Image processing for Earth Observation

3d classification

Best practices

Devis TUIA

EPFL, fall semester 2024

École polytechnique fédérale de Lausanne

Content (6 weeks)

- W1 General concepts of image classification

  Traditional supervised classification methods

  Best practices
- W2 Paper reading club!

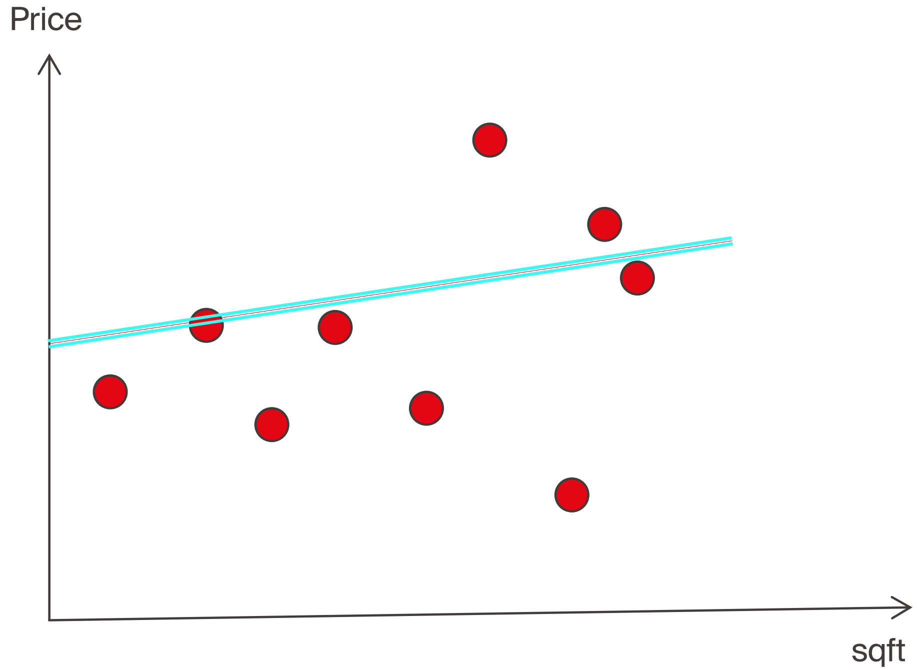
- W3 Elements of neural networks
- W4 Convolutional neural networks
- W5 Convolutional neural networks for semantic segmentation
- W6 Paper reading club!

Some good practices

- Fighting overfitting
- Crossvalidation
- Spatial splits
- Accuracy measures

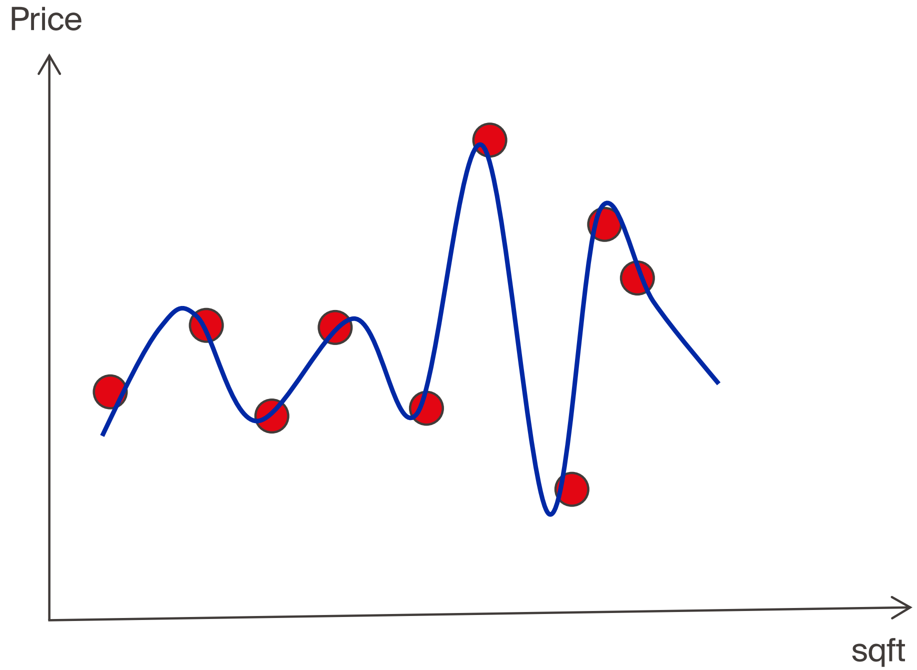
The enemy: overfitting

- Sometimes a simple linear prediction is not optimal



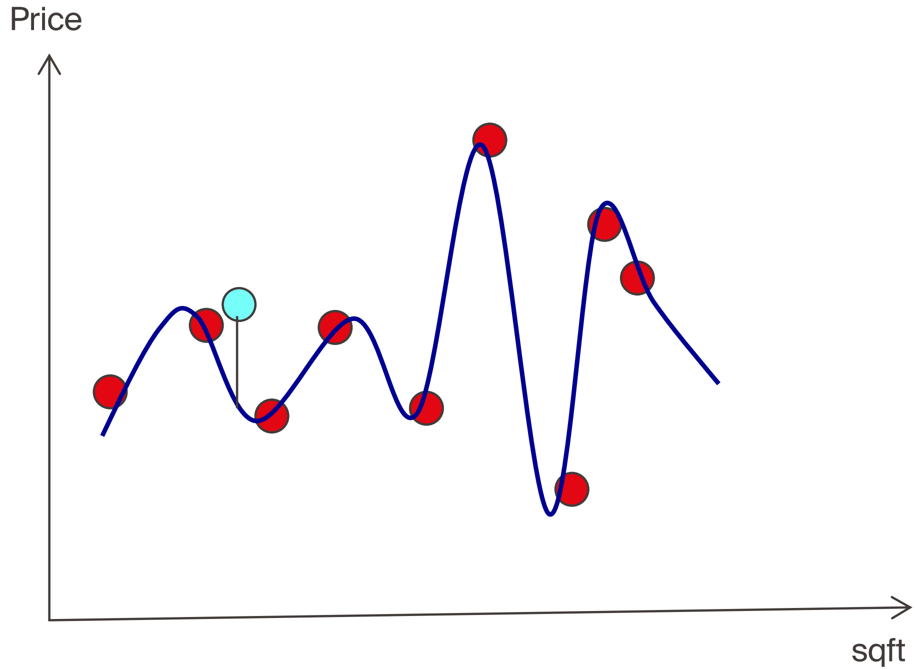
Going nonlinear

- Sometimes a simple linear prediction is not optimal
- We would prefer something nonlinear
- We would avoid to be too close to the training data



