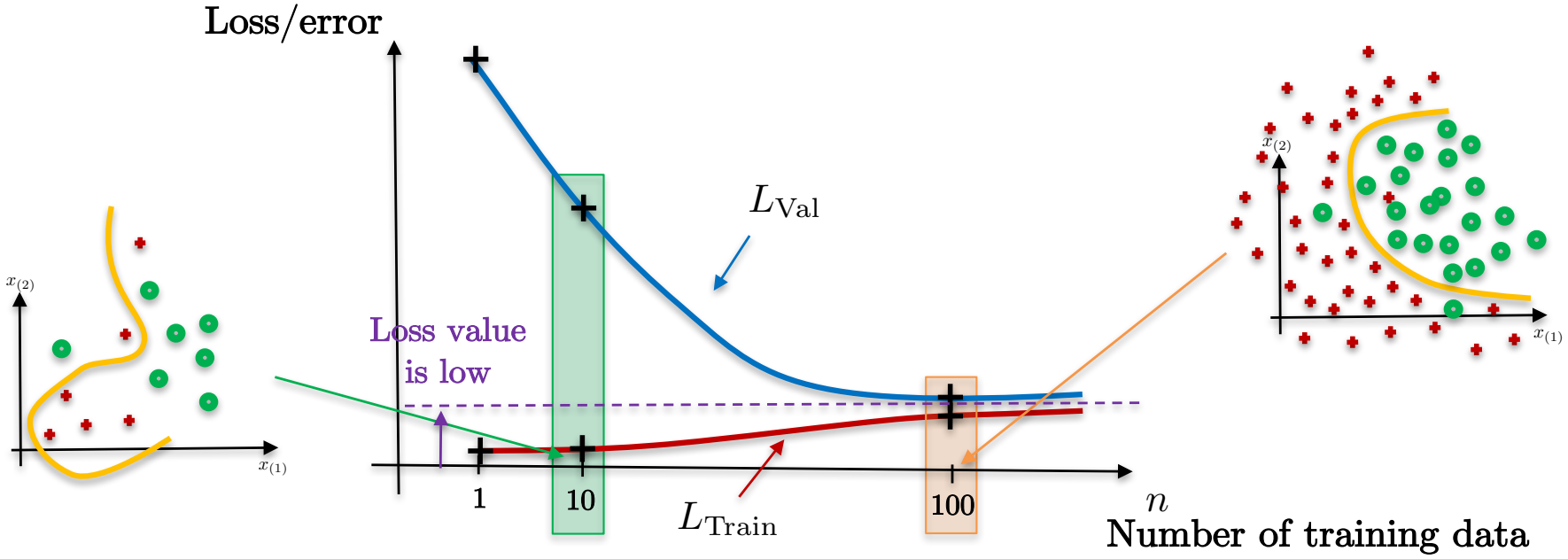
- First, let us suppose that the system has a small learning capacity (like linear learning systems). Then, we plot the losses w.r.t. the number of training data n:



At low capacity (high bias), there is no need to collect lots of training data. Besides, the losses (errors) can be high.

Learning curve for training size

- Second, let us increase the learning capacity of the system (with quadratic or polynomial terms). We plot the losses w.r.t. the number of training data n:



At high capacity (high variance), collecting lots of training data is very helpful as the losses (errors) may decrease significantly.

Good practices

- Learning curves are good debugging tools for learning systems as they allow to derive the best possible generalization property.
- The limitation is the expensive computational time to perform cross-validation on all hyper-parameters.
- This is why it is important to start with a small dataset (extract a sub-part of the original training set) $\Rightarrow$ Compute the learning curves on this small datasets to roughly estimate the hyper-parameters. Then, use the full dataset to fine-tune the hyper-parameters.

