AMME4710: COMPUTER VISION AND IMAGE PROCESSING WEEK 7

Dr. Mitch Bryson

School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney

Last Week

- Image Segmentation
  - Top Down vs. Bottom up approaches
  - Clustering algorithms based on K-means and Mean Shift

This Week's Lecture

- Introduction to Machine Learning and Image Classification

- Learning Objectives:
  - To gain an understanding of image classification algorithms
  - To provide an introduction to machine learning and data-driven methods in computer vision

Image Classification

- Image classification is the process of assigning an image, or part thereof, to one of a number of discrete classes:

cats

dogs

Input image

Output label/class:
cat

Possible classes = {cat, dog}

- Possible classes are (typically) a finite set depending on the application: two-class classification is referred to as **binary classification**, or **multi-class classification** for more than two classes

Image classification and related tasks

- Classification is related to a number of different tasks in image interpretation:

Scene Classification/Categorisation

urban

Image classification and related tasks

- Classification is related to a number of different tasks in image interpretation:

Scene Classification/Categorisation

urban

Object Segmentation: determining the locations of objects within a scene/image

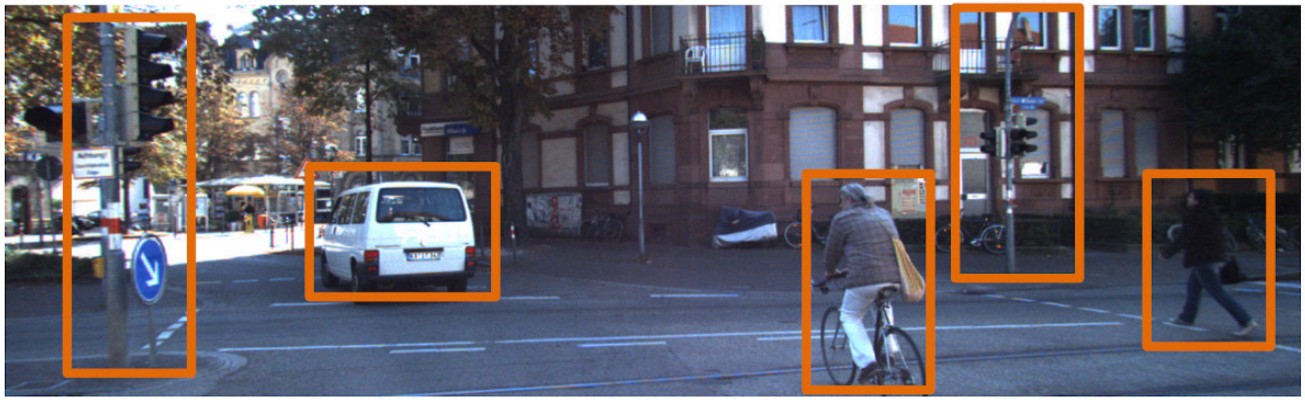


Image classification and related tasks

- Classification is related to a number of different tasks in image interpretation:

Object Classification: where are the objects, and what are they?

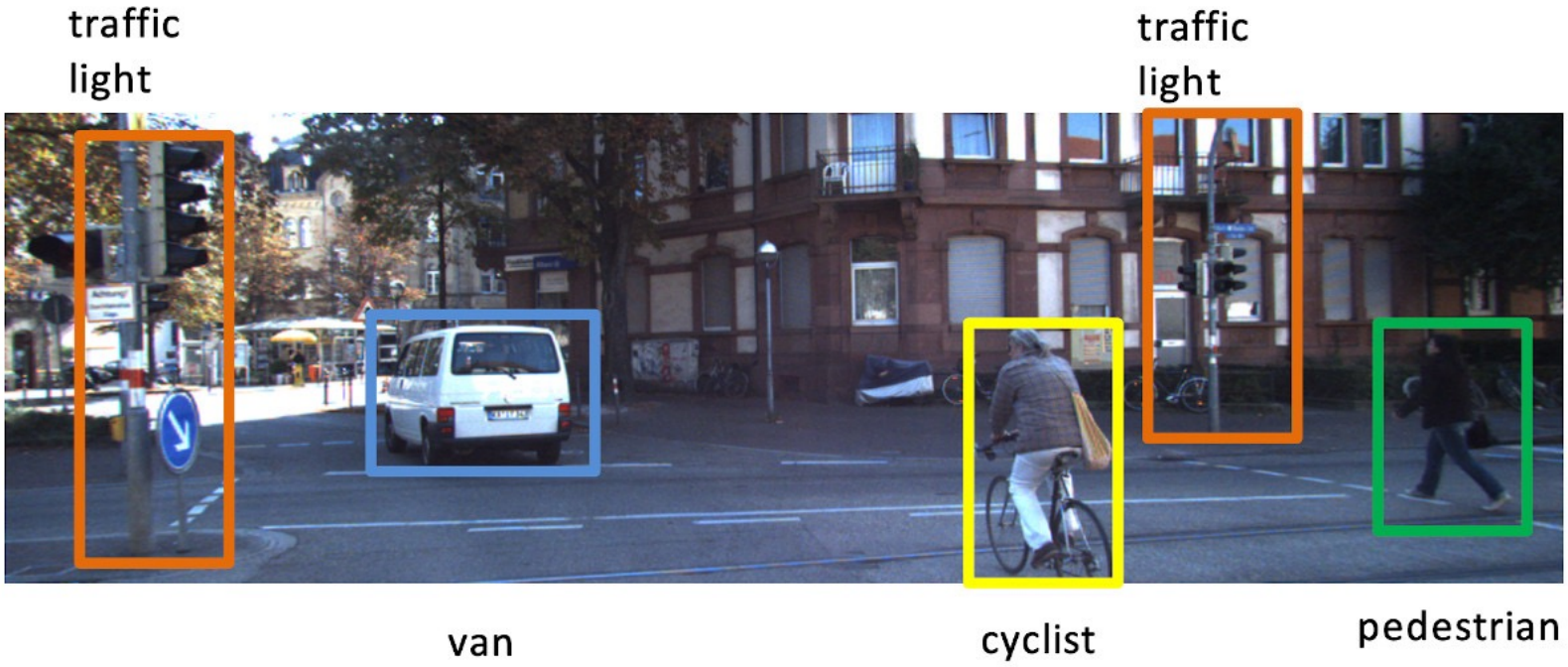
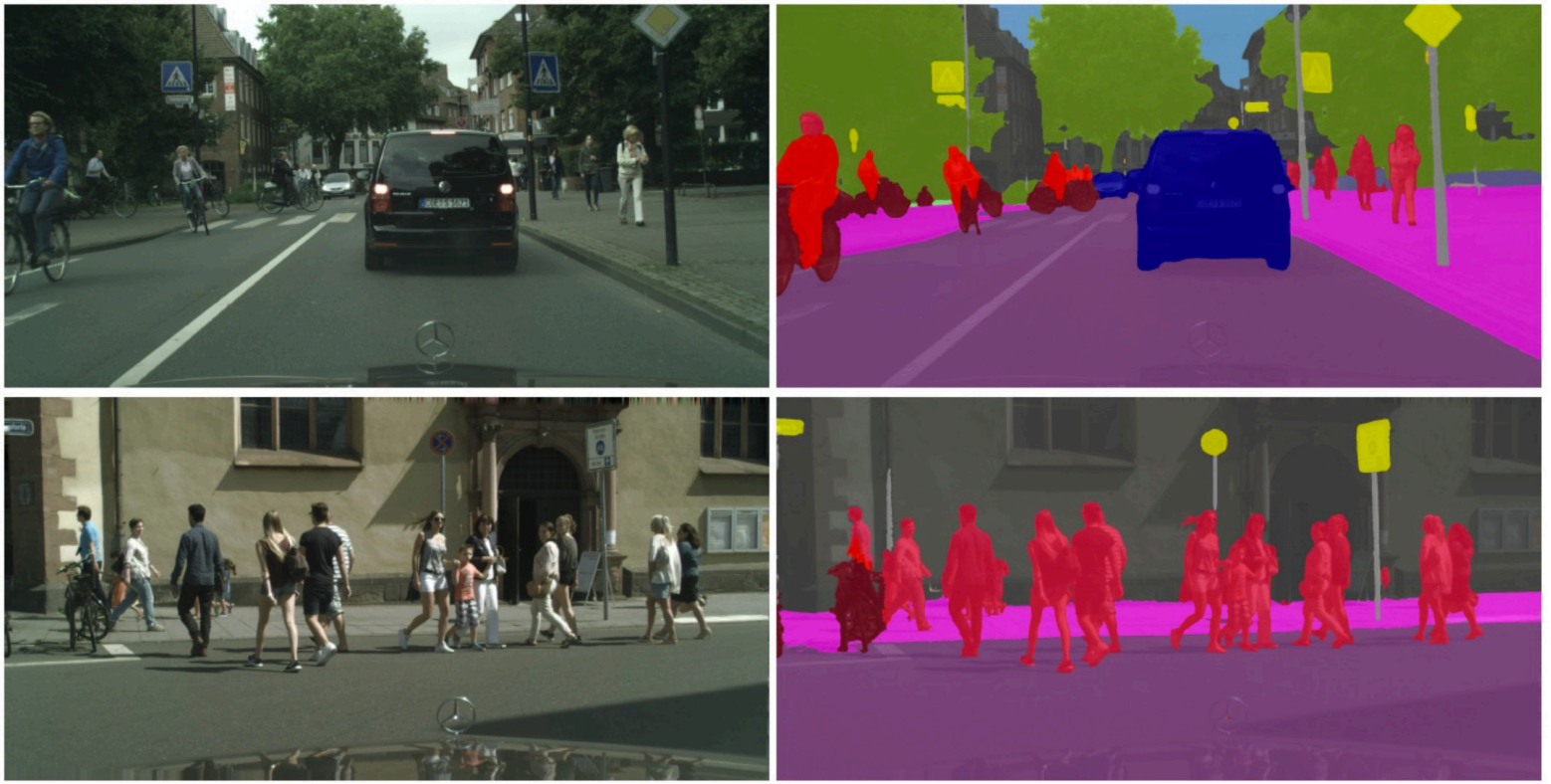


Image classification and related tasks

- Classification is related to a number of different tasks in image interpretation:

Semantic Segmentation: label each and every region/pixel of the image in one of several classes



Red: people, blue: cars, green: trees, yellow: signs, purple: road, pink: footpath

