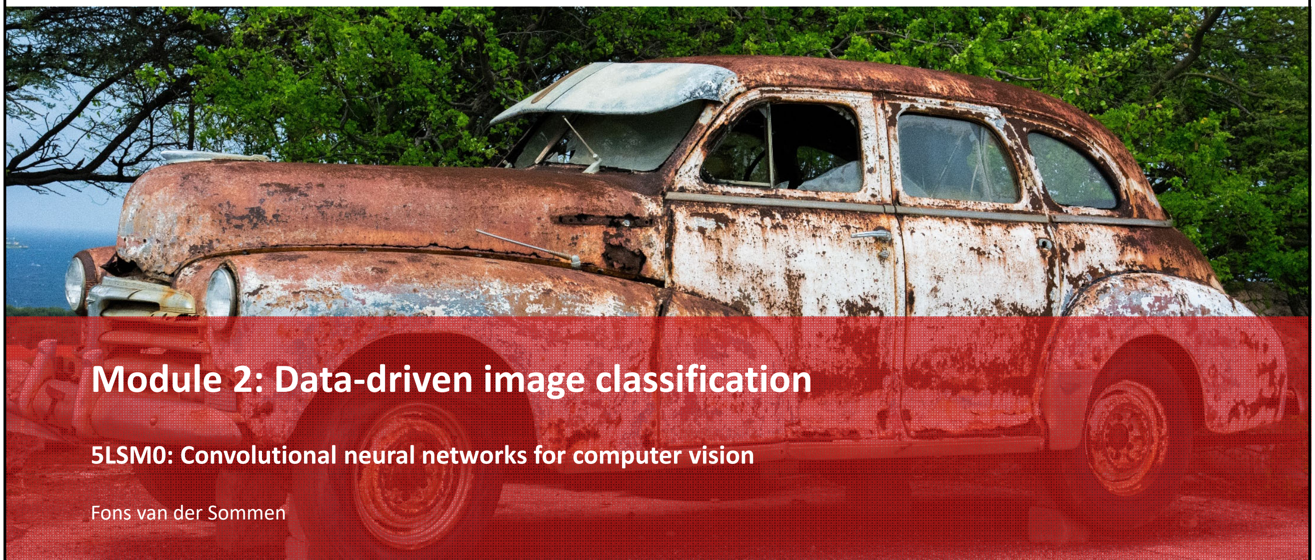
EINDHOVEN UNIVERSITY OF TECHNOLOGY



Module 2: Data-driven image classification

5LSM0: Convolutional neural networks for computer vision

Fons van der Sommen

Electrical Engineering / VCA research group

VCA Video Coding & Architectures research group

Last time

Image analysis is generally pretty hard!

- Enormous variation in appearance / illumination / viewpoint

Computer vision

- Can we make a computer see?

Deep learning

- Revolutionized machine learning/computer vision
- Convolutional neural network

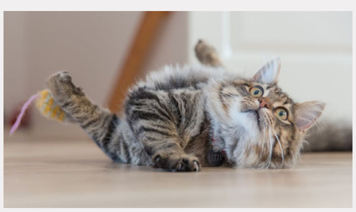
Computer vision subdomains

- Classification, Segmentation, Detection, Tracking, Re-identification

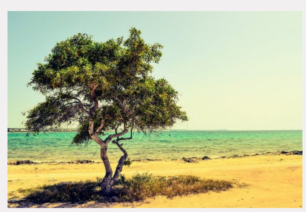
Module outline

- What is image classification?
- Data driven image classification
- Linear image classification
- Feature-based image classification
- Model performance evaluation

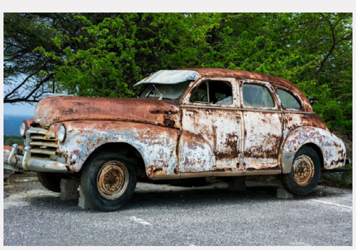
What is image classification?



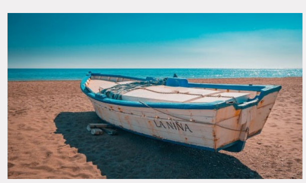
Cat



Tree



Car



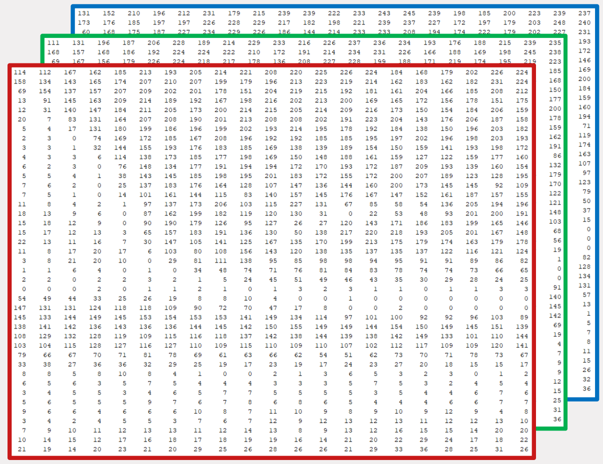
Boat

Sea?

Beach?

Sky?

What is image classification?



- An image is just a set of numbers
- An $N \times M$ matrix for each color channel
  - Typically Red Green and Blue (RGB)

What is image classification?

Why is this hard for a computer?

- Example: let's make an "owl detector"

What is image classification?

Owls…

- come in different sizes;

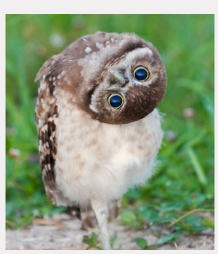



What is image classification?

Owls…

- come in different sizes;
- can deform;

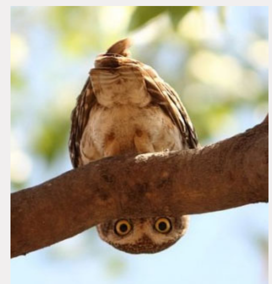


What is image classification?

Owls…

- come in different sizes;
- can deform;
- can be occluded or look like background;



What is image classification?

Owls…

- come in different sizes;
- can deform;
- can be occluded or look like background;
- there are @#$% butterflies that look like owls…

What is image classification?

Owls…

- come in different sizes;
- can deform;
- can be occluded or look like background;
- there are @#$% butterflies that look like owls…

Coming up with a set of heuristic rules for "Owlness"

=

Impossible!!!

Or at least very impractical…

What is image classification?

SOME of the challenges of image classification

- Viewpoint alteration - Occlusion
- Illumination - Intraclass variation
- Deformation - Background clutter

Instead of making a set of heuristic rules describing the "owlness" of an image…

➢ **learn the "owlness" of an image from examples (data-driven)**

Data-driven image classification

Data-driven image classification

The data-driven approach

1. Collect a set of example images and corresponding labels
2. Use machine learning to train a classification model using part of the images (training set)
3. Evaluate this model on new, unseen images (test set)

What you COULD do *(but probably shouldn't)* **:**

- Store all data points and labels
- Predict the label based on the label(s) of the most similar training image(s)
- **K-Nearest Neighbors (KNN)**

Data-driven image classification

Example on CIFAR10*

- 10 classes
- 50,000 training images
- 10,000 test images

Test image → nearest neighbors

Example from cs231n (2017): lecture 2 – slide 19

Data-driven image classification

How do we find the most similar images?