Quiz

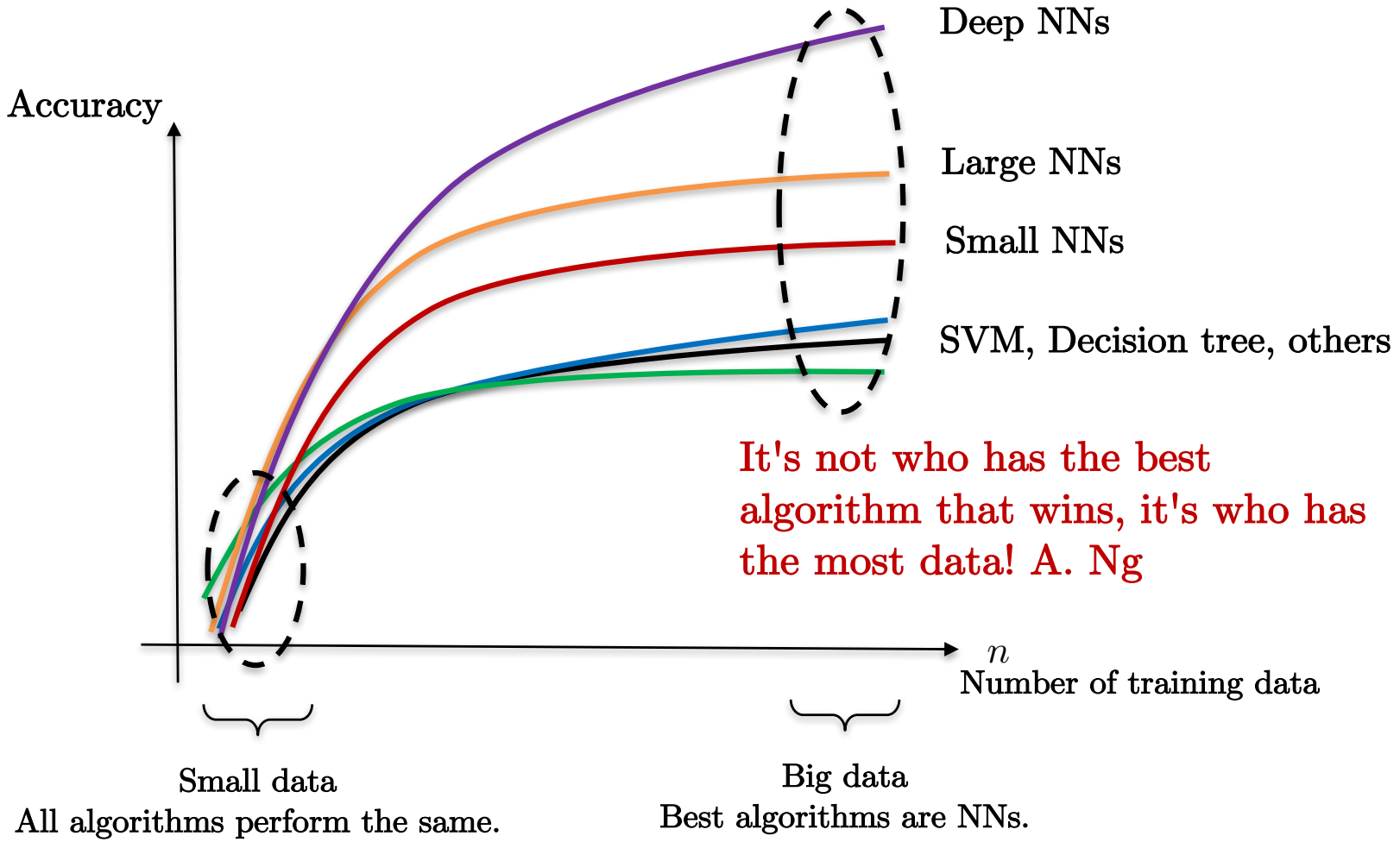
- How to design optimal classifier?
  - Step 3: How to make the spam classifier with the lowest error?
    - Collect (lots of) training data ⇒ Fixes high variance
    - Select a smaller set of hand-crafted features ⇒ Fixes high variance
    - Get additional hand-crafted features ⇒ Fixes high bias
    - Increase learning capacity with polynomial features (x^2,x^3,...), neural networks ⇒ Fixes high bias
    - Increase regularization ⇒ Fixes high variance
    - Decrease regularization ⇒ Fixes high bias

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- **The more data the better**
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- Conclusion

Big data

- How much data to train on?
  - It is not clear how much data to get x% accuracy.
  - It is clear that the more data the better.



Big data

- Assume that
  - x in R^{d} has sufficient d features (information) to predict accurately any output (millions)
  - The learning capacity is high (a NN with billion parameters)
  - The training set is big (millions)

- Then, the generalization performance is maximum. Why?

  (1) High capacity means that L_{train} will be small.