Image classification and related tasks

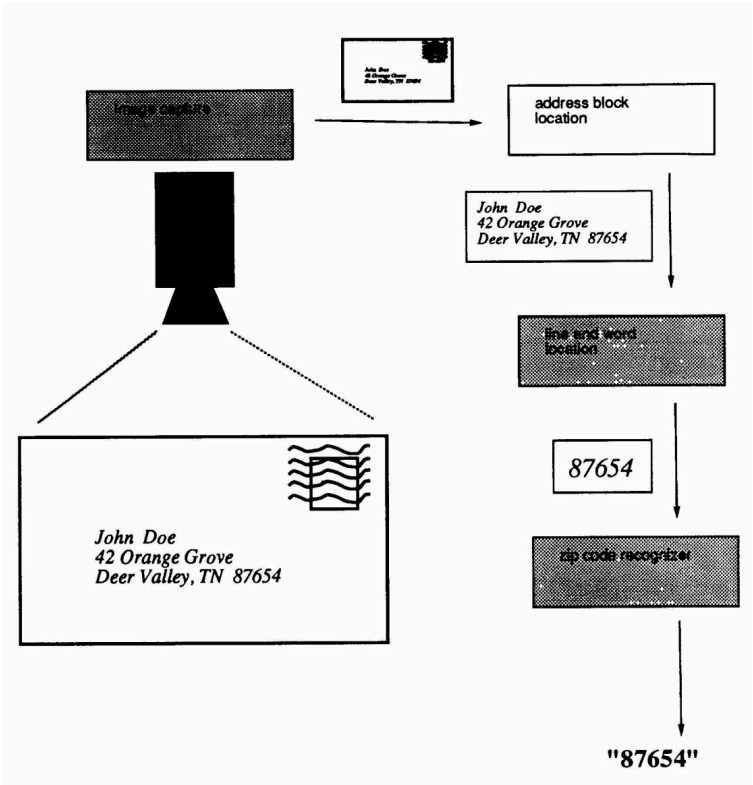
- Classification is related to a number of different tasks in image interpretation:

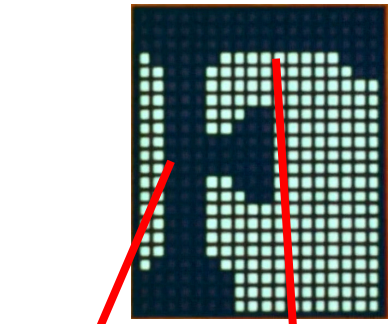
Instance Segmentation: label each and every region/pixel of the image in one of several classes, and separately for different instances of each class (i.e. separate people)

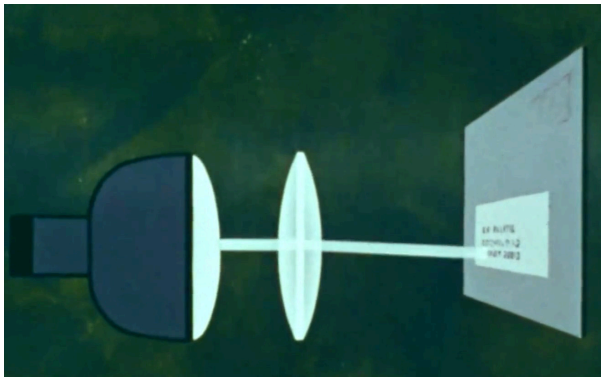


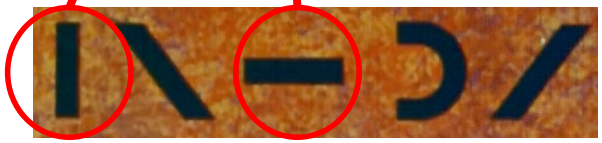
Early applications: post code recognition

- Optical Character Recognition (OCR) in 1970s for machine printed addresses
- Systems for interpreting hand-written digits in 1990s



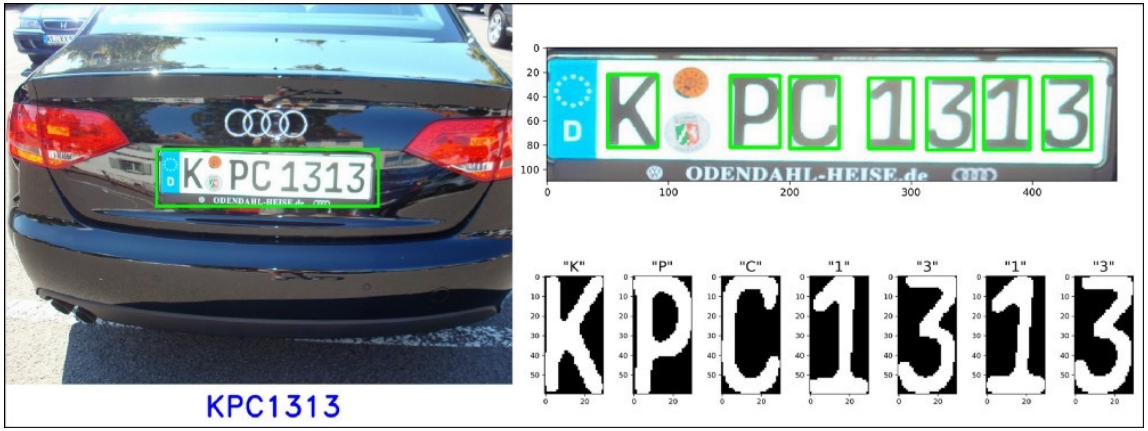






Number plate recognition

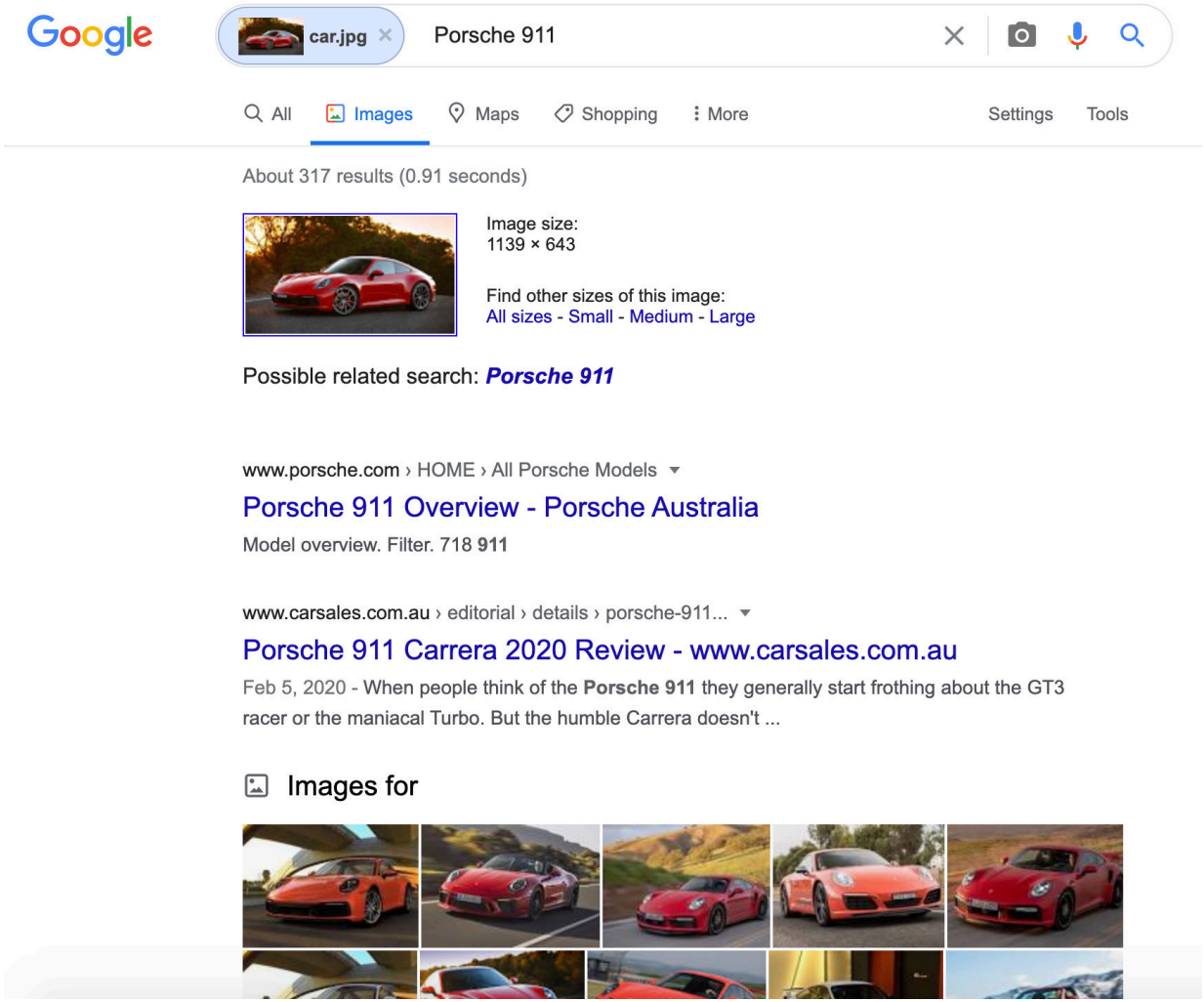
- Automatic Number Plate Recognition (ANPR) systems used since 2000s for security, police check points, toll roads and parking stations
- Based on a workflow of detecting plate, segmenting characters and recognising digits



"License Plate Detection and Recognition in Unconstrained Scenarios", S.M. Silva, C.R. Jung, ECCV, 2018.

https://security.coletek.org/
https://www.multiscan.com.au/

Reverse Image Search



- Google reverse image search:
  - Service established in 2001
  - Compares image to database of other images to look for similarity, assigns tags based on database
  - Latest version using deep convolution neural networks trained using database consisting of hundreds of millions of images
- Other similar search services such as TinEye, Pixsy etc.