- Simple option: L1-distance

$$d_1(I_1, I_2) = \sum_{p \in I_1, I_2} \left| I_1^p - I_2^p \right|$$

Query (test) image

34	21	14	18	22
62	52	55	64	76
200	152	81	101	80
194	186	189	175	152
212	235	241	250	255

−

Training image

22	29	16	18	22
66	51	65	62	71
188	142	75	95	76
192	188	195	179	155
215	237	255	255	255

=

Pixel-wise absolute difference

12	8	2	0	0
4	1	10	2	5
12	10	6	6	4
2	2	6	4	3
3	2	14	5	0

Sum = 123

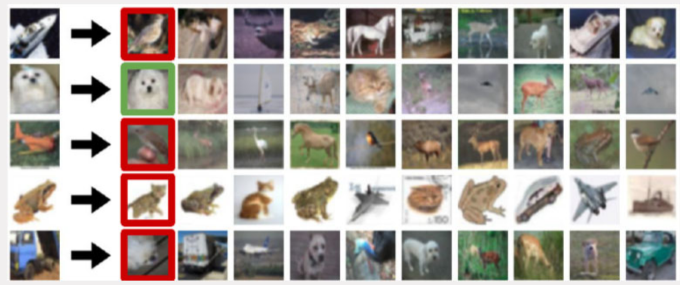
Data-driven image classification

K-nearest neighbors (KNN)

- **Q:** With N examples, how fast are training and prediction?
  - ***A:*** *Training: $\mathcal{O}(1)$, predicting: $\mathcal{O}(N)$ → Fast training, slow prediction*
  - *This is – exactly – what you **don't** want…*

How accurate is this approach?

- Not at all!
- Very sensitive to previously stated challenges (illumination, viewpoint, etc.)
- Better approach:
  - *Use "features" rather than raw pixels!*
  - *Use a different classification method!*



**Example from cs231n (2017): lecture 2 – slide 19*

Linear classification

Linear image classification

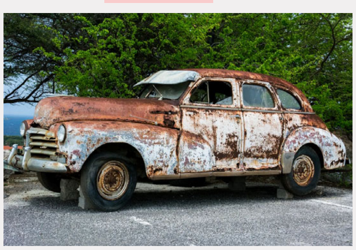
What is linear classification?

- Separate classes in the data using a “line” (hyperplane) [in *feature space*]
- Relation between input data (x) and output label (y) is linear
- Generally not the most powerful approach
  - *But fast and straightforward!*
- Can be used as building block for more classification methods
  - *Neural networks, Boosting, Random forests, …*

Linear image classification

Parametric approach

Image



Array (matrix) of
256 × 170 × 3 numbers

Function

$f(\mathbf{x}, \mathbf{W})$

Vectorized image

Parameters / weights

Prediction

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Class scores for
N classes

Q: Size of $\mathbf{W}$?

Linear image classification

Parametric approach

Image

Array (matrix) of
170 × 256 × 3 numbers

Function

$$f(\mathbf{x}, \mathbf{W})$$

Prediction

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Linear classification:

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Q: Size of $\mathbf{W}$?

Linear image classification

Parametric approach

Image

Function

Prediction

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 0.2 & 0.4 & -0.1 & 0.6 \\ 0.3 & 0.8 & 0.2 & -0.1 \\ -0.5 & 0.6 & 0.4 & -0.7 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.5 \\ -2.1 \end{bmatrix} = \begin{bmatrix} 3.6 \\ 2.6 \\ -3.0 \end{bmatrix} \begin{matrix} \text{Car} \\ \text{Boat} \\ \text{Tree} \end{matrix}$$

Class scores

Linear image classification

Mathematical interpretation

- Each score is the inner product between two vectors (+offset)
  - $\mathbf{x}$ *: the input image*
  - $\mathbf{w}_n$*: the basis vector for class* n *(representation)*
- $\mathbf{w_1 x} = |\mathbf{w_1}||\mathbf{x}|cos(\theta)$,
  - *Where* θ *is the angle between* $\mathbf{w_1}$ *and* $\mathbf{x}$

Q: What do we get if $\mathbf{w_1}$ points in the same direction as $\mathbf{x}$?

- This is just a mapping on a new basis
  - *Where each dimension represents a class*

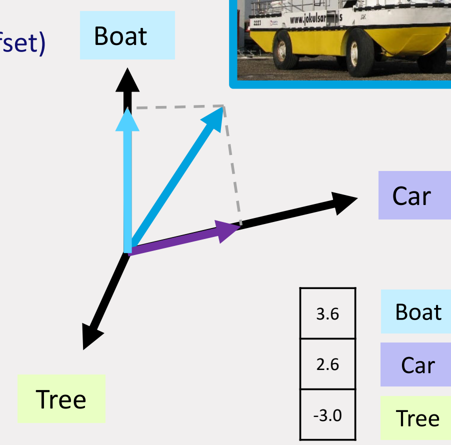
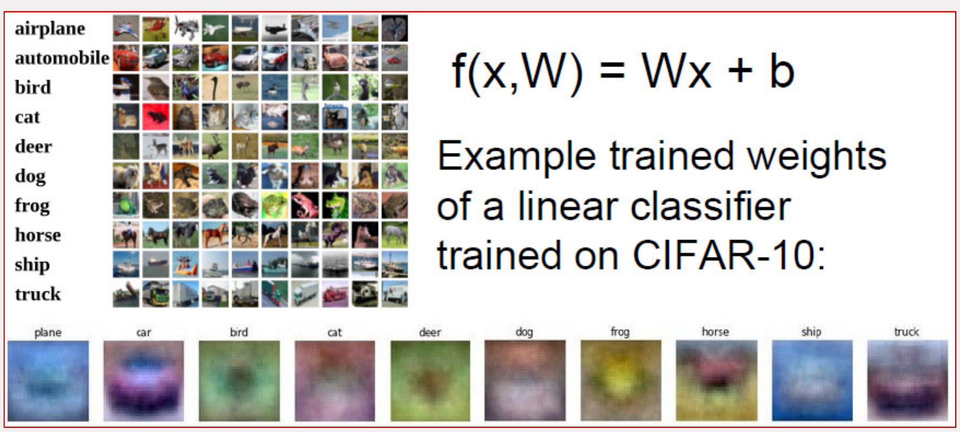


*Image from: https://www.flickr.com/photos/observethebanana/510823017

Linear image classification

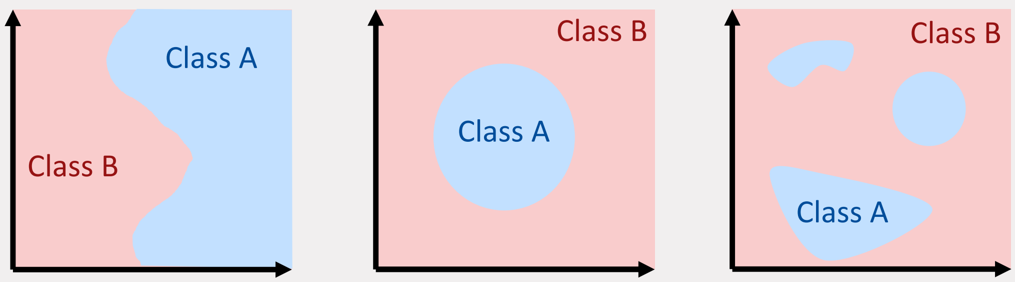
Each class will have one basis vector



**Example from cs231n (2017): lecture 2 – slide 57*

Linear image classification

Limitations of linear classification



Linear image classification

Summary

- We can use a linear function to generate class scores for images
- This function uses a set of basis vectors, in matrix W, to map the image to another space

All nice, but…

- How can we find W?
- How good is any given W?
- Next lecture!