  (2) A learning algorithm (even with high capacity) will not be able to over-fit a large-scale datasets, so we will have $L_{\text{train}} \approx L_{\text{test}}$.

- Putting (1) and (2) together implies L_{test} (test error) to be small!

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- **Hand-crafted features selection**
- Error metric for unbalanced classes
- Good practices
- Conclusion

Designing better hand-crafted features

- Error analysis technique:
  - Use a simple (linear) algorithm and a sub-set of the training data.
  - Learn the parameters of the learning system with gradient-based technique and cross-validation.
  - Manually select the examples in the validation set where the algorithm fails. Study the data and identify potential systematic errors.

- Example: Spam classifier.

  Let us extract 500 spam/non-spam emails. We learn the parameters and hyper-parameters. We observe 100 emails have been misclassified.
  $\Rightarrow$ What types of emails, what features should be removed or added?

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- **Error metric for unbalanced classes**
- Good practices
- Conclusion

Error for unbalanced classes

- Example: Disease classification task.

  Train a classifier $p_w(x)$ to predict illness $(y{=}1)$ or not $(y{=}1)$. We find 1% error on the test set (or 99% correct diagnostic)

  $\Rightarrow$ Seems an excellent classifier but there is actually only 0.5% of patients who are ill (so the classifier finds as much healthy patients as ill ones).

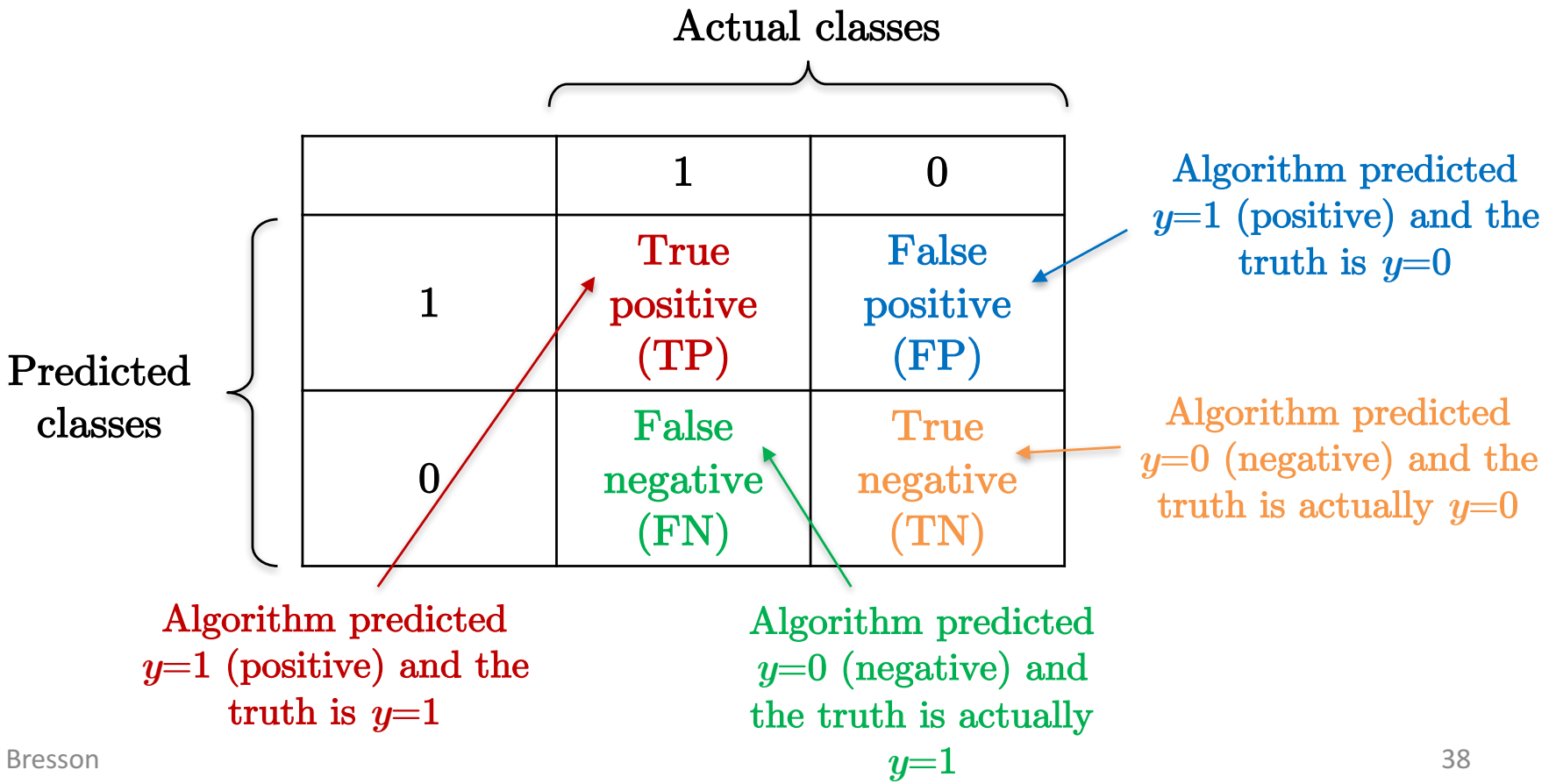
- The problem is due to unbalanced/skewed classes.