Going nonlinear

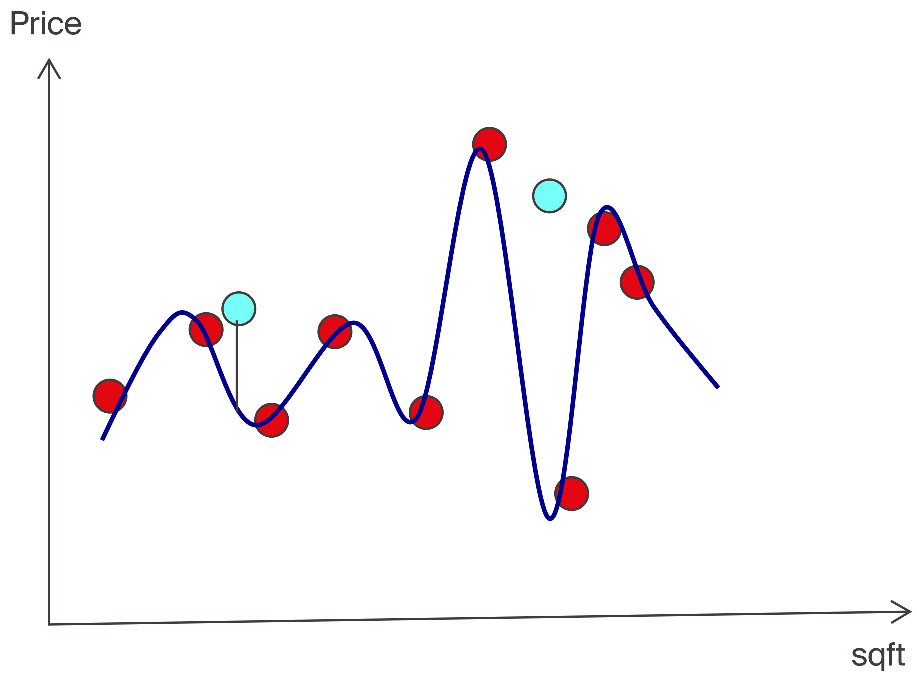
- Sometimes a simple linear prediction is not optimal
- We would prefer something nonlinear
- We would avoid to be too close to the training data
- Otherwise, when a new unseen point comes...
- This is called **overfitting**



D. Tuia

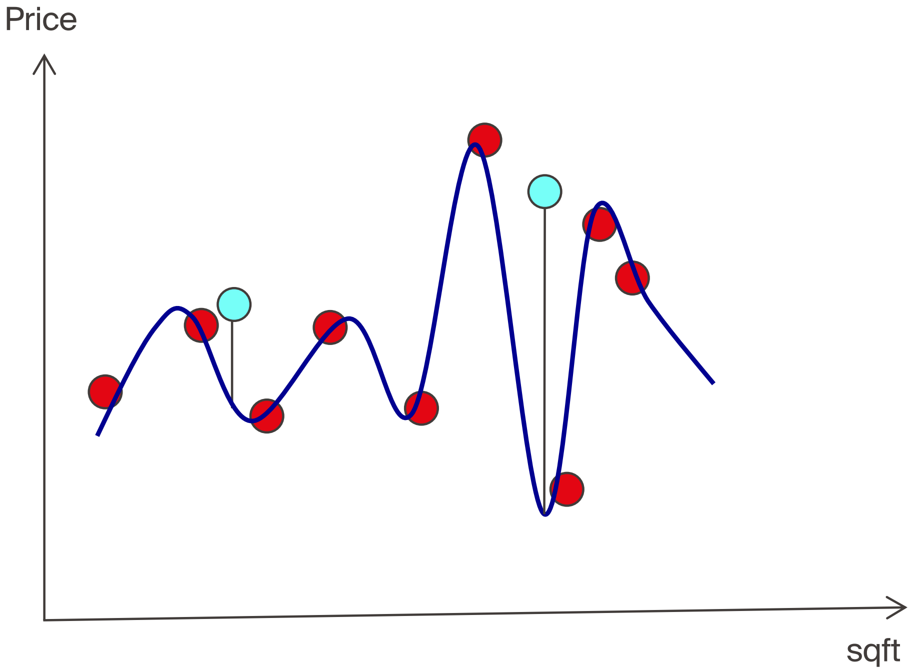
Going nonlinear

- Sometimes a simple linear prediction is not optimal
- We would prefer something nonlinear
- We would avoid to be too close to the training data
- Otherwise, when a new unseen point comes…
- This is called **overfitting**



Going nonlinear

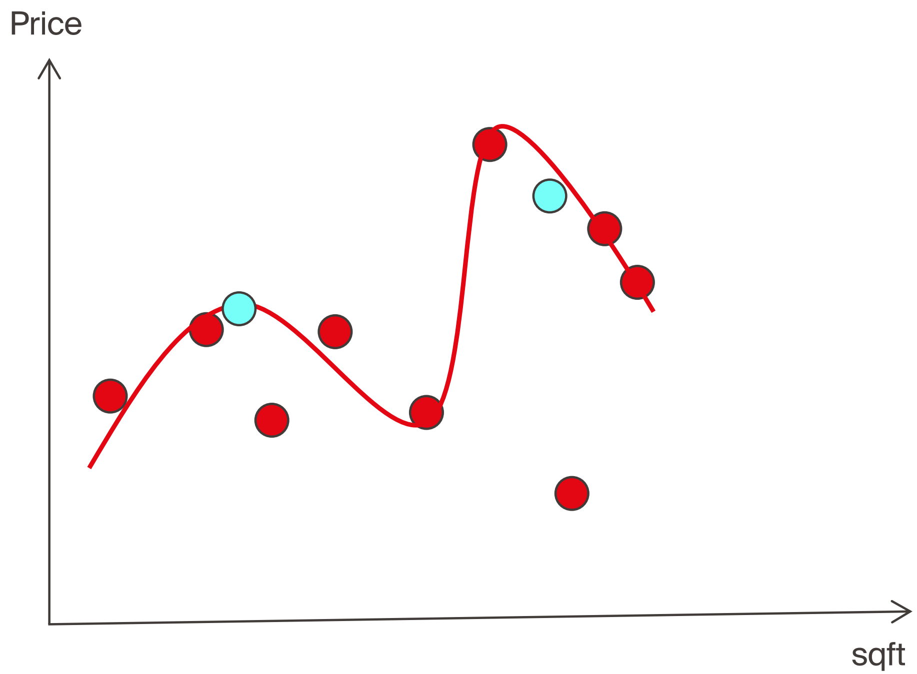
- Sometimes a simple linear prediction is not optimal
- We would prefer something nonlinear
- We would avoid to be too close to the training data
- Otherwise, when a new unseen point comes…
- This is called <u>**overfitting**</u>