	Score vectors		
Airplane	-5.1	2.1	2.0
Automobile	-2.0	3.6	0.1
Bird	-0.5	-2.1	-1.3
Cat	6.2	-1.4	-2.2
Deer	8.1	-3.6	1.9
Dog	2.3	-1.9	3.3
Frog	1.0	-8.0	0.1
Horse	0.2	-4.2	2.8
Ship	-0.2	1.1	-3.7
Truck	-1.3	4.1	-4.4

**Example from cs231n (2017): lecture 2 – slide 57*

Feature based image classification

Feature-based image classification

Generally not optimal to work on the raw data

- Very sensitive to changes in viewpoint, illumination, scaling, rotation, etc.
- Super high dimensional: curse of dimensionality

Better idea: use a low(er)-dimensional mapping of the original data

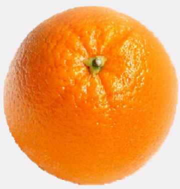
- Summarize the image into a set of descriptive features (lines, corners, colors, texture, …)
- Enables training relatively simple and robust classification models

Concept also used in neural networks

- Use an encoding scheme to obtain a representation in a latent (feature) space
- Similar to image compression!

Feature-based image classification

Linear classification example: separate lemons from oranges

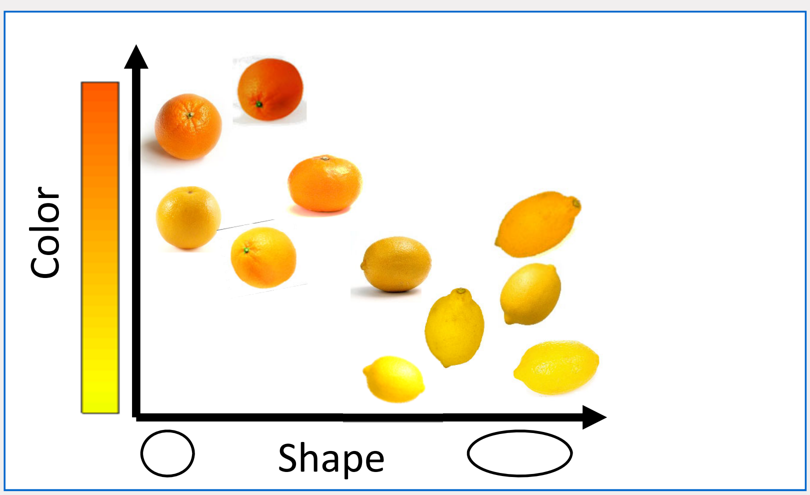


Color: orange
Shape: sphere
Diameter: ± 8 cm
Weigth: ±0.1 kg

Color: yellow
Shape: elipsoid
Diameter: ± 8 cm
Weigth: ±0.1 kg

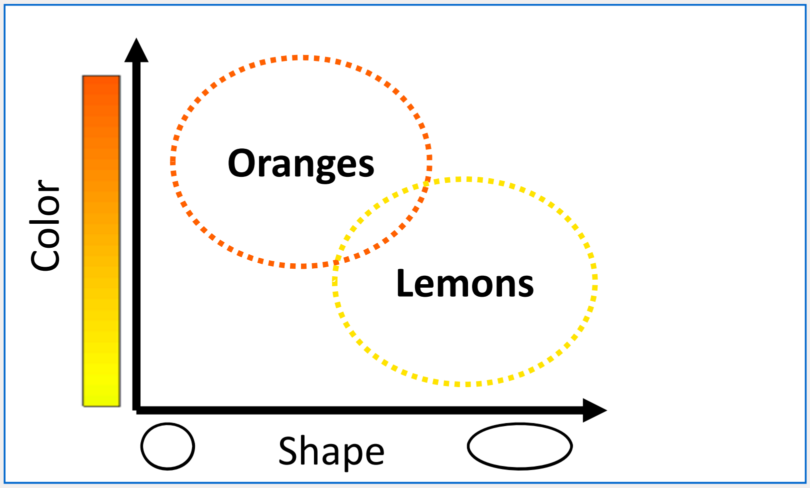
→ Use "color" and "shape" as features

Feature-based image classification



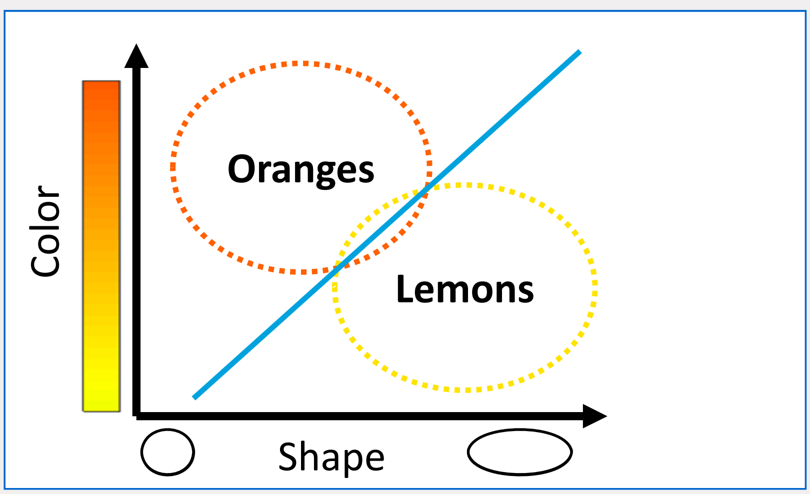
Model the given
- *training* - data

Feature-based image classification



Model the given
- *training* - data

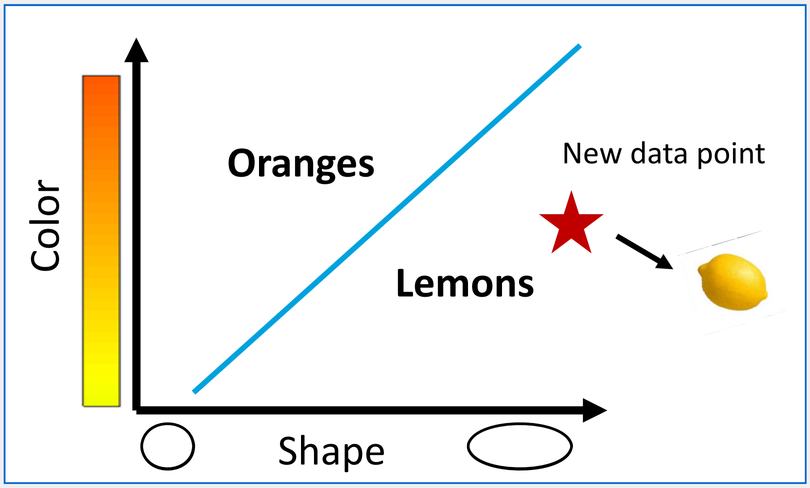
Feature-based image classification



Model the given
- *training* - data

5LSM0 Module 2: Data-driven image classification

Feature-based image classification



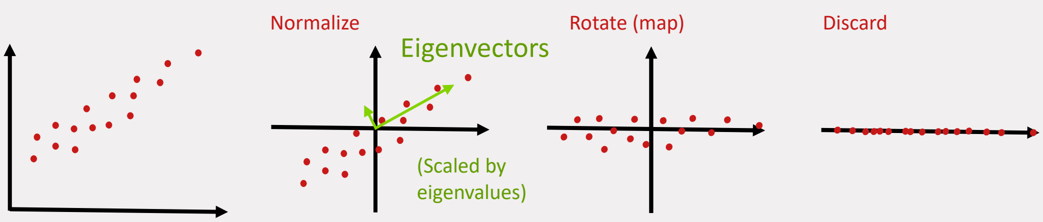
Model the given - *training* - data

Classifier:
"It's a lemon!"

Feature-based image classification

Reduce dimensionality using Principal Component Analysis (PCA)

- Find an orthonormal basis for the data
  - *By means of Eigenvalue Decomposition (EVD) or Singular Value Decomposition (SVD)*



- Rank the basis vectors according to their eigen-/singular values
- Rotate the feature vectors (map to new basis)
- Use the first P<N dimensions of the data

Very similar to linear image classification!

Feature-based image classification

Conventional (hand-crafted) image features

- Statistical measures on the Gray-Level Co-occurrence Matrix (GLCM) [Haralick, 1973]