Supervised Machine Learning

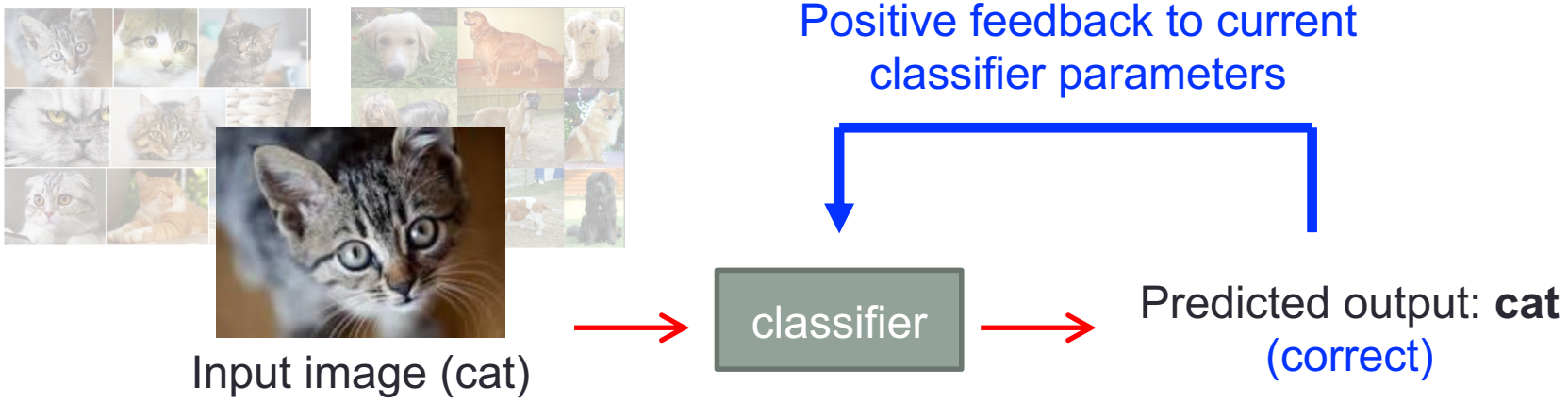
- **Machine Learning**: a branch of computer science focused on developing algorithms that improve with experience
- **Supervised Machine Learning**: Algorithms that perform a function/provide an output based on some input (e.g. image classification) where the "experience" is provided in the form input/output pairs that represent the correct way the task should be performed

Supervised Machine Learning and image classification

- **Machine Learning**: a branch of computer science focused on developing algorithms that improve with experience
- **Supervised Machine Learning**: Algorithms that perform a function/provide an output based on some input (e.g. image classification) where the "experience" is provided in the form input/output pairs that represent the correct way the task should be performed
- In the context of image classification, the provided input/output pairs (referred to as training examples) are sets of example images and their corresponding output classes
- Supervised machine learning uses a process of optimization to "tune" the parameters of an algorithm that best performs the task on new input images

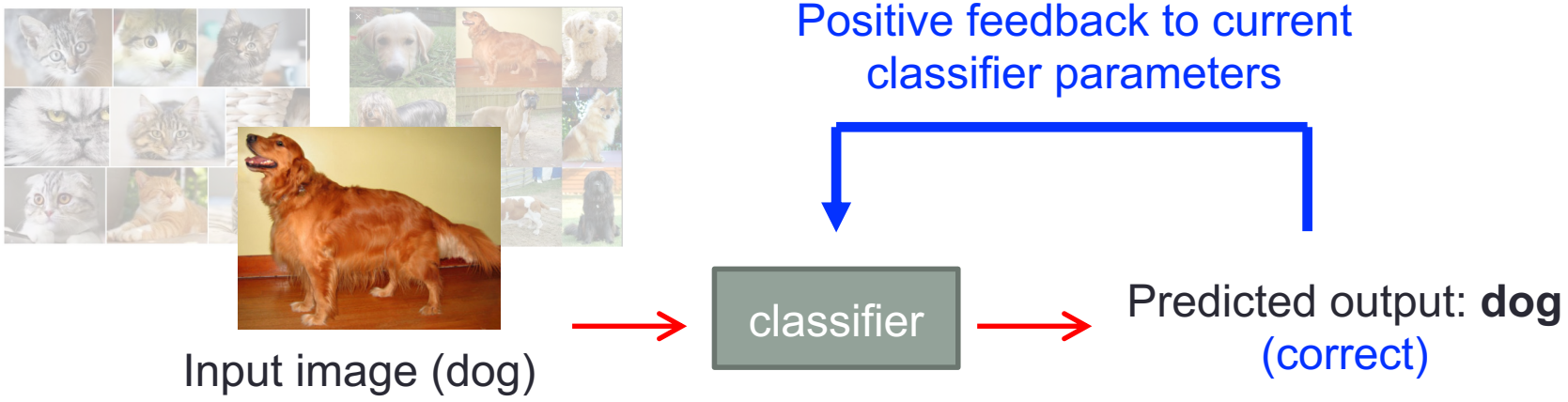
Supervised Machine Learning and image classification

- **Machine Learning**: a branch of computer science focused on developing algorithms that improve with experience
- **Supervised Machine Learning**: Algorithms that perform a function/provide an output based on some input (e.g. image classification) where the "experience" is provided in the form input/output pairs that represent the correct way the task should be performed
- In the context of image classification, the provided input/output pairs (referred to as training examples) are sets of example images and their corresponding output classes
- Supervised machine learning uses a process of optimization to "tune" the parameters of an algorithm that best performs the task on new input images



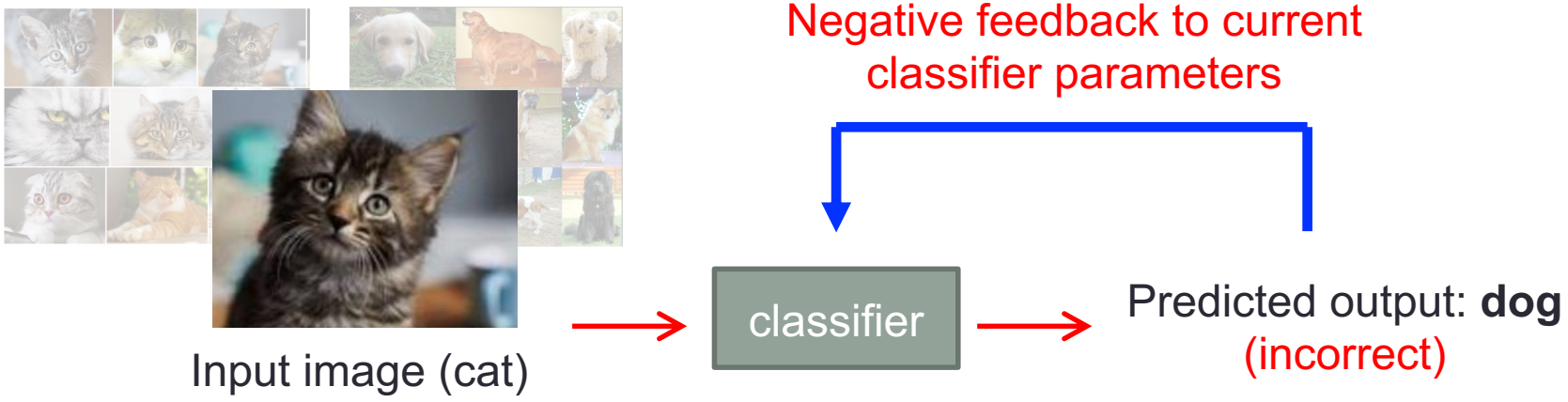
Supervised Machine Learning and image classification

- **Machine Learning**: a branch of computer science focused on developing algorithms that improve with experience
- **Supervised Machine Learning**: Algorithms that perform a function/provide an output based on some input (e.g. image classification) where the "experience" is provided in the form input/output pairs that represent the correct way the task should be performed
- In the context of image classification, the provided input/output pairs (referred to as training examples) are sets of example images and their corresponding output classes
- Supervised machine learning uses a process of optimization to "tune" the parameters of an algorithm that best performs the task on new input images

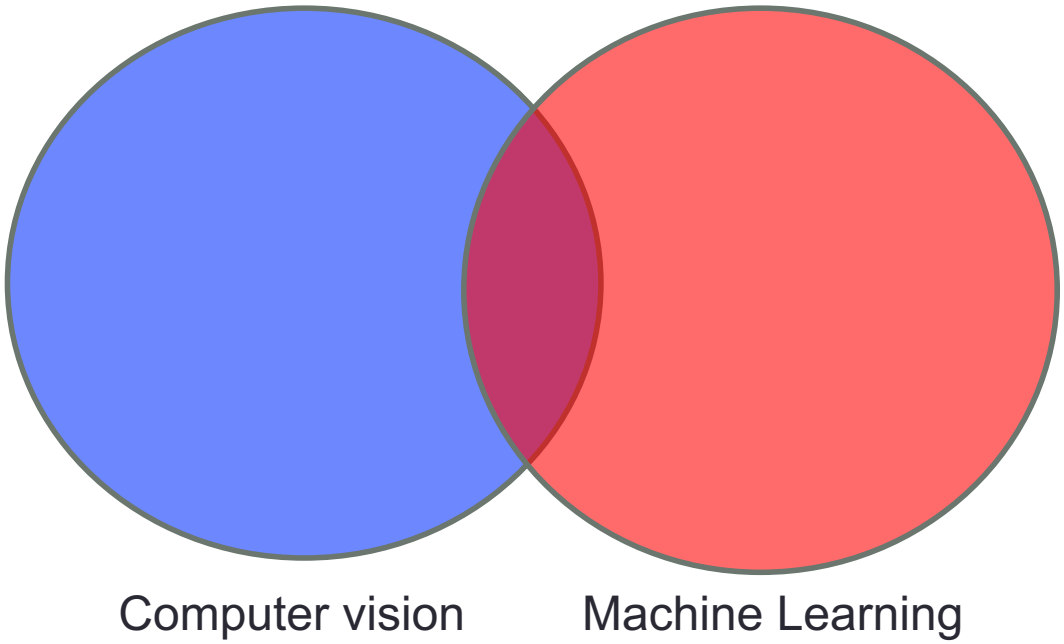


Supervised Machine Learning and image classification

- **Machine Learning**: a branch of computer science focused on developing algorithms that improve with experience
- **Supervised Machine Learning**: Algorithms that perform a function/provide an output based on some input (e.g. image classification) where the "experience" is provided in the form input/output pairs that represent the correct way the task should be performed
- In the context of image classification, the provided input/output pairs (referred to as training examples) are sets of example images and their corresponding output classes
- Supervised machine learning uses a process of optimization to "tune" the parameters of an algorithm that best performs the task on new input images