  For this example, we have 995 heathy patients and only 5 ill patients.

- A new error metric is required to deal with unbalanced class sizes: Precision and recall are standard metrics for this situation.

Confusion matrix

- Assume that $y=1$ for the rare/small classes.
- Confusion matrix (a.k.a. error matrix): Each row of the matrix represents the instances in a predicted class and each column represents the instances in an actual class.

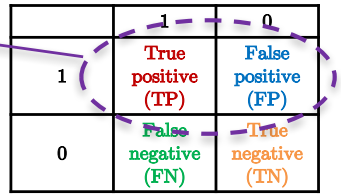


Precision and recall

- **Precision:** Fraction of patients who are actually ill among all patients who are predicted ill.

$$P = \frac{\#\text{True Positive}}{\#\text{Predicted Positive}} = \frac{\#\text{True Positive}}{\#TP + \#FP}$$

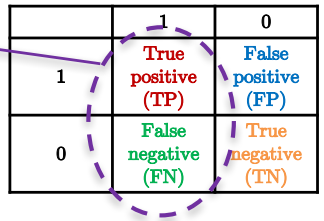
$\Rightarrow$ High P value (close to 1) is good.



- **Recall:** Fraction of patients who are predicted ill among all patients who are actually ill.

$$R = \frac{\#\text{True Positive}}{\#\text{Actual Positive}} = \frac{\#\text{True Positive}}{\#TP + \#FN}$$

$\Rightarrow$ High R value (close to 1) is good.



Precision and recall

- Example: Disease classification task for 1000 patients
  - Actual class: 995 (y=0) and 5 (y=1)
  - Predicted class: 990 (y=0) and 10 (y=1)

- Algorithm predicted 10 patients ill (y=1)
  - Among these 10 patients:
    - 5 patients are actual ill (y=1) $\Rightarrow$ True positive=5
    - 5 patients are actual healthy (y=0) $\Rightarrow$ False positive=5

- Algorithm predicted 990 patients healthy (y=0)
  - Among these 990 patients:
    - 990 patients are actual healthy (y=0) $\Rightarrow$ True negative=990
    - 0 patients are actual ill (y=1) $\Rightarrow$ False negative=0

Precision and recall

- Confusion matrix:

Predicted classes \ Actual classes	1	0
1	TP = 5	FP = 5
0	FN = 0	TN = 990

$$P = \frac{5}{5+5} = 0.5$$

Precision is low

$$R = \frac{5}{5+0} = 1$$

Recall is high

Precision and recall

- Recipe to compute precision and recall:
  - Learn the predictive logistic regression function $p_w(x)$.
  - Define the two predicted classes y=1 and y=0 with $p_w(x)$ and thresholding value 0.5:
    - A patient with probability larger than 50% belong to the class y=1:

      Predict y=1 if $p_w(x) \geq 0.5$

    - A patient with probability larger than 50% belong to the class y=0:

      Predict y=0 if $p_w(x) < 0.5$
  - Compute the confusion matrix (TP, FP, FN, TN) given the predicted and actual classes.
  - Compute precision P and recall R.

	1	0
1	True positive (TP)	False positive (FP)
0	False negative (FN)	True negative (TN)

Trading precision and recall

- Suppose we want to predict patients who are ill only if we are very confident (minimize false positive). We can bias the classifier for this task.