  - *Statistics on local contrast relations*
- Local Binary Patterns (LBP) [Ojala et al., 1996, 2002]
  - *Statistics of local contrast transitions*
- Histogram of Oriented Gradients (HOG) [Dalal & Trigs, 2005]
  - *Capture statistics of local image gradients*
- …

Feature-based image classification

Statistical measures on the Gray-Level Co-occurrence Matrix (GLCM)

3	0	1	2
1	2	3	0
0	0	1	3
0	2	1	2

x=1, y=0

	0	1	2	3
0	1	2	1	0
1	0	0	3	1
2	0	1	0	1
3	2	0	0	0

Statistical measures

Homogeneity	$\sum_i \sum_j \frac{P(i,j)}{1+\lvert i-j \rvert}$
Contrast	$\sum_i \sum_j (i-j)^2 P(i,j)$
Energy	$\sum_i \sum_j P(i,j)^2$
Dissimilarity	$\sum_i \sum_j P(i,j)\lvert i-j \rvert$
Entropy	$-\sum_i \sum_j P(i,j)\log(P(i,j)+\varepsilon)$
Correlation	$\sum_i \sum_j \frac{(i-\mu_x)(i-\mu_y)P(i,j)}{\sigma_x \sigma_y}$

Summarize in N-dimensional vector

92	30	75	84	10	33	…	3	2	7	6	4	5	4

Number of parameters:

- Binsize
- Displacements
- Directions

Feature-based image classification

Local Binary Patterns (LBP)

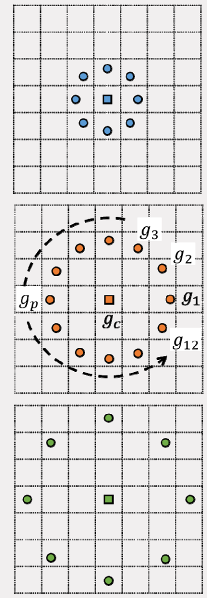
- Define some neighborhood around center pixel g_c
- Assign a code to each pixel position

$$C_{LBP}(m,n) = \sum_{n=1}^{N} s(g_n - g_c)2^{n-1}, \qquad s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 1 \end{cases}$$

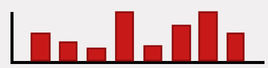
3	0	1	2
1	2	3	0
0	0	1	3
0	2	1	2

Neighbor #	1	2	3	4	5	6	7	8
g_n	0	3	1	0	0	1	3	1
$s(g_n - g_c)$	0	1	0	0	0	0	1	0
2^{n-1}	1	2	4	8	16	32	64	128

$C_{LBP}(2,2) = 66$



- Compute the histogram of these code vectors.



Feature-based image classification

Histogram of Oriented Gradients (HOG)

- Obtain the gradient of the image

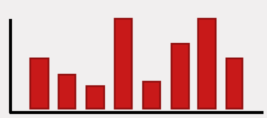
$G_x = \mathbf{w}_x \star I(x,y)$ $\qquad G_y = \mathbf{w}_y \star I(x,y)$ $\qquad \mathbf{w}_x = \mathbf{w}_y^T = [\ -1 \quad 0 \quad 1\ ]$

- Compute the angle and the magnitude

$\theta = \arctan\left(\frac{G_y}{G_x}\right)$ $\qquad M(x,y) = \sqrt{G_x^2 + G_y^2}$

- Bin angles and make a histogram using the magnitudes

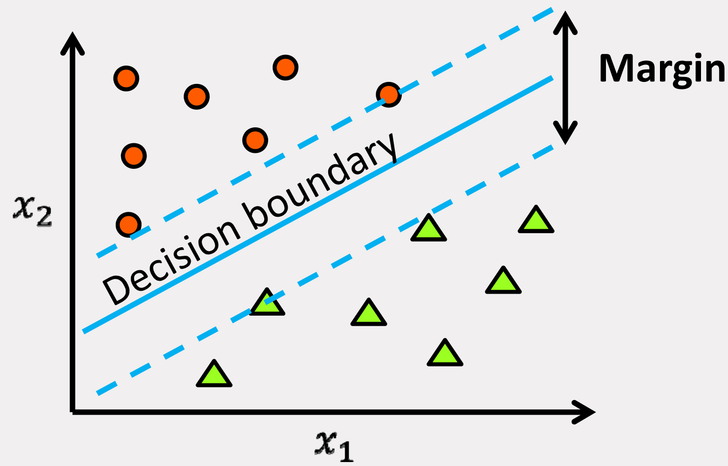
$$B_n = \sum_{\theta(x,y)\in\Theta_n} M(x,y) \Longrightarrow$$



Feature-based image classification

Classification models

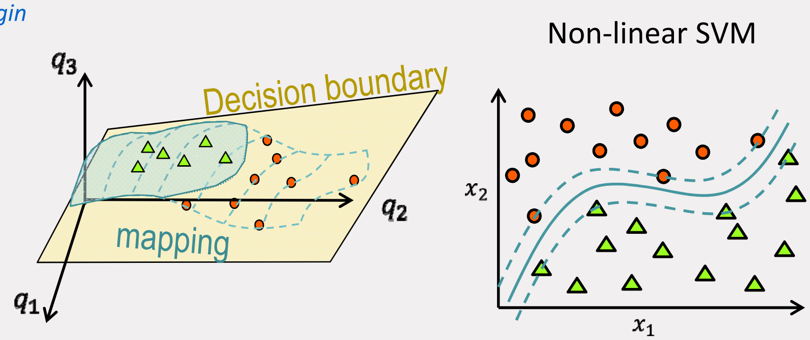
- Support Vector Machine (SVM)
  - *Binary classification model*
  - *Separate data with largest margin*