Going nonlinear

- Sometimes a simple linear prediction is not optimal

- We would prefer something nonlinear



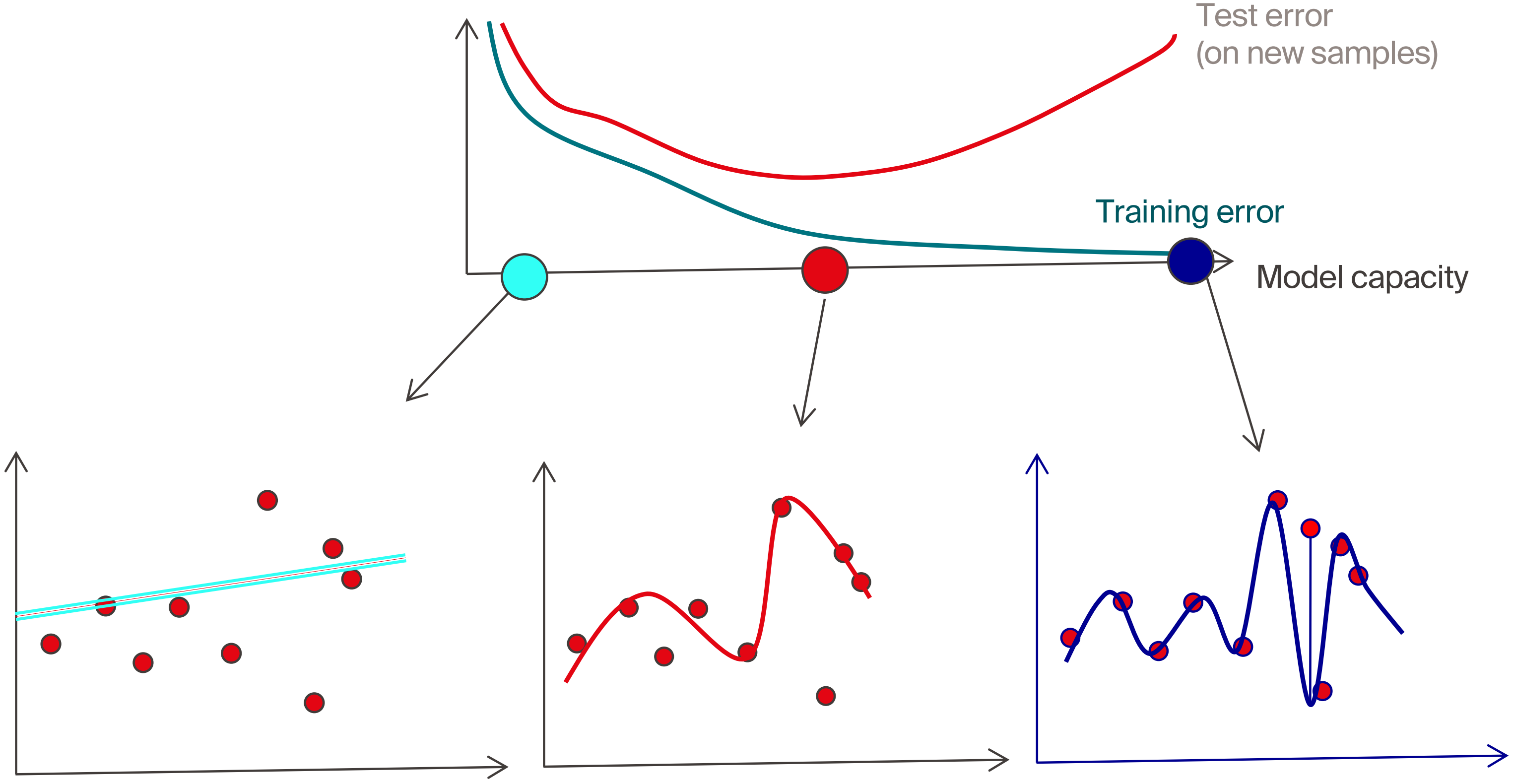
Overfitting (and generalization)

- Overfitting happens when you trust your training data too much.
- Our aim is to model the data structure well, not to represent the training data well
- If we fail miserably on unseen data, the learning was a failure
- So we need to
  - Train well

    = minimize errors where we can estimate them
  - Limit model overcomplexification

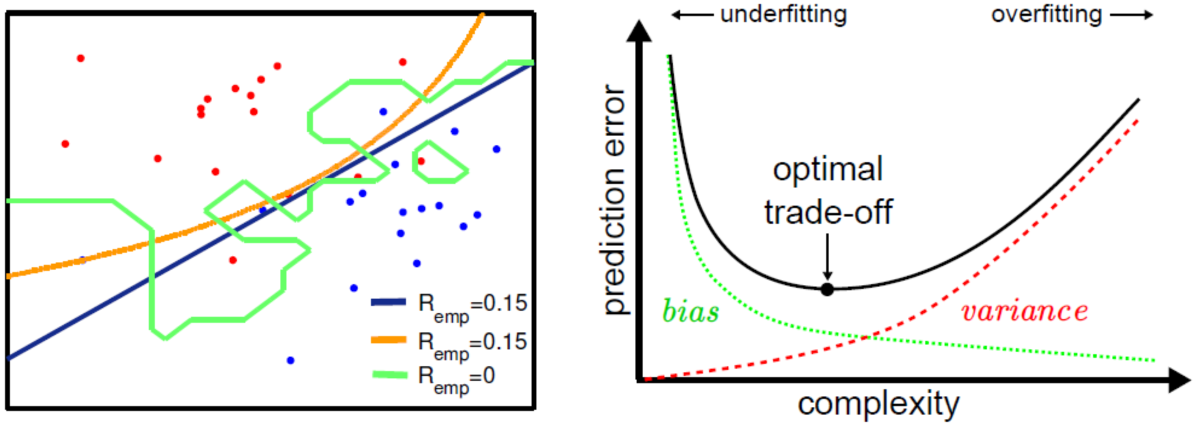
    = avoid overfitting

Overfitting (and generalization)



More complexity is not always good

- A classification algorithm has free parameters to be tuned
- Typically related to the generalization capability
- A good generalization is a trade-off between :
  - Having a low training error (fitting well enough the training examples)
  - But keeping the model “simple”, with a low complexity



Source: F. de Morsier, PhD Thesis

How do we limit complexity?