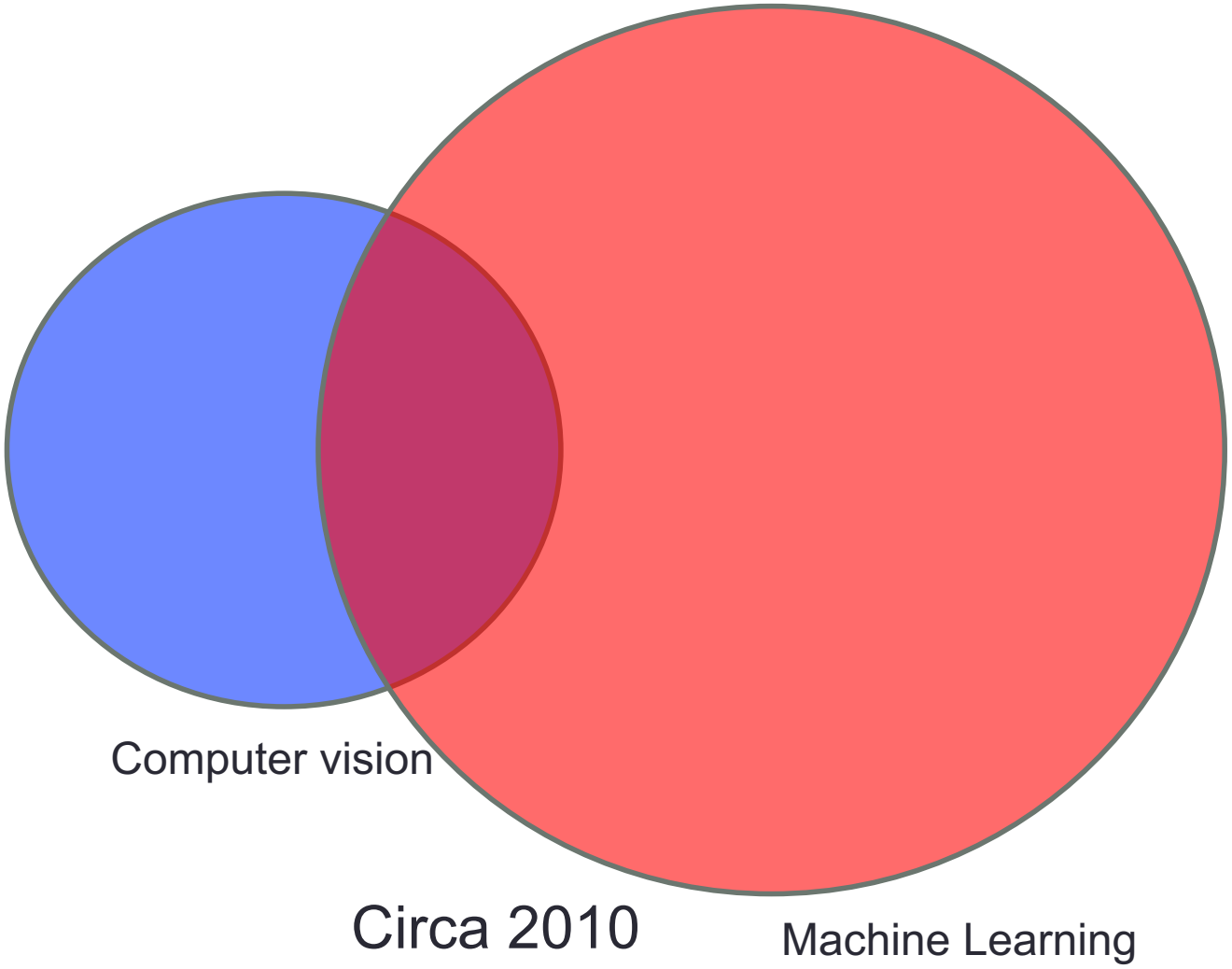


Computer Vision and Machine Learning



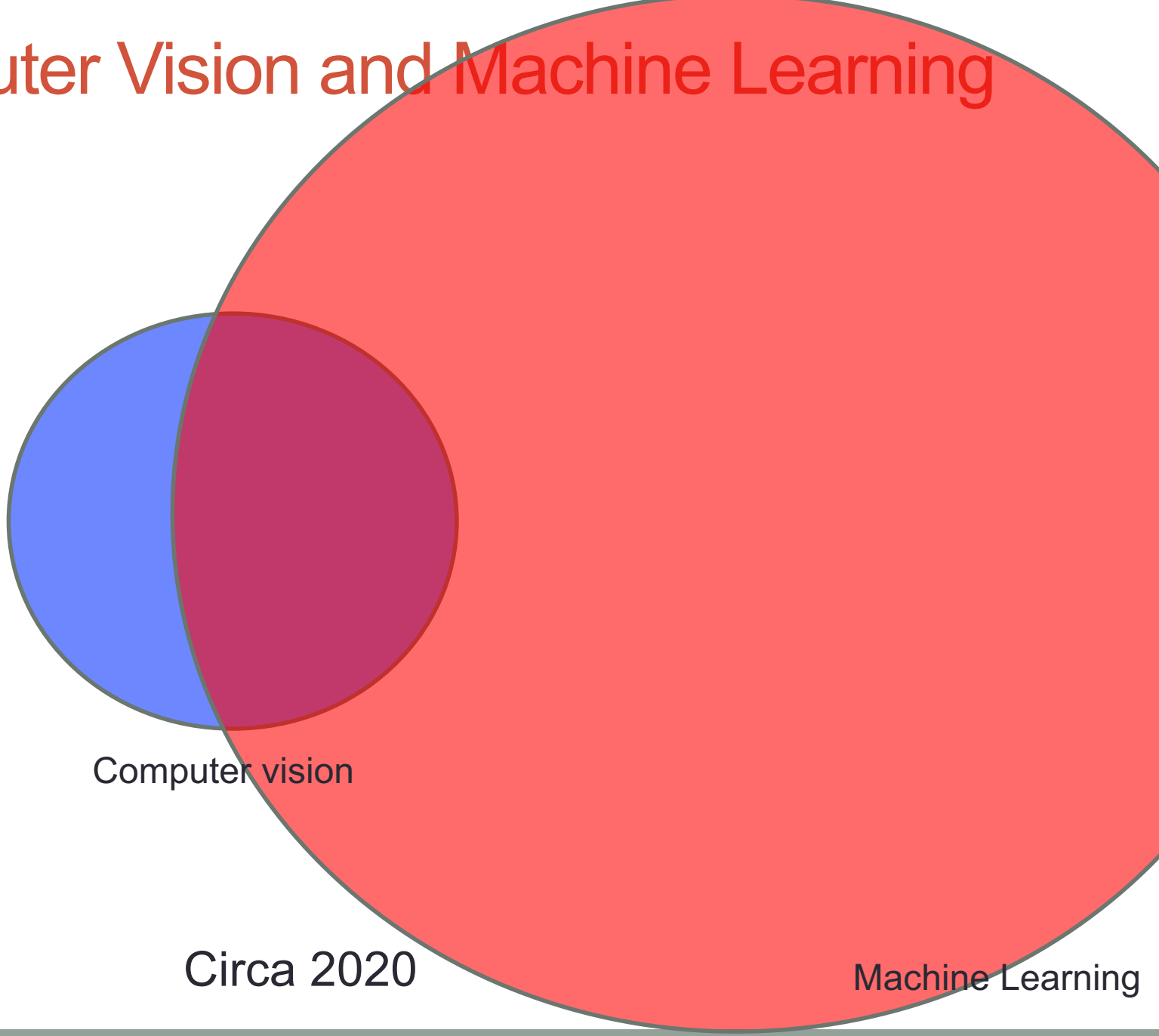
Circa 2000

Computer Vision and Machine Learning



Circa 2010

Computer Vision and Machine Learning



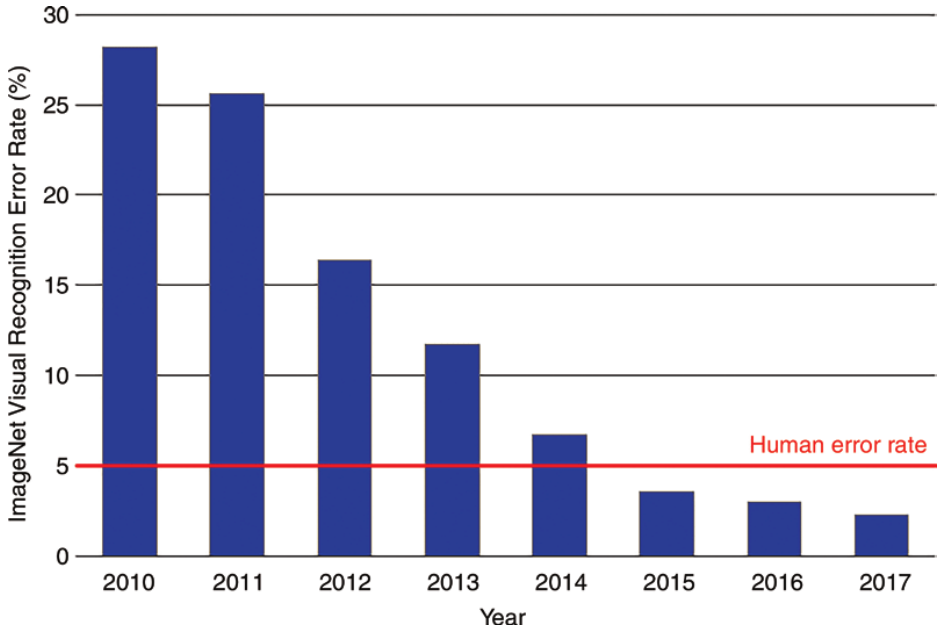
Circa 2020

Computer Vision and Machine Learning

- Advances in machine learning and it's use in computer vision have been driven by:
  - Availability of data: online image repositories containing millions of images provide ample training data for new algorithm development
  - Advances in computation and Graphical Processor Units (GPUs): allows for complex image classification models to be developed and trained efficiently
  - Resurgence of Deep Learning methods and algorithms such as neural networks: enabled by large quantities of data and low-cost, general purpose GPUs

Computer Vision and Machine Learning

- Competitions such as the Imagenet Large-scale Visual Recognition Challenge (ILSVRC) used to encourage research/development
- Imagenet: large dataset containing 14 million images across 200,000 categories
- Since 2015, state-of-the-art algorithms outperform humans



Computer Vision and Machine Learning

- Supervised Machine Learning can also be used for non-classification applications such as Image Colourisation and Image Super-resolution

Image colourisation (Zhang, Isola, Efros. "Colorful Image Colorization", ECCV, 2016.)

Image Super-resolution ("Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", Ledig et. al., 2017.)

Computer Vision and Machine Learning

Overview of a process for developing an image classification algorithm

1. Training Data Collection and Image Pre-Processing
2. Image Feature Extraction
3. Classification Algorithms and Algorithm Selection
4. Model Training and Testing

Image Training Data Collection

- An image classifier will only perform as well as the data that is provided to it
- Training datasets should represent the real variation in image data for a given class including variations owing to illumination, pose, scale, form etc.

Variation in Scale

Variation in Form

Variation in Pose

Data Augmentation

- Data augmentation is a process of artificially providing additional training data by “augmenting” existing training examples through random variations in scale, rotation, skew etc.