Feature-based image classification

Classification models

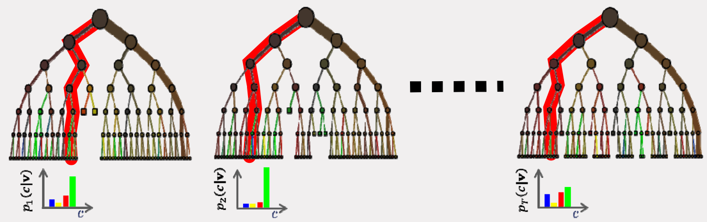
- Support Vector Machine (SVM)
  - *Binary classification model*
  - *Separate data with largest margin*

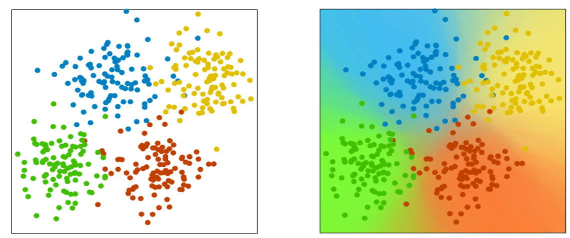


Feature-based image classification

Classification models

- Support Vector Machine (SVM)
  - *Binary classification model*
  - *Separate data with largest margin*
- Random forest
  - *Ensemble of decision trees*
  - *Increase robustness by randomness*

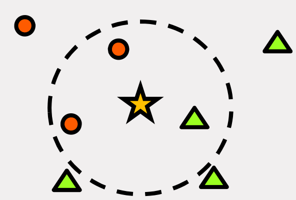


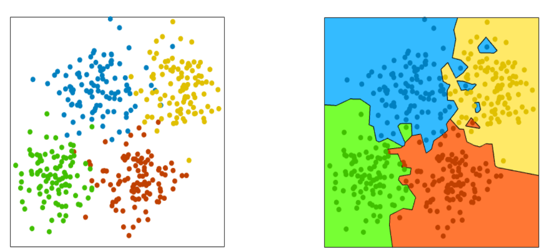


Feature-based image classification

Classification models

- Support Vector Machine (SVM)
  - *Binary classification model*
  - *Separate data with largest margin*
- Random forest
  - *Ensemble of decision trees*
  - *Increase robustness by randomness*
- K-Nearest Neighbours (KNN)
  - *No explicit modeling: store all training data*
  - *Prediction = majority class of N nearest neighbors*



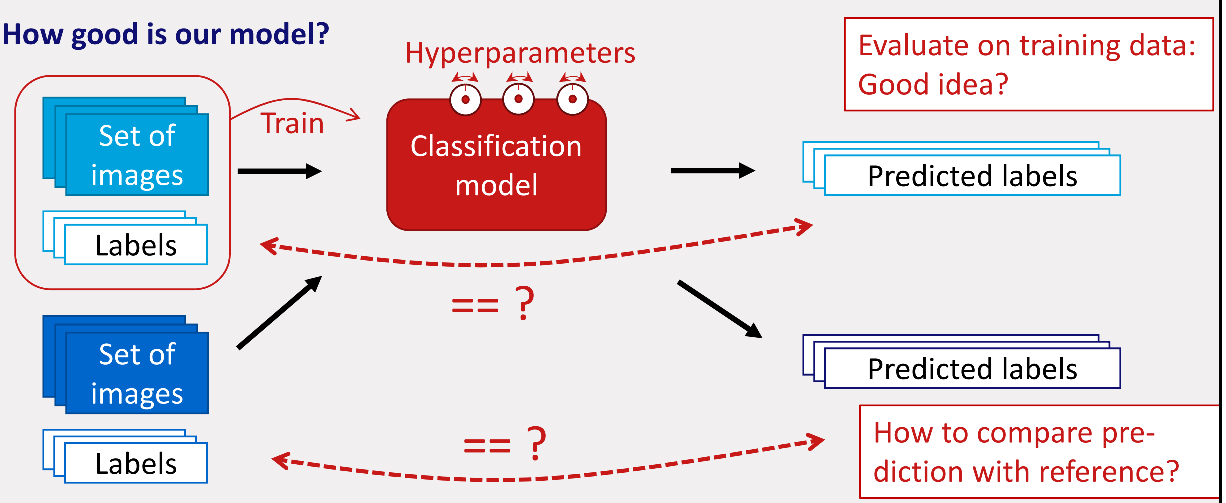


Model performance evaluation

5LSM0 Module 2: Data-driven image classification

Model performance evaluation

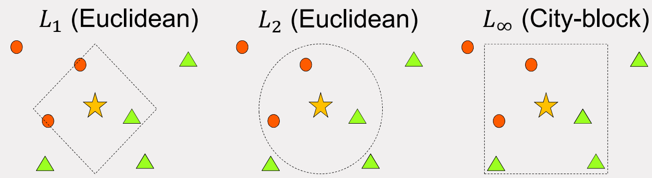
How good is our model?

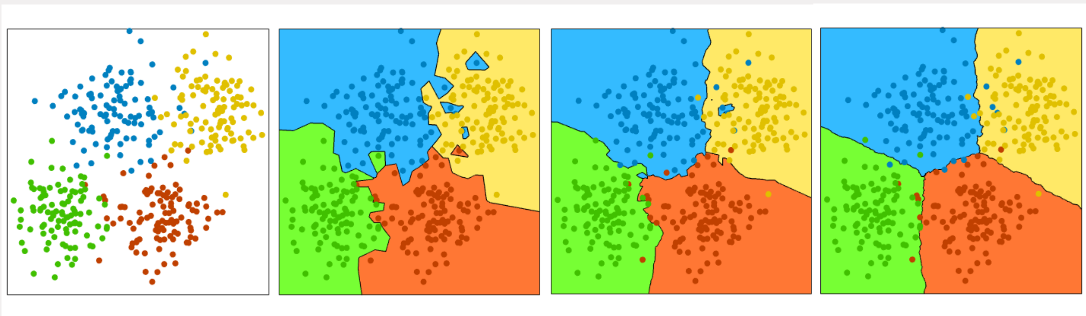


Model performance evaluation

Hyper-parameters example: KNN

- Distance metric
- Number of neighbours





Model performance evaluation

How to measure the model performance?

- Binary classification
- N-ary classification
- Segmentation

How to split the data?