- By regularizing solutions

By penalising every time we split into a deeper tree, we avoid taking decisions on smaller and smaller "leaves"

By keeping parameters small, we avoid the model to rely too much on the training data

How do we construct the partitioning?

- Solution
  - A. Stop early (e.g. set a minimum depth)
  - B. Prune the tree using cost:

$$\sum_{m=1}^{|T|} Gini(m) + \alpha|T|$$

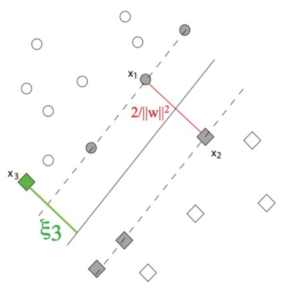
- First grow a very large (deep) tree T_0. You now have the solution for $\alpha = 0$.
- For each α, there is a subtree $T \subset T_0$
- Increase α and re-run. The regularizer is the price to pay for increasing the number of terminal nodes
- Use a k-fold cross-validation approach to evaluate the error function above. Use the average error as final measure.
- Once minimized, grow an optimal tree using all data.

Tolerance to errors?

- One single missclassification can make the classifier overfit!

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 + C\sum_i \xi_i$$

- Every time we make a mistake, we penalize by ξ_i (= the distance of the sample from the classification plane)
- If no mistake, $\xi_i = 0$
- Meaning : the bigger the mistake, the more we penalize



How do we limit complexity?

- By regularizing solutions

How do we construct the partitioning?

- Solution
  - A. Stop early (e.g. set a minimum depth)
  - B. Prune the tree using cost:

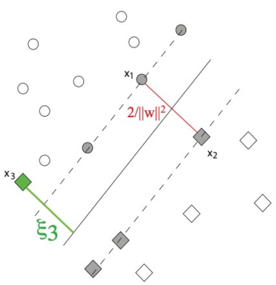
$$\sum_{m=1}^{|T|} Gini(m) + \alpha|T|$$

- First grow a very large (deep) tree T_0. You now have the solution for $\alpha = 0$.
- For each α, there is a subtree $T \subset T_0$
- Increase α and re-run. The regularizer is the price to pay for increasing the number of terminal nodes
- Use a k-fold cross-validation approach to evaluate the error function above. Use the average error as final measure.
- Once minimized, grow an optimal tree using all data.

Tolerance to errors?

- One single missclassification can make the classifier overfit!

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 + C\sum_i \xi_i$$



- Every time we make a mistake, we penalize by ξ_i (= the distance of the sample from the classification plane)
- If no mistake, $\xi_i = 0$
- Meaning : the bigger the mistake, the more we penalize

Optimization objective: **min (errors on training data + regularizer)**

How do we limit complexity?

- By regularizing solutions
- By choosing parameters that do not prone overfitting (e.g. min number of samples in each leaf, which has a similar effect of the regularizer seen in previous slide)

How do we limit complexity?

- By regularizing solutions
- By choosing parameters that do not prone overfitting (e.g. min number of samples in each leaf, which has a similar effect of the regularizer seen in previous slide)
- By early stopping (important in neural netwroks)

In other words: know when to stop minimizing the training error.

(cross) validation is a possible way to estimate that.

Some good practices

- Overfitting
- (Cross)validation
- Spatial splits
- Accuracy measures

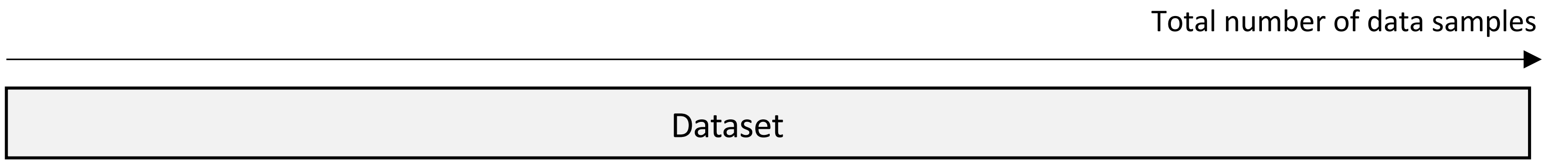
Use a validation set when you have it!

- A classification algorithm has free parameters to be tuned
- Cross-validation can allow to estimate the generalization capability (=accuracy on unseen test data)

- If you have a lot of data, you can take out part of it for validating how well your model is doing on **data never seen during training**