	1	0
1	True positive (TP)	False positive (FP)
0	False negative (FN)	True negative (TN)

As classes are defined according to a threshold (0.5):

Predict y=1 if $p_w(x) \geq 0.5$ and y=0 if $p_w(x) < 0.5$,

we can change the threshold to e.g. 0.7 to bias the classifier to high precision (and lower recall):

Predict y=1 if $p_w(x) \geq 0.7$ and y=0 if $p_w(x) < 0.7$.

But we increase the number of patients predicted with no illness but actually are ill.

- Suppose we want to prevent to avoid missing cases of illness (minimize false negative). Then, we can bias the classifier to high recall (and lower precision):

	1	0
1	True positive (TP)	False positive (FP)
0	False negative (FN)	True negative (TN)

Predict y=1 if $p_w(x) \geq 0.3$ and y=0 if $p_w(x) < 0.3$.

But we increase the number of patients predicted with illness but actually are healthy.

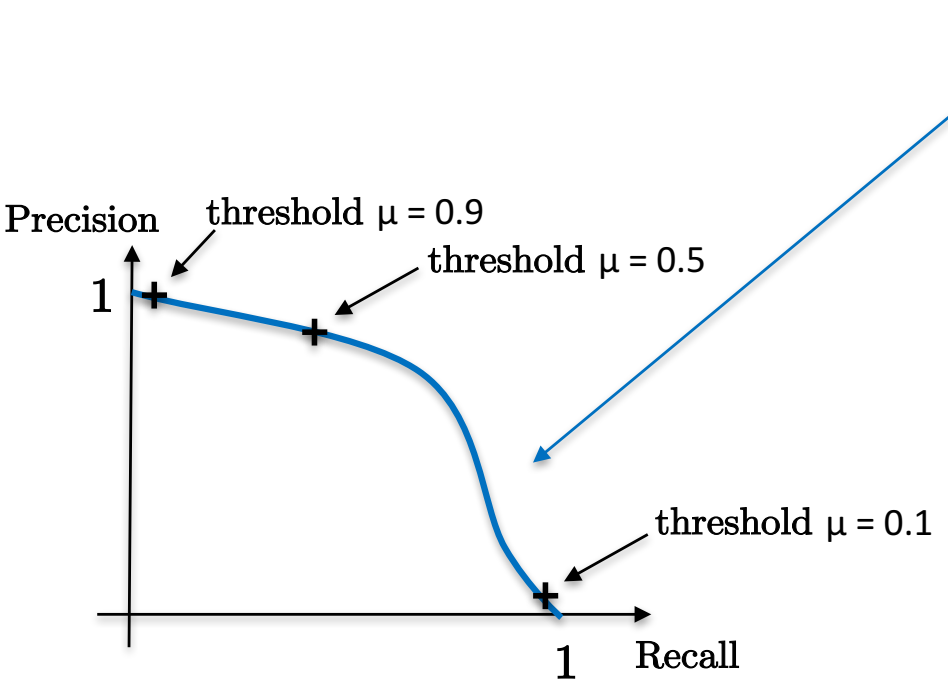
Trading precision and recall

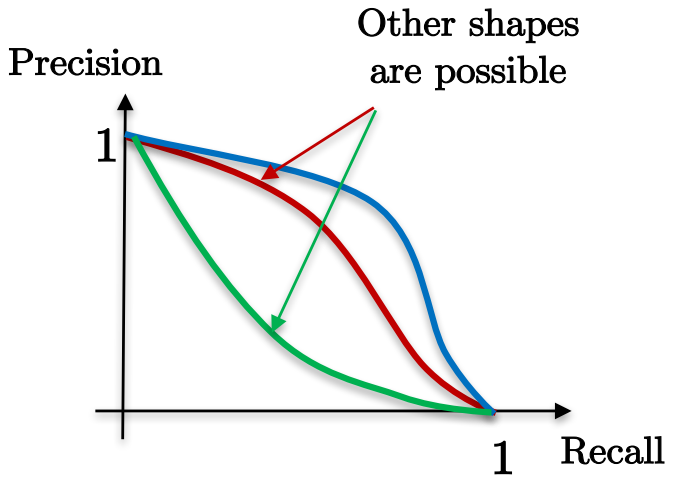
- Classes are defined for each threshold μ :