Additional training examples using random variations in scale, warp, lighting and rotation

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)

Avg. [R,G,B] = [200, 205, 100]

OR

Avg. [R,G,B] =

Top: [100, 230, 100]
Middle: [200, 205, 100]
Bottom: [240, 110, 100]

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)

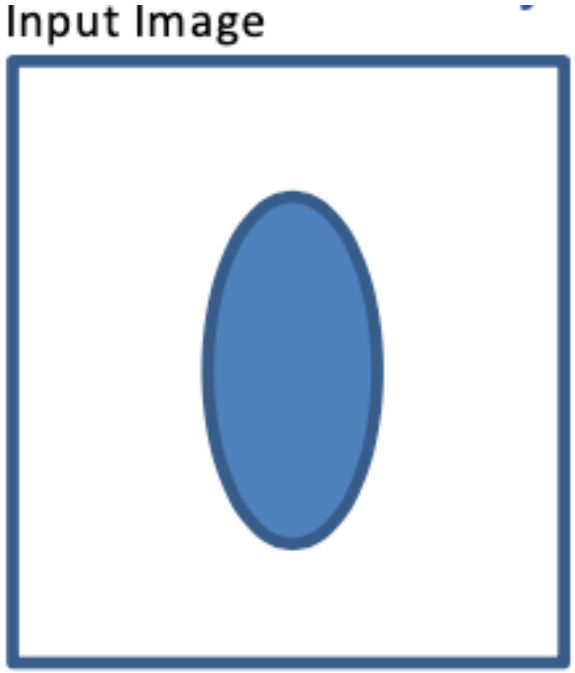
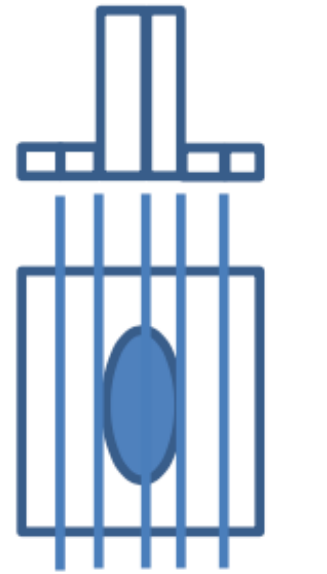


Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)



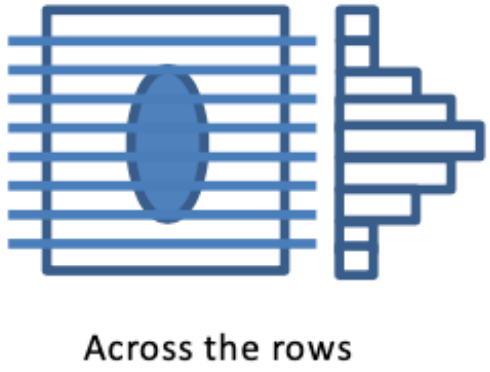
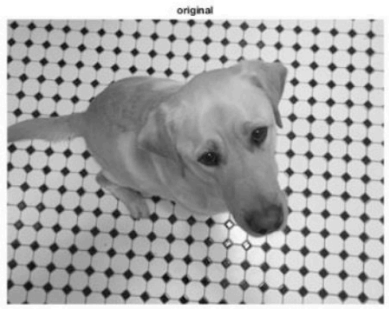
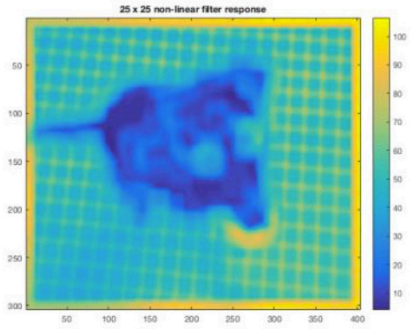


Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)





Avg. [R,G,B] of filter A response = [80,100,220]

Avg. [R,G,B] of filter B response = [20,104,230]

…

Image Classification and Features

- Input image data is very high-dimensional: rather than use the image data itself as an input to the classifier, it is common to reduce this dimensionality by using a **feature vector**

- A feature vector contains a number of properties of the image that can be reproducibly extracted from the raw image, such as:
  - Image intensity, colour
  - Image histograms: over intensity and spatially
  - Textural properties: response to different filters
  - Even just the raw pixel data itself (perhaps downsampled)

Image Classification and Features

- Extracted features are then mapped to a vector of real-valued numbers **x** that form a point in N-dimensional space for a single image:

Raw image data

Example Feature Space

$$\mathbf{x} = [x_1, x_2, x_3, \ldots, x_N]^T$$

x_1: Average red intensity
x_2: Average response to texture filter A
x_3: First x-direction histogram value
$\vdots$

Feature extraction

$$\mathbf{x} = [0.012, 0.3, 20.45, \ldots, 1.17]^T$$

Image Classification and Features

- Extracted features are then mapped to a vector of real-valued numbers **x** that form a point in N-dimensional space for a single image:

Raw image data

Example Feature Space

$$\mathbf{x} = [x_1, x_2, x_3, \ldots, x_N]^T$$

x_1: Average red intensity
x_2: Average response to texture filter A
x_3: First x-direction histogram value
$\vdots$

Feature extraction

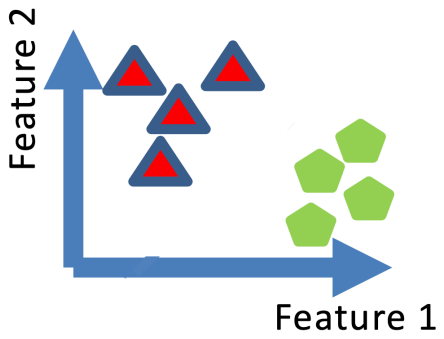
$$\mathbf{x} = [0.012, 0.3, 20.45, \ldots, 1.17]^T$$

- One trivial type of feature extraction is to use the raw pixel data: this essentially maps the 2D raster data into a 1D column vector **x**, where values are the intensity

Classification Algorithms

- Formally, a classifier is a non-linear function approximator **f()** that maps an input feature point **x** into a discrete class value:

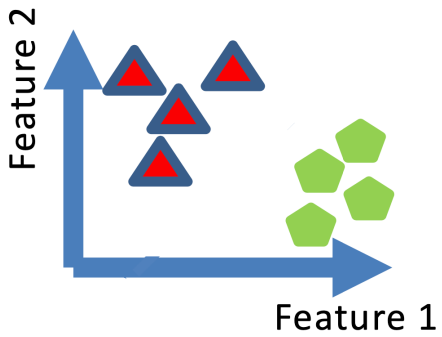
$$y = f(\mathbf{x}) \qquad y \in \{0, 1, 2, ...\}$$



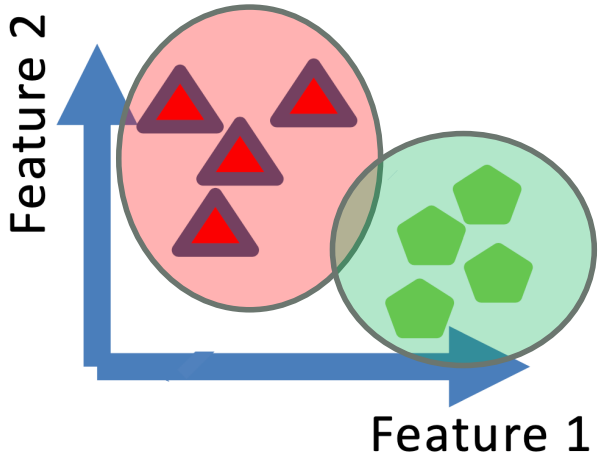
Classification Algorithms

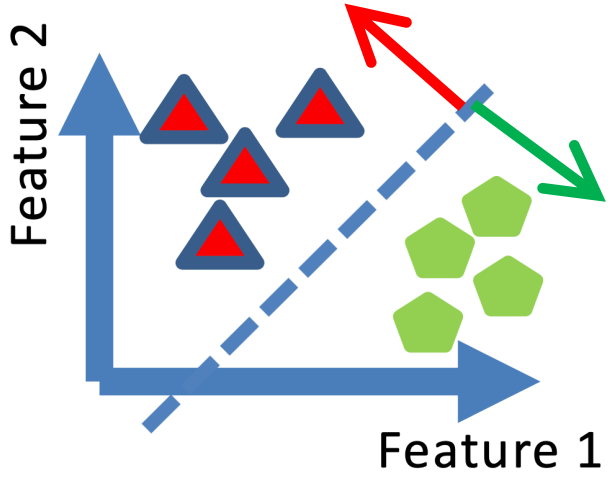
- Formally, a classifier is a non-linear function approximator **f()** that maps an input feature point **x** into a discrete class value:

$$y = f(\mathbf{x}) \qquad y \in \{0, 1, 2, ...\}$$



- Classification algorithms use various scheme to perform this, but most algorithms work on one of two principles:

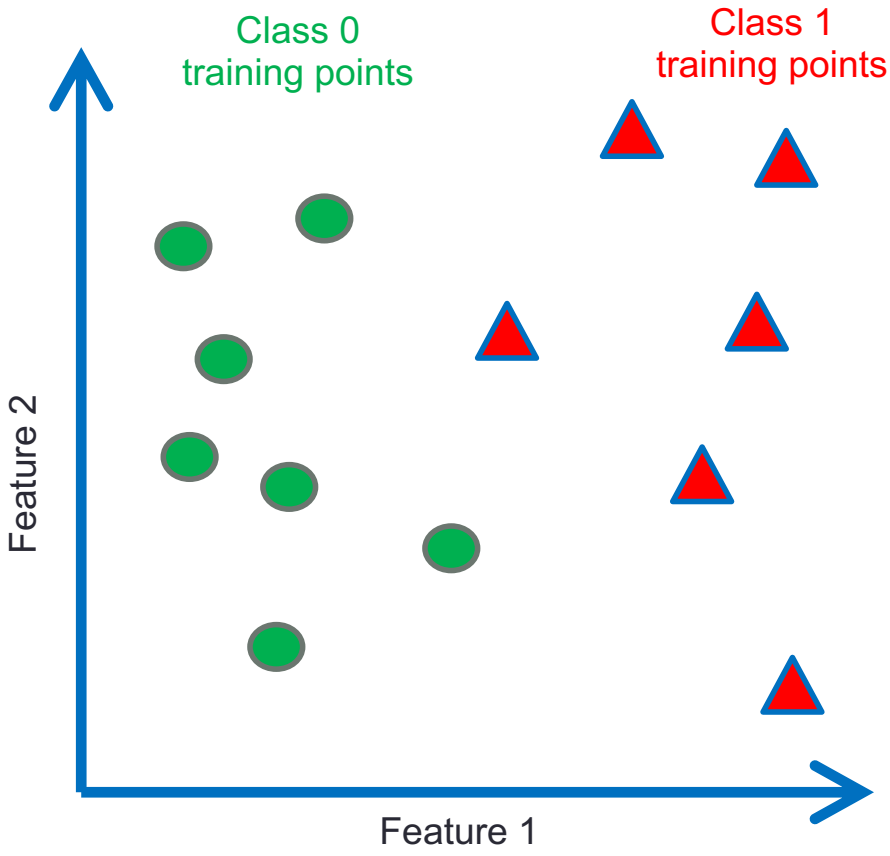




- **Generative Classifiers:** create a distribution in the feature space that represents each class
- **Discriminative Classifiers:** create a boundary between difference classes

k-NN (k-Nearest Neighbours)