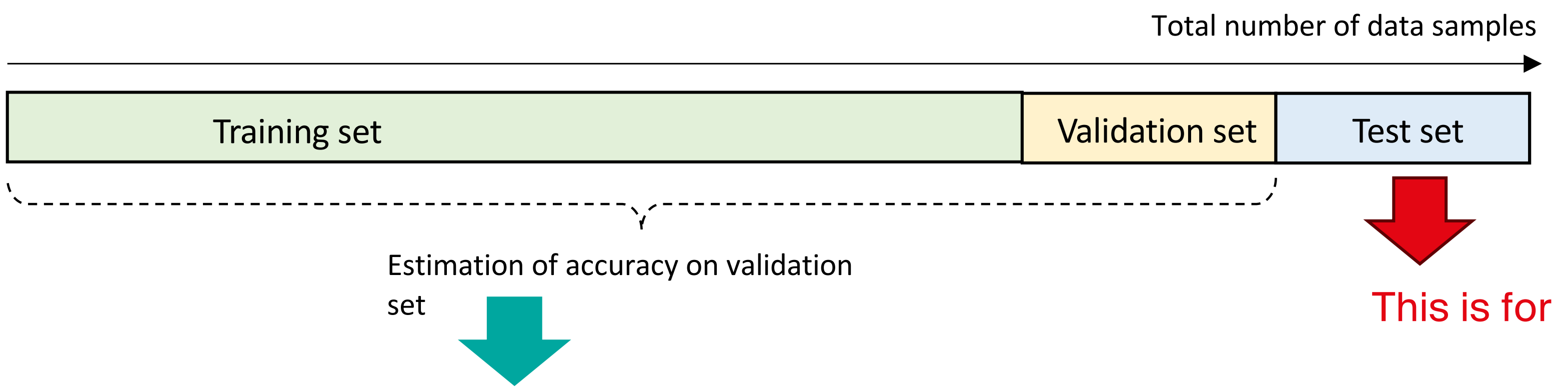
Validation with some left out data

- Validation of model parameters



Validation with some left out data

- Validation of model parameters



This is to find the best model parameters.

Every set of parameters is optimized on the training set. Lowest error on validation set wins.

This is for final testing only.

It is data the model never sees until the very end.

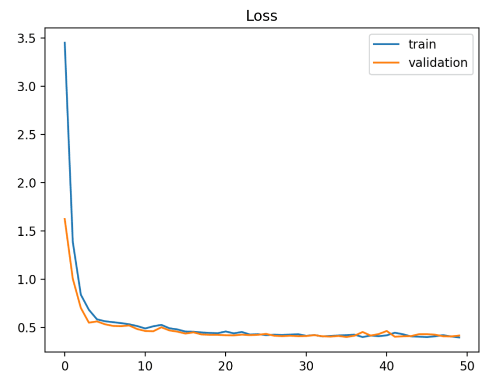
The validation data are used to set model parameters, so the model is optimized for them

Speaker

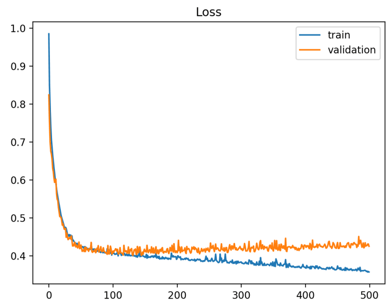
Use a validation set when you have it!

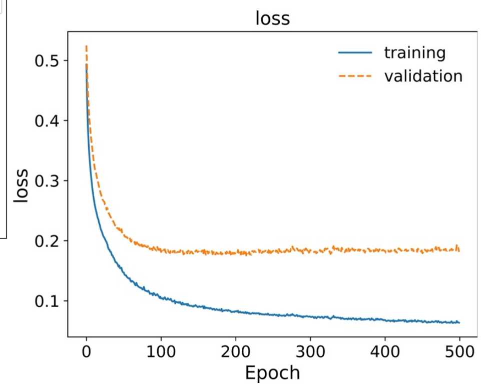
- A classification algorithm has free parameters to be tuned
- Cross-validation can allow to estimate the generalization capability (=accuracy on unseen test data)

- If you have a lot of data, you can take out part of it for validating how well your model is doing on **data never seen during training**
- This can be used for:
  - Comparing different parameters sets
  - Seeing if you are overfitting



VS





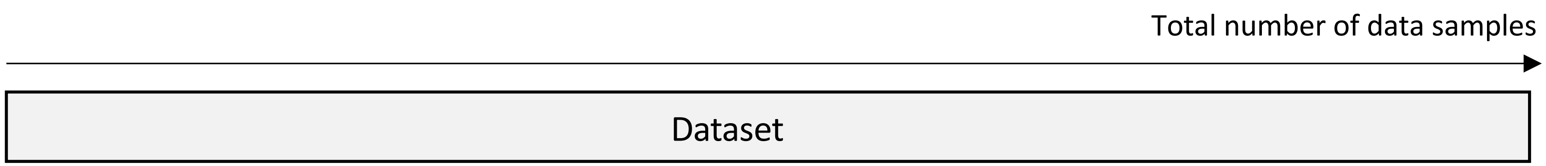
K-fold Cross-validation

- When you don’t have many data, you need to work with what you have…

- Idea behind cross-validation:
  - Split the training examples into different subset or “folds”
  - Leave one fold for testing and estimating accuracy
  - Use the remaining folds for training the model
  - Repeat above steps until all folds have been once tested
  - Take average error as performance metric