- Validation methods
- Stratification
- Hyperparameters

Measuring performance

Some popular figures of merit:

- Accuracy, Sensitivity / Specificity, Precision / Recall
- F-score
- Area Under the [ROC] Curve (AUC)
- DICE coefficient / Jaccard index

Popular visualizations

- Confusion matrix
- Receiver Operating Characteristic
- Boxplots and error bars

Measuring performance: binary classification

Some nomenclature

- Define one class as "positive"

		Prediction	
		Positive	***Negative***
Ground truth	***Positive***	***True positive***	***False negative***
	Negative	***False positive***	***True negative***

Measuring performance: binary classification

Some nomenclature

- Define one class as "positive"

EXAMPLE:

Predict if a patient has a certain disease

Positive = disease

Negative = no disease

Ground truth \ Prediction	Positive	Negative
Positive	***True positive*** Patient has the disease and the algorithm has detected it	***False negative*** Patient has the disease but the algorithm has not detected it
Negative	***False positive*** Patient has no disease but the algorithm detects disease	***True negative*** Patient has no disease and the algorithm does not detect disease

Measuring performance: binary classification

Some abbreviatons

- #TP: number of true positives
- #TN: number of true negatives
- #FP: number of false positives
- #FN: number of false negatives
- #P: number of positives *(#P = #TP + #FN)*
- #N: number of negatives *(#N = #TN + #FP)*
- #S: total number of samples *(#S = #TP + #TN + #FP + #FN)*

Measuring performance: binary classification

Accuracy

- How well can our model predict the correct class of a sample?
- Number of correct predictions over the number of samples

$$Accuracy = \frac{\#TP + \#TN}{\#S} = \frac{\#TP + \#TN}{\#P + \#N} = \frac{\#TP + \#TN}{\#TP + \#FP + \#TN + \#FN}$$

When does this metric fail to reflect the performance?

Measuring performance: binary classification

Sensitivity

- How well can our model find the positive samples?

$$Sensitivity = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP + \#FN}$$

Specificity

- How well can our model find the negative samples?

$$Specificity = \frac{\#TN}{\#N} = \frac{\#TN}{\#TN + \#FP}$$

Can we always compute the specificity?

Measuring performance: binary classification

Recall = Sensitivity

- What fraction of the positive samples can we find?

$$Recall = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP + \#FN}$$

Precision

- What fraction of our positive predictions is correct?

$$Precision = \frac{\#TP}{\#TP + \#FP}$$

When use Sens./Spec.?
When Prec./Recall?

Measuring performance: binary classification

F1-score

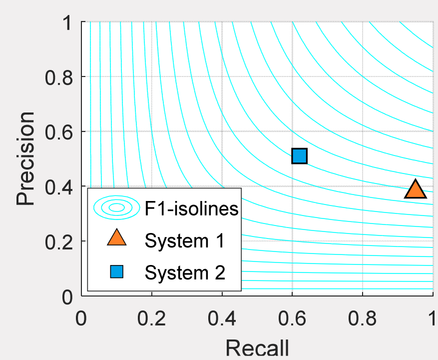
- Sensitivity/Specificity and Precision/Recall always yield two numbers, how can we compare results?

What do you think is better:

- *System 1: precision = 0.93, recall= 0.62;*
- *System 2: precision = 0.73, recall= 0.77;*

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$\boxed{F1_{sys1} = 0.54}$ $\boxed{F1_{sys2} = 0.56}$



Measuring performance: binary classification

Receiver Operating Characteristic (ROC)

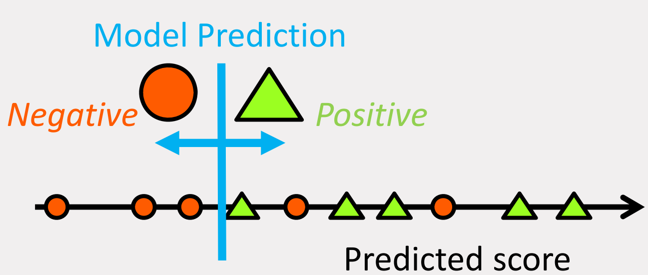
- Sensitivity / specificity give the performance for just one possible setting (i.e. decision threshold) of the model
- We can vary this threshold and recompute these performance metrics
- This yields a curve of possible combinations of sensitivity and specificity, called the ROC curve
- Generally true: ↑ sensitivity ↓ specificity and vice versa

How to compute the ROC curve?

- For each sample we have a predicted class and a score
- Sort the samples according to score and move the threshold

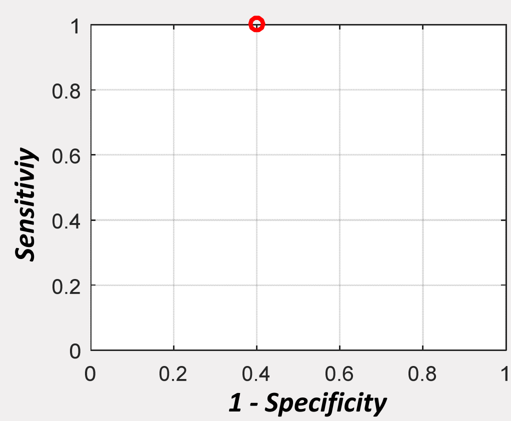
Measuring performance: binary classification

Computation of ROC curve



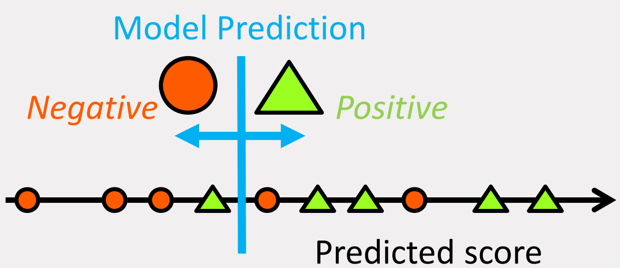
Sensitivitiy = #TP / (#TP + #FN) = 5 / (5+0) = 1.00

Specificity = #TN / (#TN + #FP) = 3 / (3+2) = 0.60



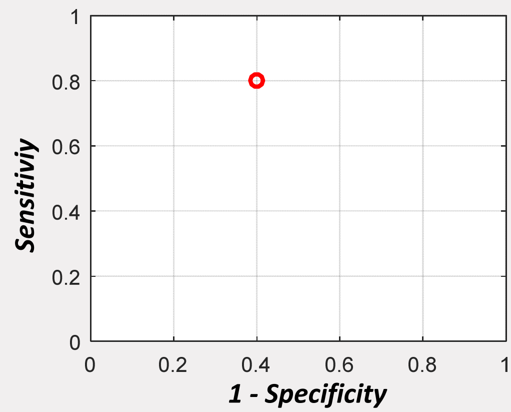
Measuring performance: binary classification

Computation of ROC curve



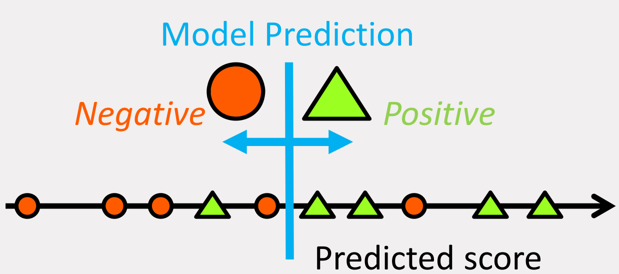
Sensitivitiy = 4 / (4+1) = 0.80

Specificity = 3 / (3+2) = 0.60



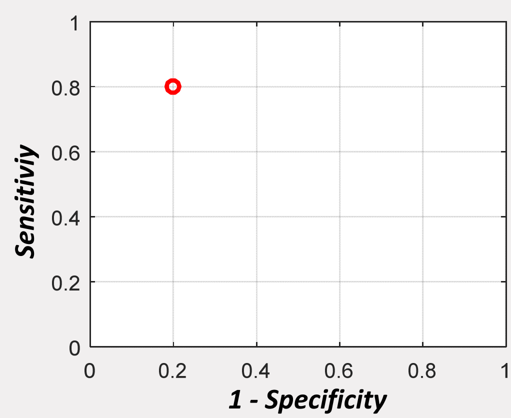
Measuring performance: binary classification