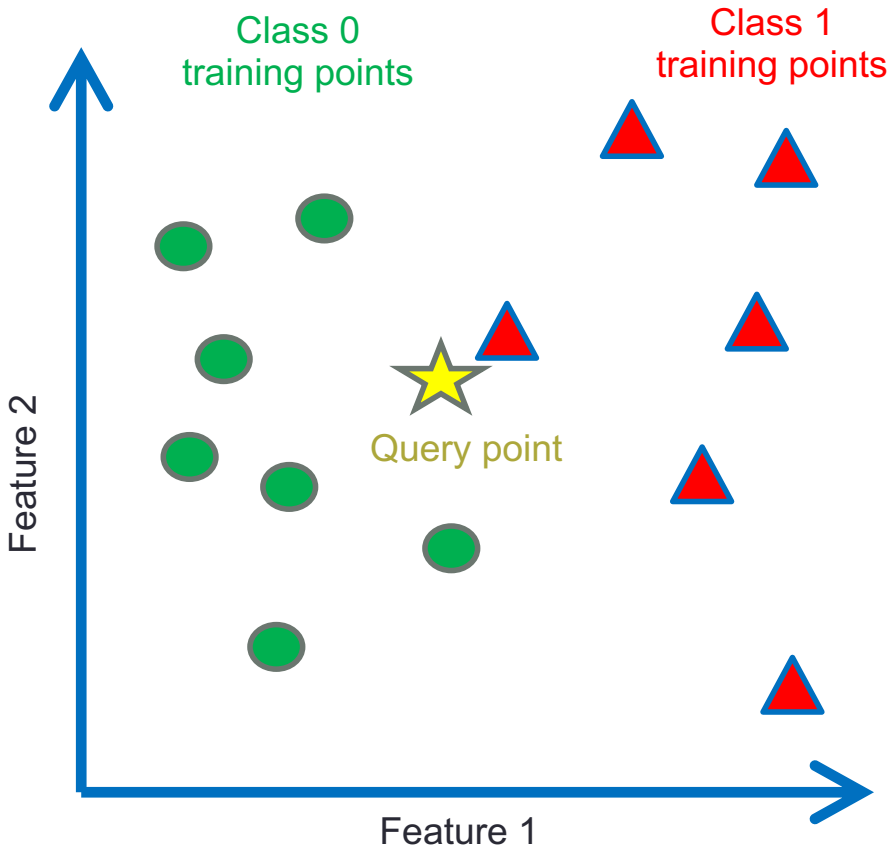
- k-NN is a very simple approach to classification in which the output class corresponds to the classes of the k closest points in the feature space to a given input/query point:



- The value of k is a tuning parameter that ends up controlling for noise and outliers
- K-NN can be varied by setting the class based on the weighted distances to the k-closest points (where weights are inversely proportionate to distance)
- Algorithm typically constructs a KD-tree of the training points and queries points in O(log(N)) time to reduce computation

k-NN (k-Nearest Neighbours)

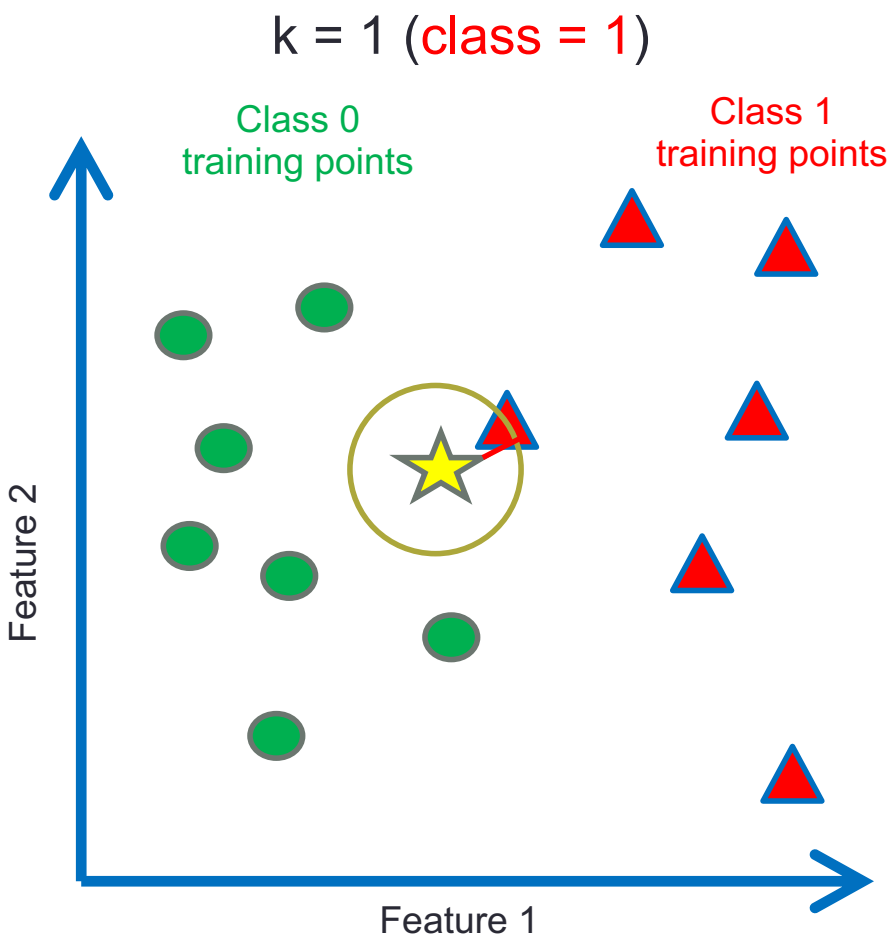
- k-NN is a very simple approach to classification in which the output class corresponds to the classes of the k closest points in the feature space to a given input/query point:



- The value of k is a tuning parameter that ends up controlling for noise and outliers
- K-NN can be varied by setting the class based on the weighted distances to the k-closest points (where weights are inversely proportionate to distance)
- Algorithm typically constructs a KD-tree of the training points and queries points in O(log(N)) time to reduce computation

k-NN (k-Nearest Neighbours)

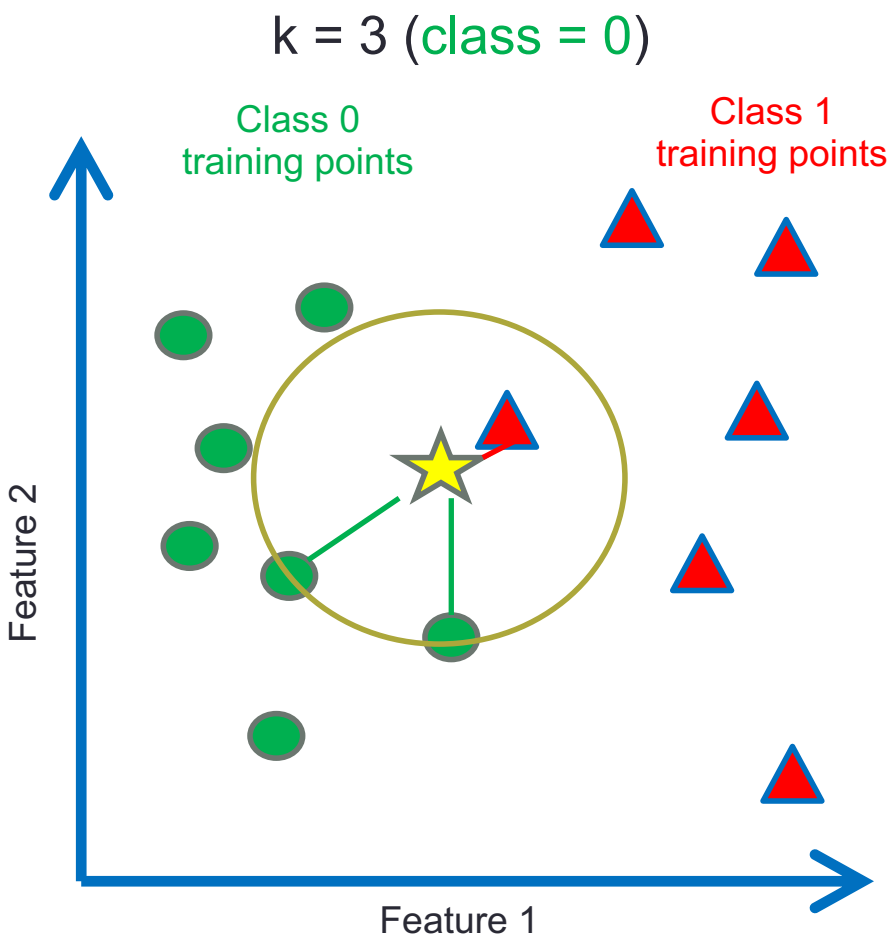
- k-NN is a very simple approach to classification in which the output class corresponds to the classes of the k closest points in the feature space to a given input/query point:



- The value of k is a tuning parameter that ends up controlling for noise and outliers
- K-NN can be varied by setting the class based on the weighted distances to the k-closest points (where weights are inversely proportionate to distance)
- Algorithm typically constructs a KD-tree of the training points and queries points in O(log(N)) time to reduce computation

k-NN (k-Nearest Neighbours)

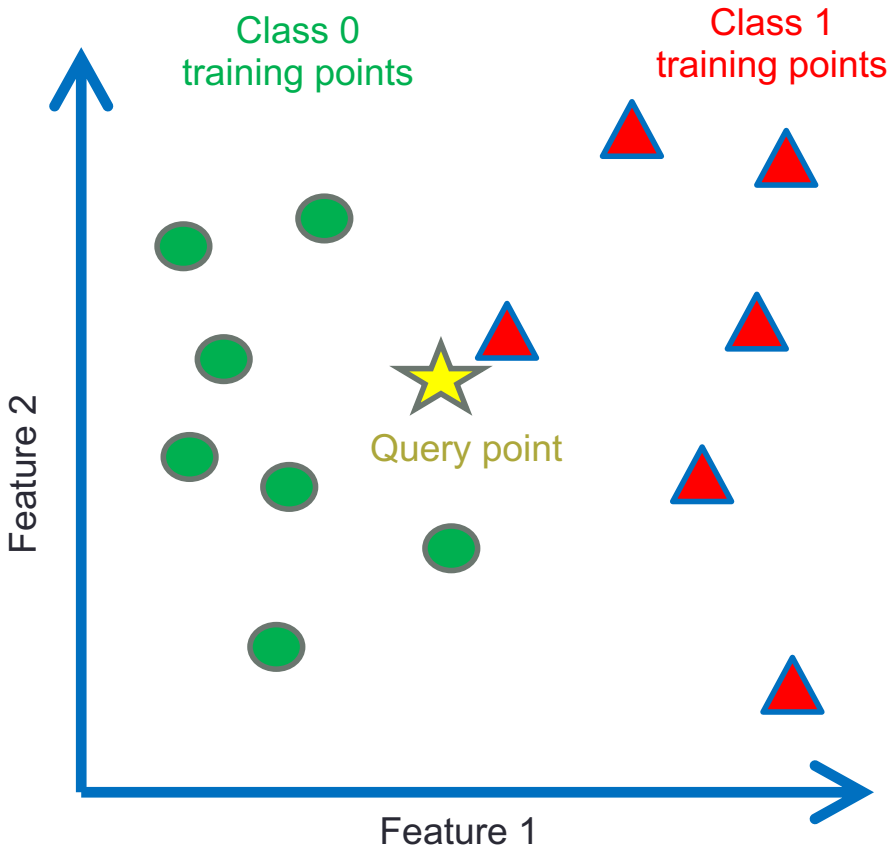
- k-NN is a very simple approach to classification in which the output class corresponds to the classes of the k closest points in the feature space to a given input/query point:



- The value of k is a tuning parameter that ends up controlling for noise and outliers
- K-NN can be varied by setting the class based on the weighted distances to the k-closest points (where weights are inversely proportionate to distance)
- Algorithm typically constructs a KD-tree of the training points and queries points in O(log(N)) time to reduce computation

k-NN (k-Nearest Neighbours)

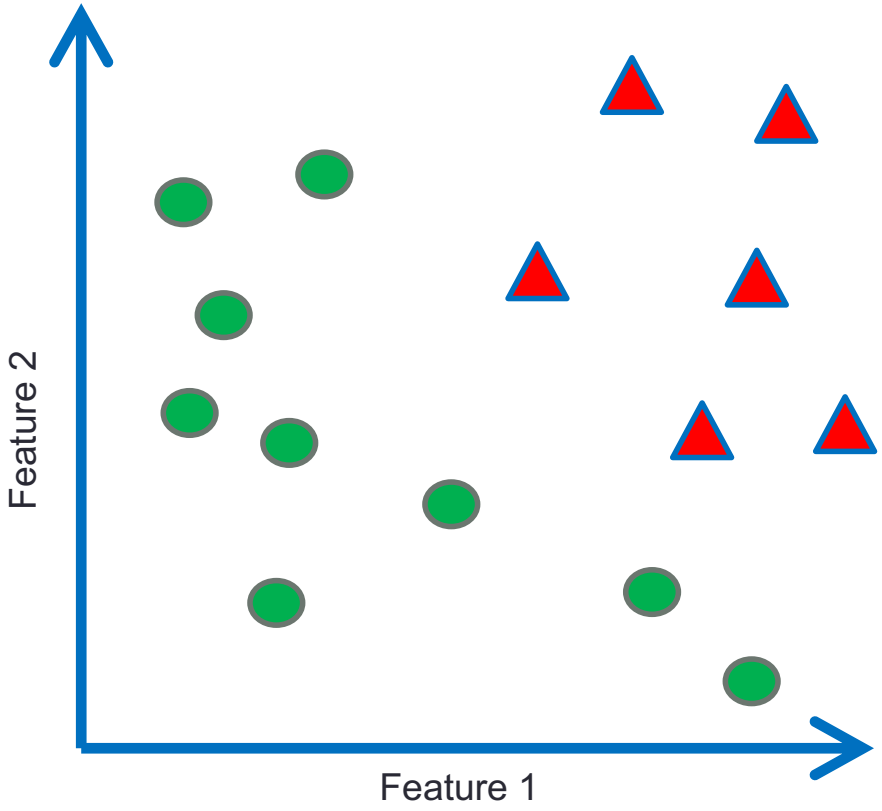
- k-NN is a very simple approach to classification in which the output class corresponds to the classes of the k closest points in the feature space to a given input/query point:



- The value of k is a tuning parameter that ends up controlling for noise and outliers
- K-NN can be varied by setting the class based on the weighted distances to the k-closest points (where weights are inversely proportionate to distance)
- Algorithm typically constructs a KD-tree of the training points and queries points in O(log(N)) time to reduce computation

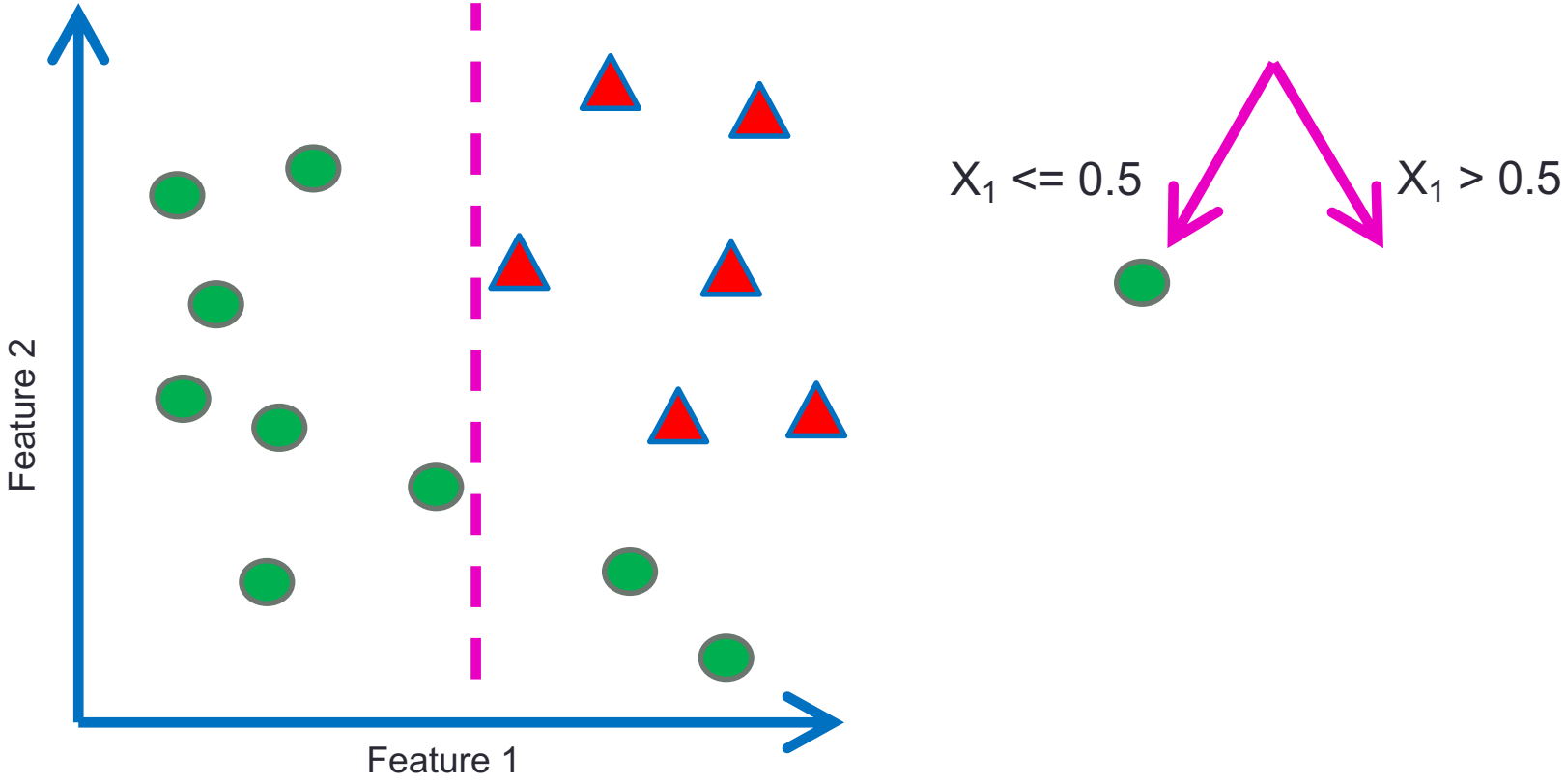
Decision Trees

- Decision Trees are an algorithm for classification that create a series of decision rules based on thresholds over individual features
- The algorithm find the feature and threshold which best splits the data and then recursively applies this across remaining features



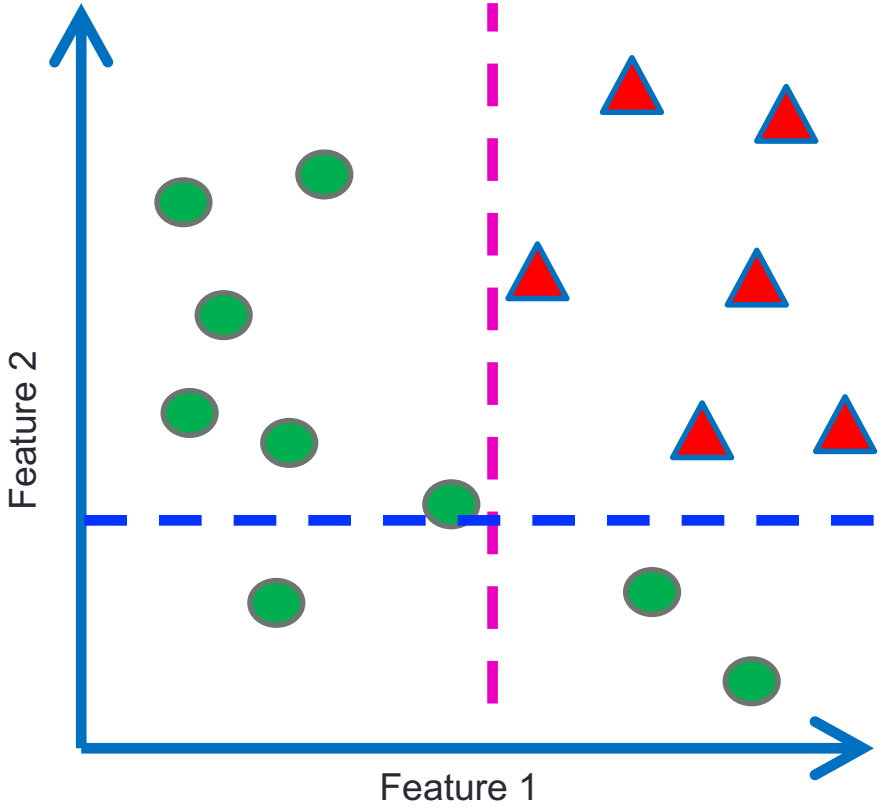
Decision Trees

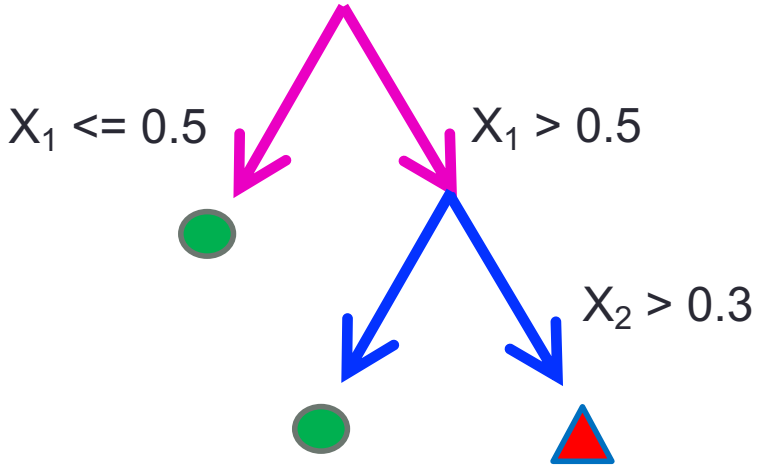
- Decision Trees are an algorithm for classification that create a series of decision rules based on thresholds over individual features
- The algorithm find the feature and threshold which best splits the data and then recursively applies this across remaining features