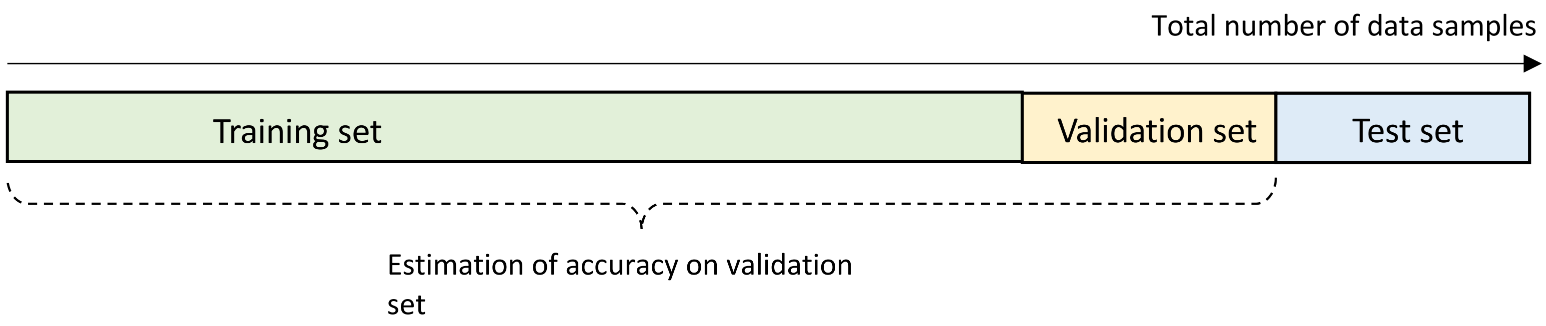
Cross-validation

- K-folds cross-validation



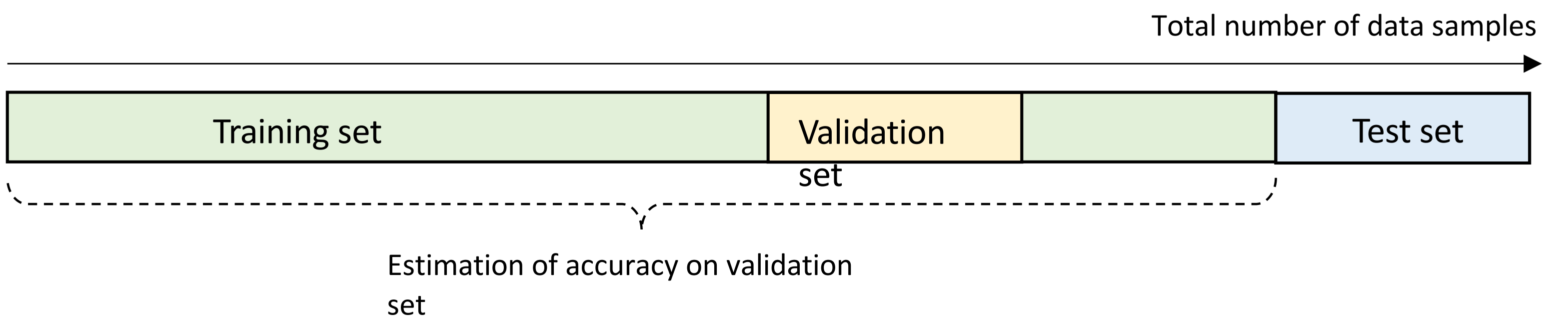
Cross-validation

- K-folds cross-validation



Cross-validation

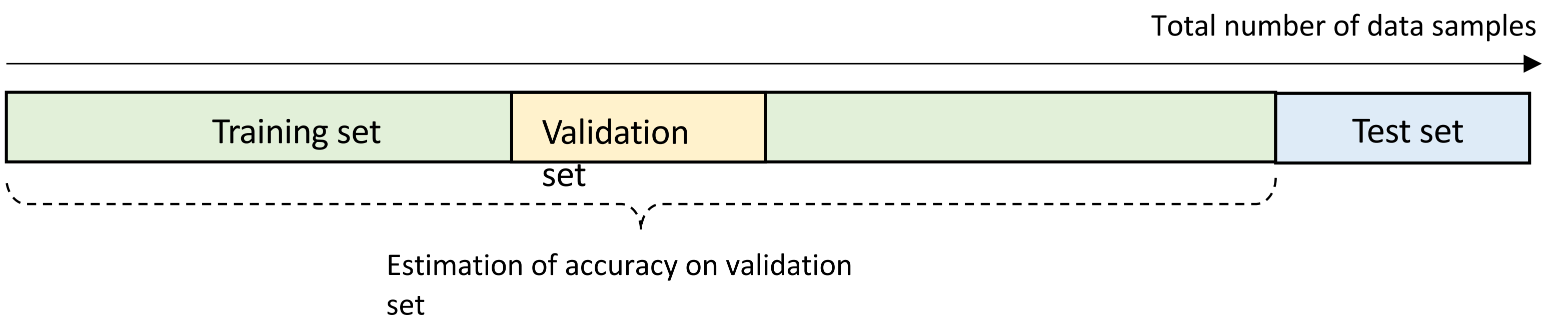
- K-folds cross-validation



D. Tuia

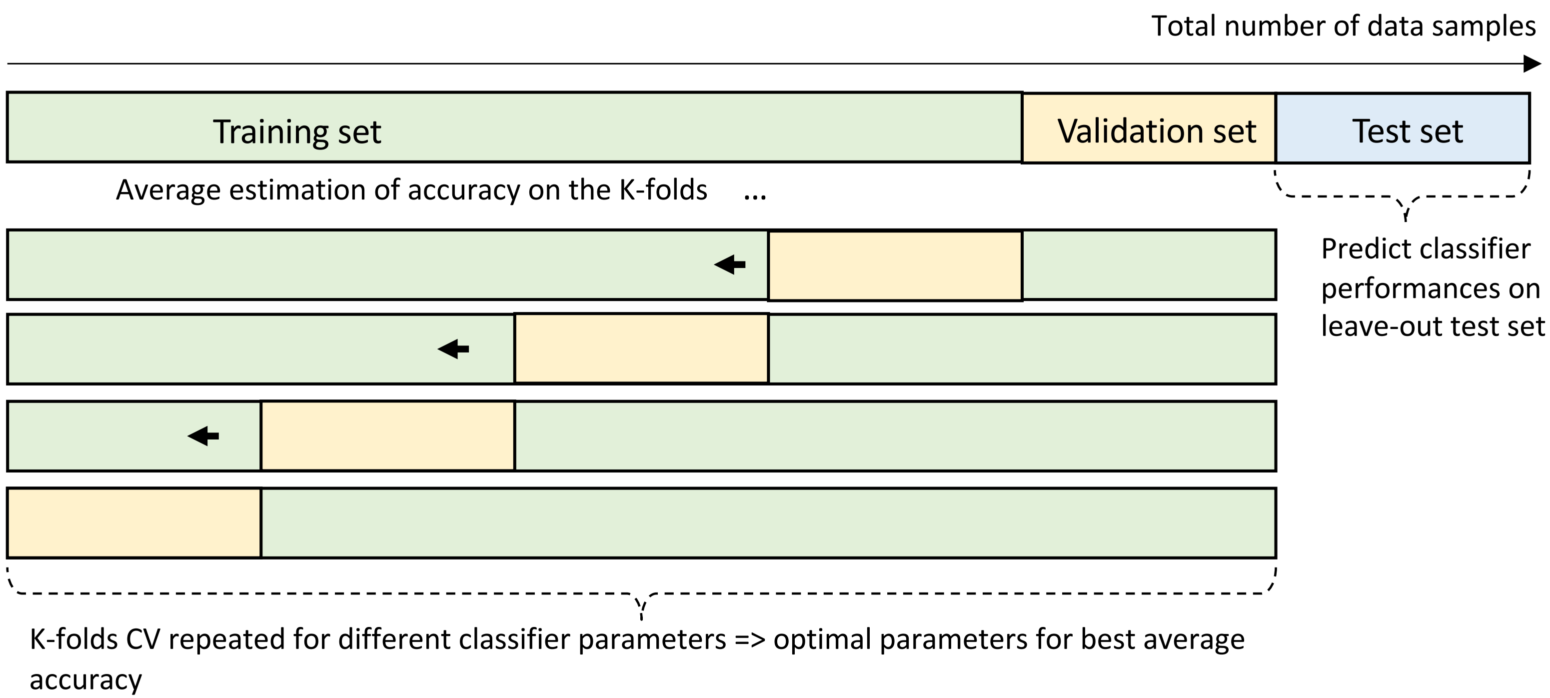
Cross-validation

- K-folds cross-validation