Computation of ROC curve



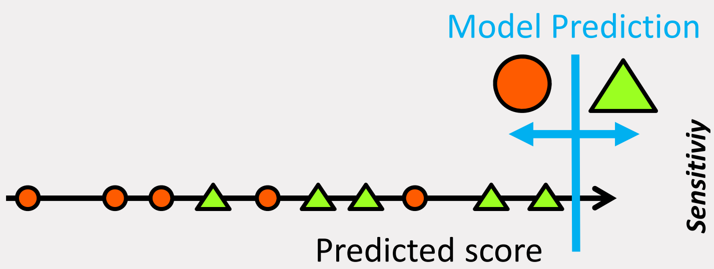
Sensitivitiy = 4 / (4+1) = 0.80

Specificity = 4 / (4+1) = 0.80



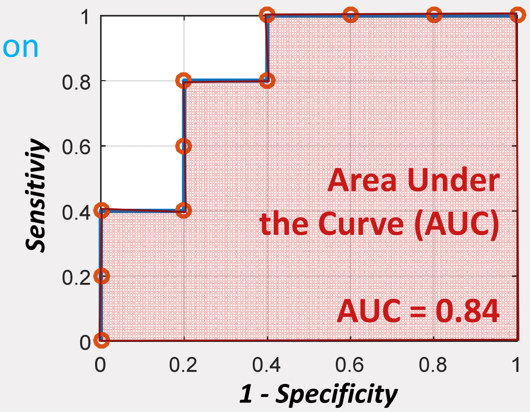
Measuring performance: binary classification

Computation of ROC curve



Sensitivitiy = 0 / (0+5) = 0.00

Specificity = 5 / (5+0) = 1.00



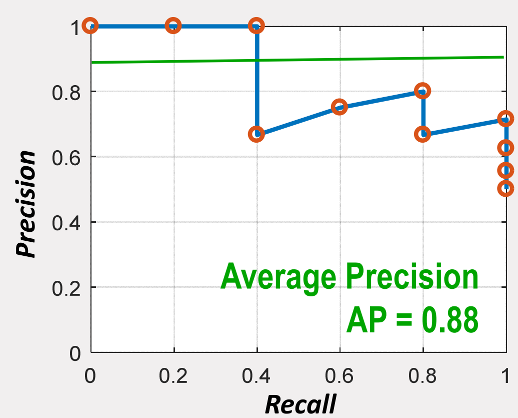
Measuring performance: binary classification

You can do the same thing for precision/recall

- Precision-Recall curve (PR)
- AUC for PR-curve is called Average Precision (AP)

$$AP = \int_{r=0}^{1} p(r)dr \approx \sum_{k=1}^{N} P(k)\Delta r(k)$$

$$\approx 1.00 \times 0.2 + 1.00 \times 0.2 + 0.67 \times 0 + 0.75 \times 0.2 \ldots$$
$$+0.80 \times 0.2 + 0.67 \times 0 + 0.71 \times 0.2 \ldots$$
$$+0.63 \times 0 + 0.56 \times 0 + 0.50 \times 0$$



Measuring performance: N-ary classification

How to compute Precision/Recall for N classes?

Macro- and micro averaging

- Compute #TP, #FP and #FN for each class individually

$$PRE_{micro} = \frac{\#TP_1 + \#TP_2 + \cdots + \#TP_K}{\#TP_1 + \cdots + \#TP_K + \#FP_1 + \cdots + \#FP_K}$$

$$REC_{micro} = \frac{\#TP_1 + \#TP_2 + \cdots + \#TP_K}{\#TP_1 + \cdots + \#TP_K + \#FN_1 + \cdots + \#FN_K}$$

$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_K}{K}$$

$$REC_{macro} = \frac{REC_1 + \cdots + REC_K}{K}$$

What's the difference?

Measuring performance: N-ary classification

Confusion matrix

Considering class 2, how can I get #TP and #FP from this table?

And #FN?

Prediction

Ground truth	*Class 1*	*Class 2*	*Class 3*
Class 1	#	#	#
Class 2	#	#	#
Class 3	#	#	#

Measuring performance: N-ary classification

How can we capture the performance in one number?

Mean average precision (MAP)

- Consider each class seperately as positive (all-vs-one)
- Compute Average Precision (AP) for each class
- Take the mean over all classes

Often used in image classification

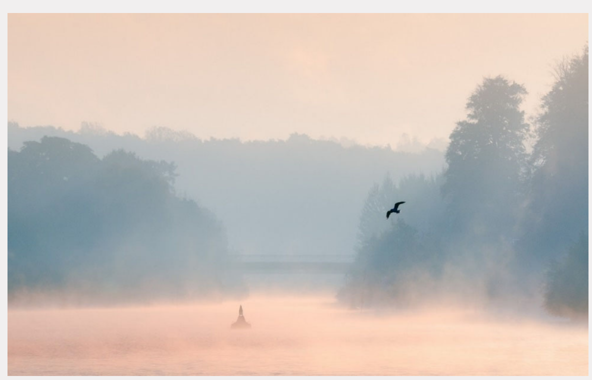
Measuring performance: Segmentation

Segmentation ≠ classification of pixels

- Do not use pixel-based metrics, they are misleading.
- Example:

Why never ever use accuracy?

Look at my bird-segmentation algorithm with a whopping 99.98% accuracy!!!



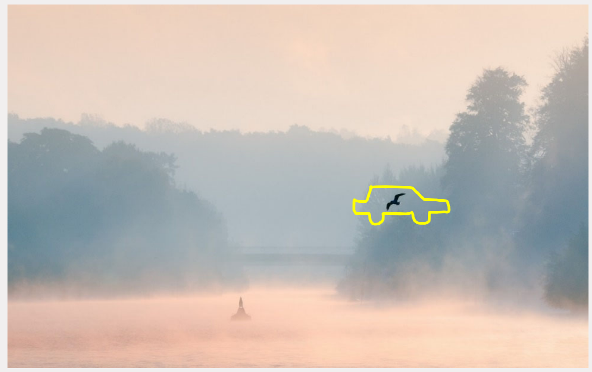
Measuring performance: Segmentation

Segmentation ≠ classification of pixels

- Do not use pixel-based metrics, they are misleading.
- Example:

Why not use specificity and sensitivity?

Ok, new segmentation algorithm:
sensitivity = 1.00,
specificity = 0.99,
Wow!!!



Data splitting

Option 1: Evaluate on training data — Bad idea!