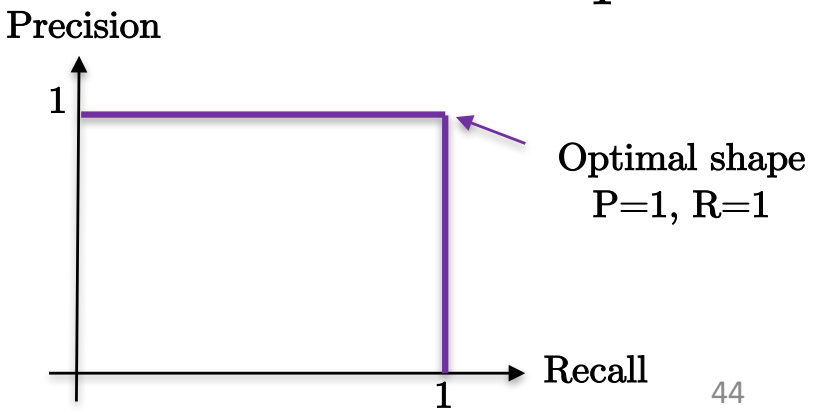
  Predict y=1 if $p_w(x) \geq \mu$ and y=0 if $p_w(x) < \mu$

  And the classes fix the values of precision P and recall R.

  The values of P and R can be plotted as follows:







Threshold selection

- There is an approach to select the threshold value automatically.
- Ideally, we want to have P=1 and R=1:
  - A naive and bad metric is the mean: $M = \frac{P+R}{2}$
  - The F_1 score is a good metric: $F_1 = 2\frac{PR}{P+R}$
- Example:

	P	R	M	F_1
μ = 0.9	0.5	0.4	0.45	0.44
μ = 0.5	0.7	0.1	0.4	0.17
μ = 0.1	0.01	1	0.51	0.03

Best F_1 ⇒ best compromise precision and recall

Predict y=1 all the time

Best recall, but worst precision

Quiz

- What is the difference between the loss function and the error function (or equivalently the accuracy function)?

Loss function is a surrogate of the error function.

Continuous and differentiable function $\Rightarrow$ Gradient-descent techniques can be used.

Discrete (not-differentiable) function $\Rightarrow$ No gradient-descent techniques can be used. Alternative are grid search (brute force) and genetic algorithms (may be slow and not applicable to NNs).

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- **Good practices**
- Conclusion

Good practices

- A recipe for designing learning systems:
  - Step 1: Pre-process data (zero-mean, unit variance)
  - Step 2: Choose a learning algorithm (linear, NNs)
  - Step 3: Make sure you have enough learning capacity. Extract a sub-set of training data and over-fit them, i.e. $L_{\text{Train}} \approx 0$ (L_{Val} is high) by manually selecting the hyper-parameters.
  - Step 4: Add regularization and evaluate the generalization performance on the validation set. We should have $L_{\text{Val}} \searrow$ and $L_{\text{Train}} \nearrow$. The gap between L_{Val} and L_{Train} should be as small as possible.
  - Step 5: Use all training data and cross-validation to estimate the parameters and the hyper-parameters (long running time). Ideally, $L_{\text{Val}} \approx L_{\text{Train}} \approx$ small value.

Outline

- Strategy for developing a data analysis system
- Cross-validation for hyper-parameter estimation
- Learning curves for over- and under-fitting control
- The more data the better
- Hand-crafted features selection
- Error metric for unbalanced classes
- Good practices
- **Conclusion**

Conclusion

- Designing high-quality learning systems depends on:
  - Complexity and availability of data
  - Mathematical (and intuitive) understanding of learning algorithms
  - Coding skills (Python, Java, etc)
  - Computational infrastructure (CPU, GPU, cloud, data storage, database, etc)
  - Best practices (different recipes exist)
  - Personal practice..

Coding exercise

- tutorial06.ipynb

```
# plot
x = list_n
plt.figure(1)
plt.plot(x, train_error_tab,label='Train error'.format(i=1))
plt.plot(x, val_error_tab,label='Val error'.format(i=2))
plt.legend(loc='best')
plt.title('Error w.r.t. n')
plt.xlabel('n')
plt.ylabel('Error')
plt.ylim([0,35])
plt.show()
```

Questions?