Cross-validation

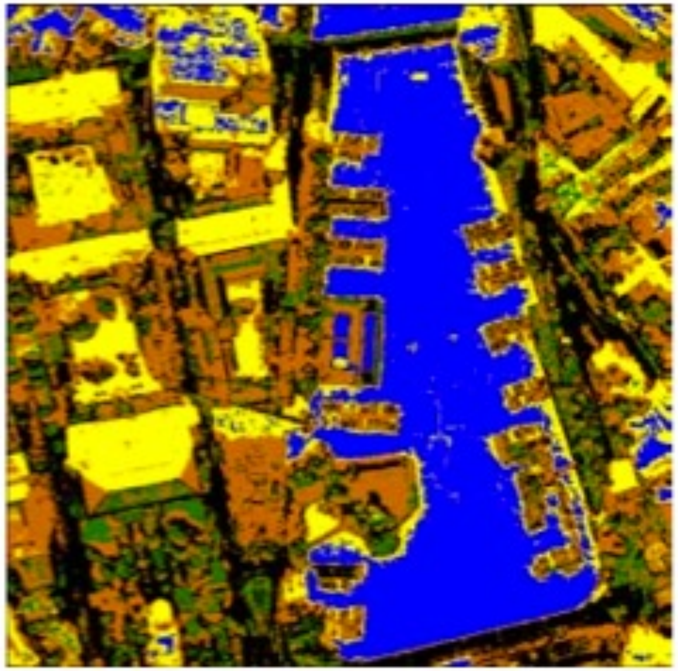
- K-folds cross-validation



Some good practices

- Overfitting
- Crossvalidation
- Spatial splits
- Accuracy measures

Am I doing a good job?

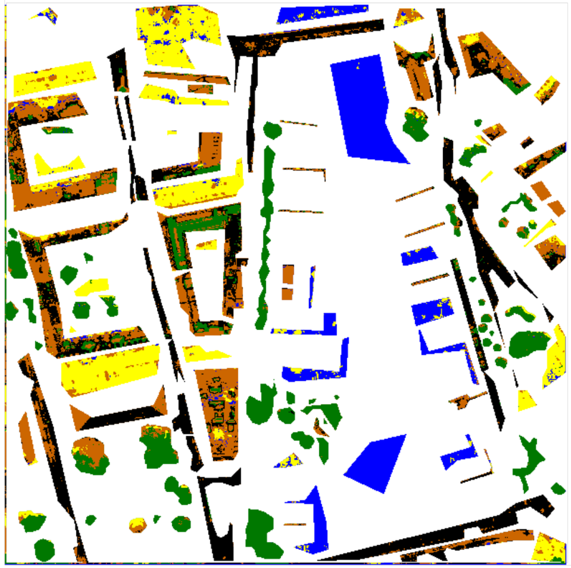


Predicted classes

True labels

EPFL

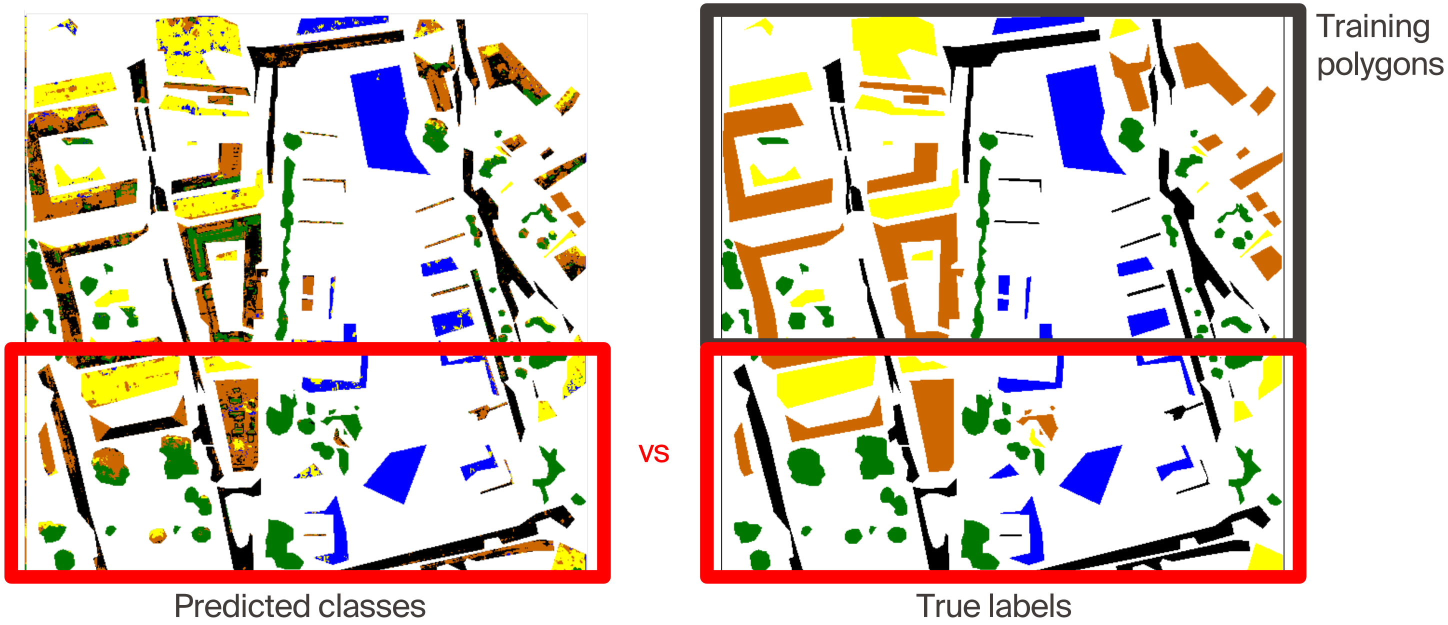
We can compare our predictions to the true labels