Available data

Option 2: Evaluate on separate data set — Better idea…

Training | Test

Option 3: Tune hyper-parameters on validation set, then evaluate on test set

Training | Validation | Test

Good idea!

Data splitting

When you're done tuning your model, evaluate on the test set: external validation

Data splitting

Training | Validation | Test

But how to split training/validation? Can make a difference for small sets…

- K-fold cross-validation

Internal validation

Training | Validation → Score

Training | Validation | Training → Score

Training | Validation | Training → Score

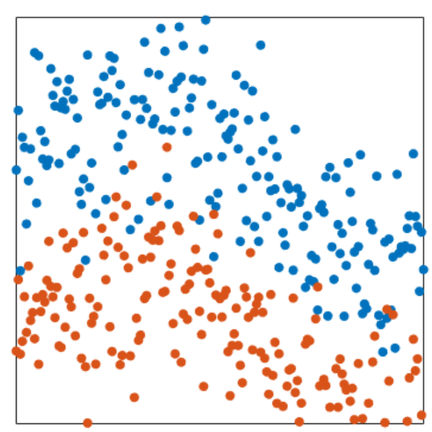
Validation | Training → Score

Average score

Data splitting

Why don't we evaluate on the training set?

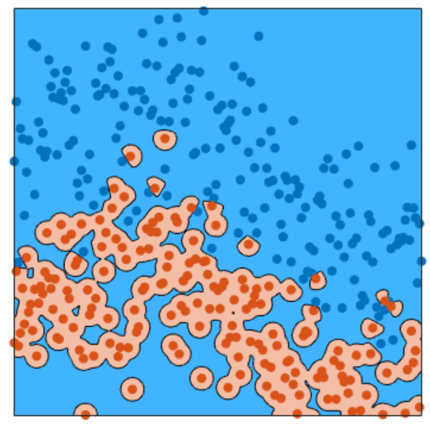
- Example:



Data splitting

Why don't we evaluate on the training set?

- Example:



Is this a good classifier?

- No errors on the training set!!!
- 100% accuracy

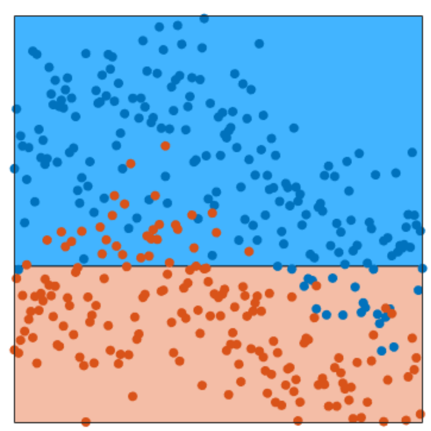
NO!

- Very poor generalization
- On new, identically distributed data:
  - 81% accuracy…
- **Overfitting!**

Data splitting

Why don't we evaluate on the training set?

- Example:



Is this a good classifier?

- Many errors on the training set…
- 86% accuracy

NO!

- Model complexity too low!
  - **Underfitting!**
- On new, identically distributed data:
  - 84% accuracy… (≈ train acc. !)

Data splitting

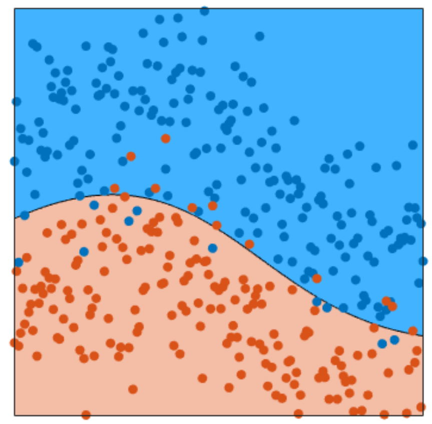
Why don't we evaluate on the training set?

- Example:

Is this a good classifier?

- Accuracy on trianing set: 94%
- Accuracy on test set: 95% ≈
- Approximately equal train and test error
  - **Good generalization**!

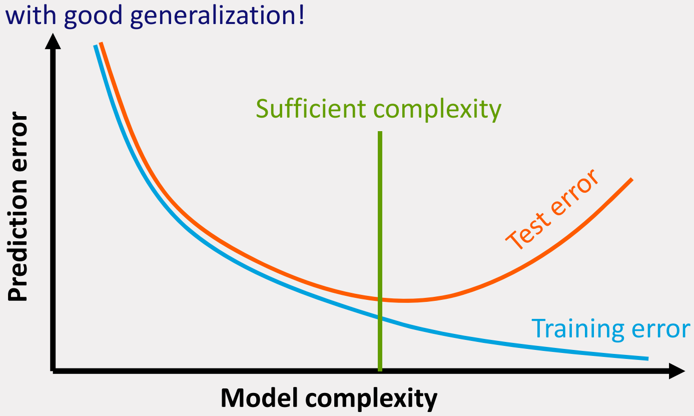
YES!



Model performance evaluation

Model complexity: what is a good model?

- A model with good generalization!

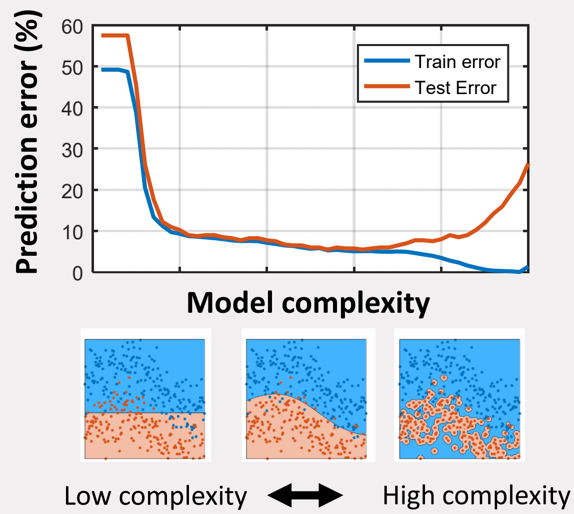


Good prediciton accuracy on both the training and the test set!

Model performance evaluation

Model complexity: what is a good model?

- Example:
  - *Non-linear SVM*
  - *Fixed cost parameter C*
  - *Complexity increases with reducing the size of the kernel scale (flexibility)*
  - *10-fold cross validation to estimate the test error*
  - *Validate on training set for computing the train error*



Summary

Image classification

- Typically very hard due to varying imaging conditions
- Manually making a set of heuristic rules that should be satisfied is infeasible

Data driven image classification

- We can learn “a set of rules” from the data!
- K-NN yields a result, but far from optimal

Linear image classification

- Classification problem can be formulated as a linear model
- Map the data to a set of (template basis) vectors

Summary

Feature-based image classification

- Instead of using the raw image pixels, extract discriminative features
- Separate the corresponding features in feature space by conventional classification models

Model performance evaluation

- Important to choose appropriate metrics that reflect the desired performance
- Use separate sets for training and testing a model
- Tune your model by means of internal validation, preferable using cross-validation
- A good model exhibits good generalization (no overfitting or underfitting)

Next time:

Loss functions

- What is a good weight matrix W and can we quantify this?

Optimization

- Given a way to measure the "quality" of W, how do we find the best W?