Predicted classes

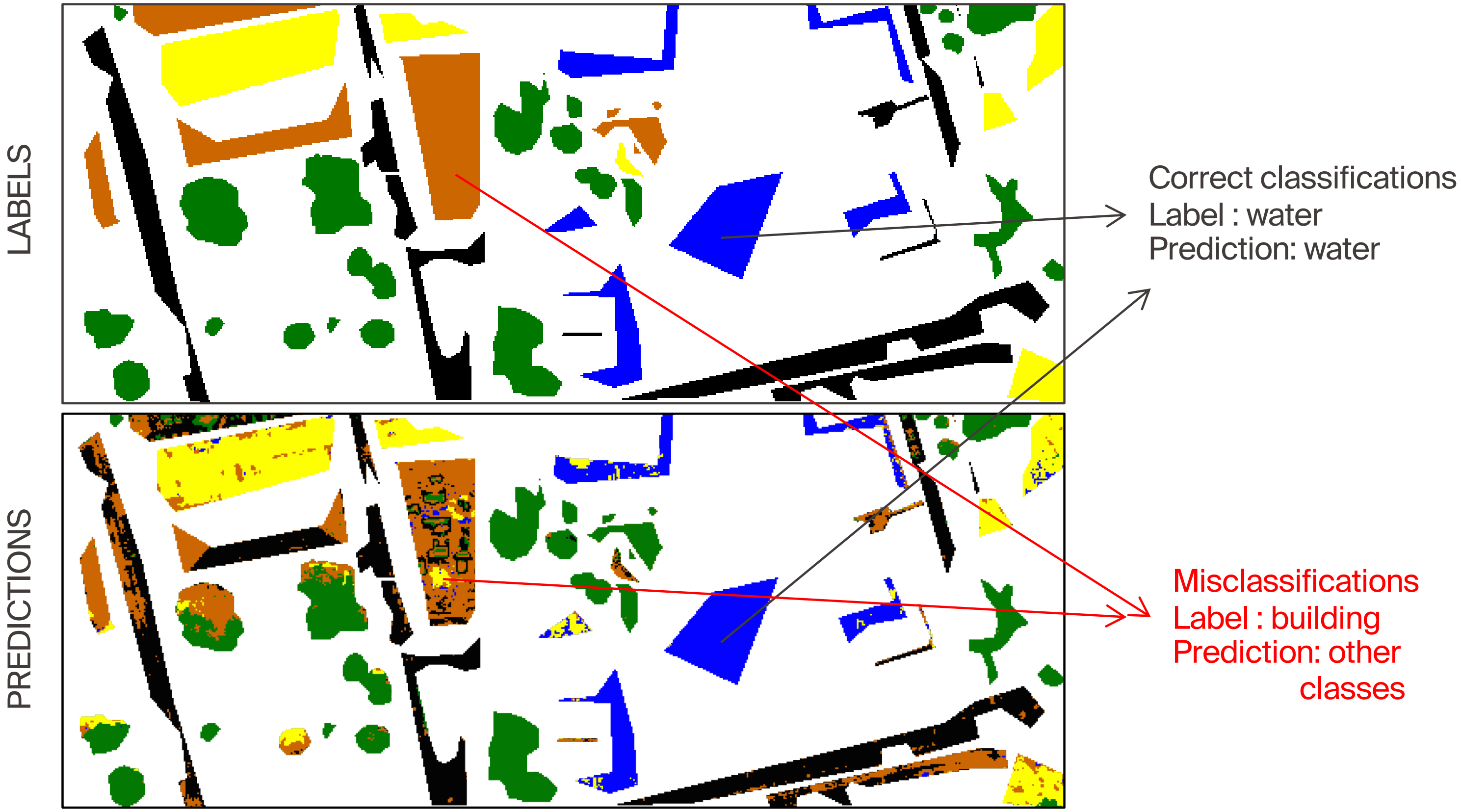
True labels

And we better do it only on areas we didn't use for establishing the model (to avoid positive biases)



EPFL

Then we compare the true and predicted pixel by pixel



Confusion matrix

Relates known truth pixels and predictions

Sample from class C
wrongly predicted in class A

Sample from class A
Correctly predicted in class A

		Predicted labels y*				
		Class A	Class B	Class C	Class D	PA
True labels y	Class A	n_{AA}				$n_{AA} / n_{A\bullet}$
	Class B		n_{BB}			$n_{BB} / n_{B\bullet}$
	Class C	n_{CA}		n_{CC}		$n_{CC} / n_{C\bullet}$
	Class D				n_{DD}	$n_{DD} / n_{D\bullet}$
	UA	$n_{AA} / n_{\bullet A}$	$n_{BB} / n_{\bullet B}$	$n_{CC} / n_{\bullet C}$	$n_{DD} / n_{\bullet D}$	OA and AA

AccuracieS

- Overall accuracy: percentage of correct classifications

  = sum of the diagonal of the CM / num of pixels

- Average accuracy: average of the per-class accuracies

  = the honest one. If you fail on one small class, it is going to show!

- User's accuracy: commission errors, % that a pixel classified into a class belongs to that class.

  = sum of the column sums of the CM

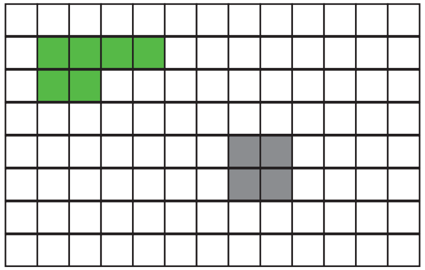
- Producer's accuracy: omission errors, % that a reference sample has been classified correctly.

  = sum of the row sums of the CM

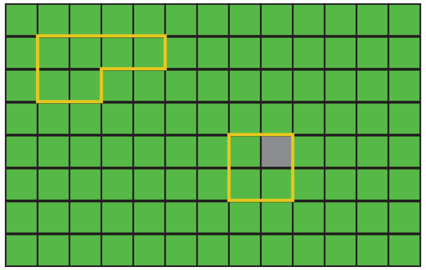
D. Tuia

User vs producer's

Ground truth



Your map



Predicted

		F	U	
True	F	6	0	6/6*100 = 100%
	U	3	1	1/4*100 = 25%
		6/9*100 = 66%	1/1*100 = 100%	

Producer Accuracy
"how many of true GT has been predicted correctly"

User Accuracy
"how many of the classifier's predictions were correct"