Decision Trees

- Decision Trees are an algorithm for classification that create a series of decision rules based on thresholds over individual features
- The algorithm find the feature and threshold which best splits the data and then recursively applies this across remaining features

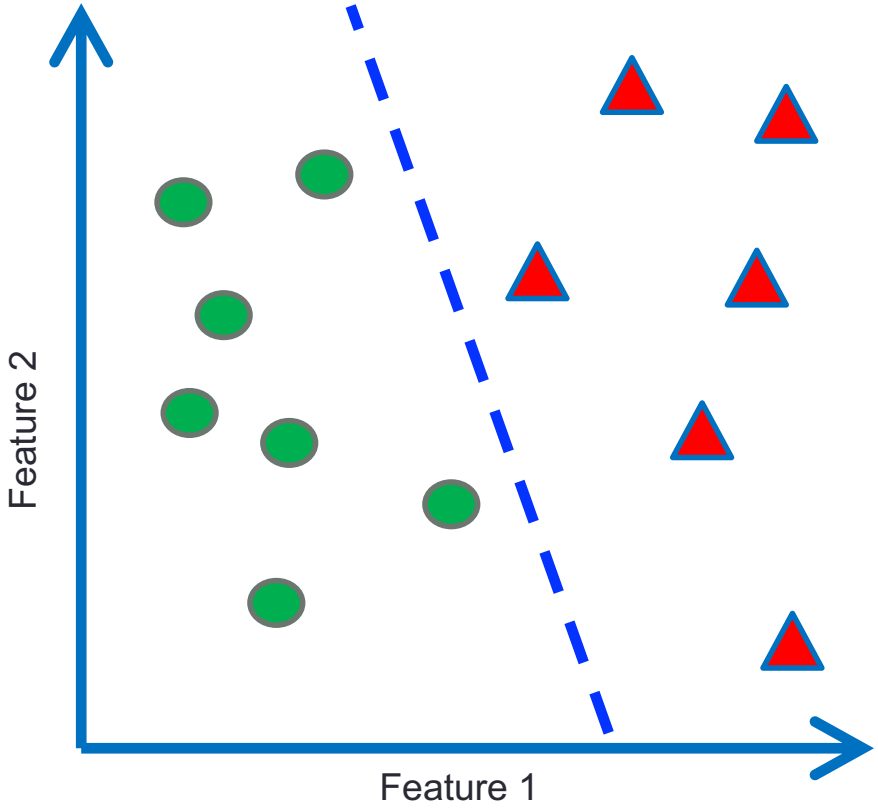




- Final tree can be pruned for “weak” branches

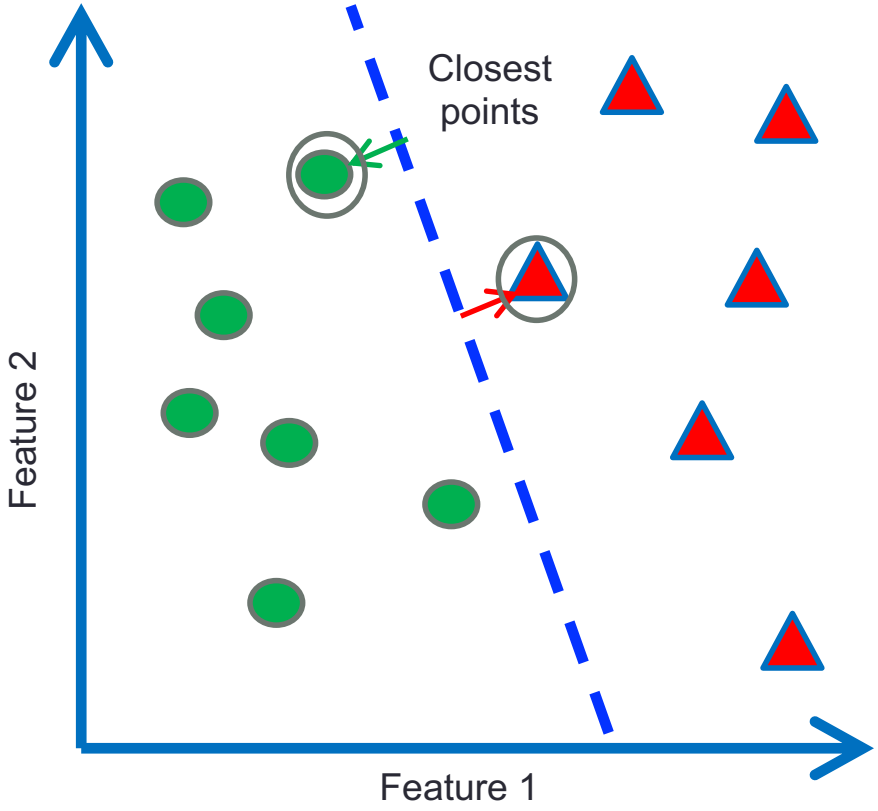
Support Vector Machines (SVMs)

- SVMs create hyperplane between two classes that has the maximum margin between the closest data point on each side of the boundary



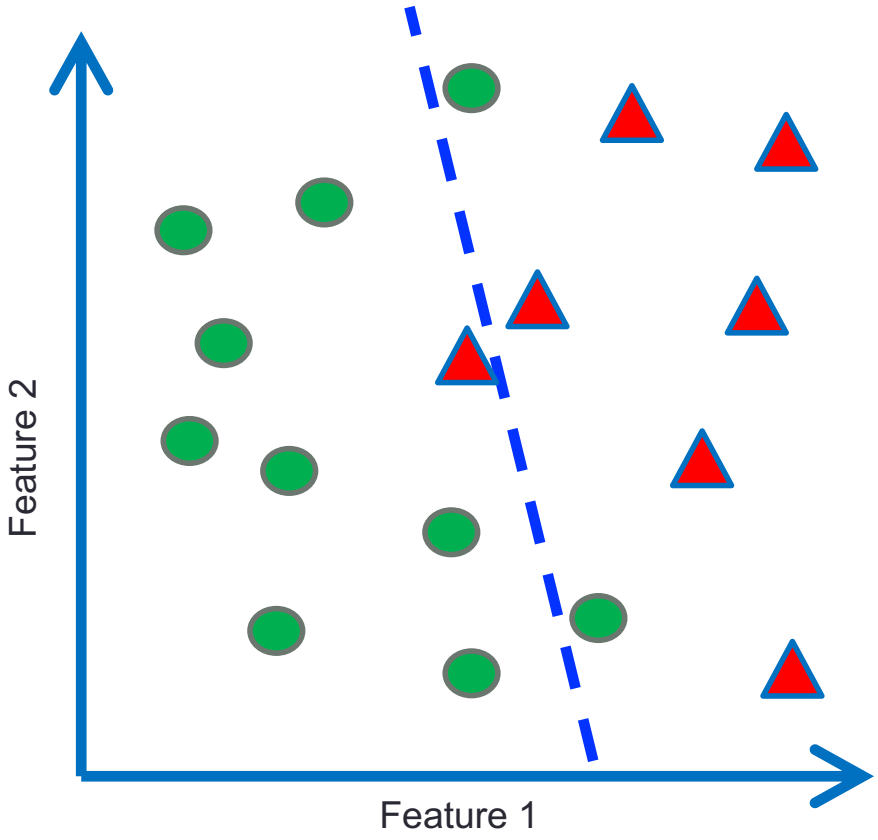
Support Vector Machines (SVMs)

- SVMs create hyperplane between two classes that has the maximum margin between the closest data point on each side of the boundary
- The hyper-plane is defined by the parameters (**w**,b) such that points satisfying **w**.**x** – b = 0 lie on the hyperplane



Support Vector Machines (SVMs)

- SVMs create hyperplane between two classes that has the maximum margin between the closest data point on each side of the boundary
- The hyper-plane is defined by the parameters (**w**,b) such that points satisfying **w**.**x** – b = 0 lie on the hyperplane



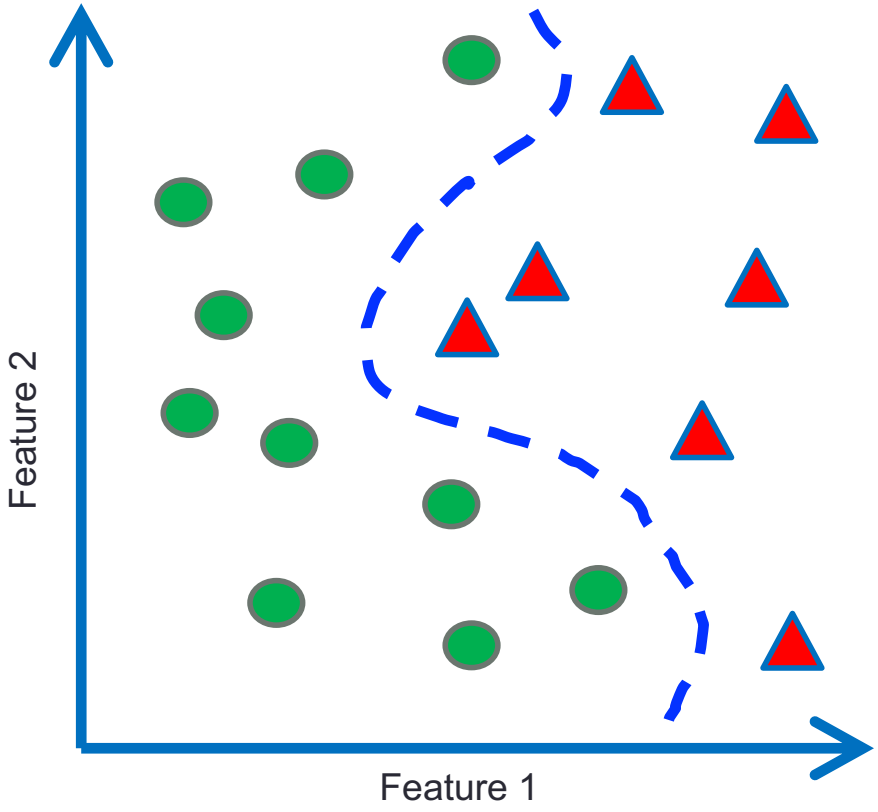
- To handle the situation that the data is not linearly separable, quadratic programming optimization is used to compute (**w**,b) that minimize the objective function:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2$$

- For training points i=1:n, where $y_i()$ is a function that maps to > 1 when the training point lies on the correct side of the hyper plane, and lambda is a tuning parameter that controls how strictly the plane separates the two classes

Support Vector Machines (SVMs)

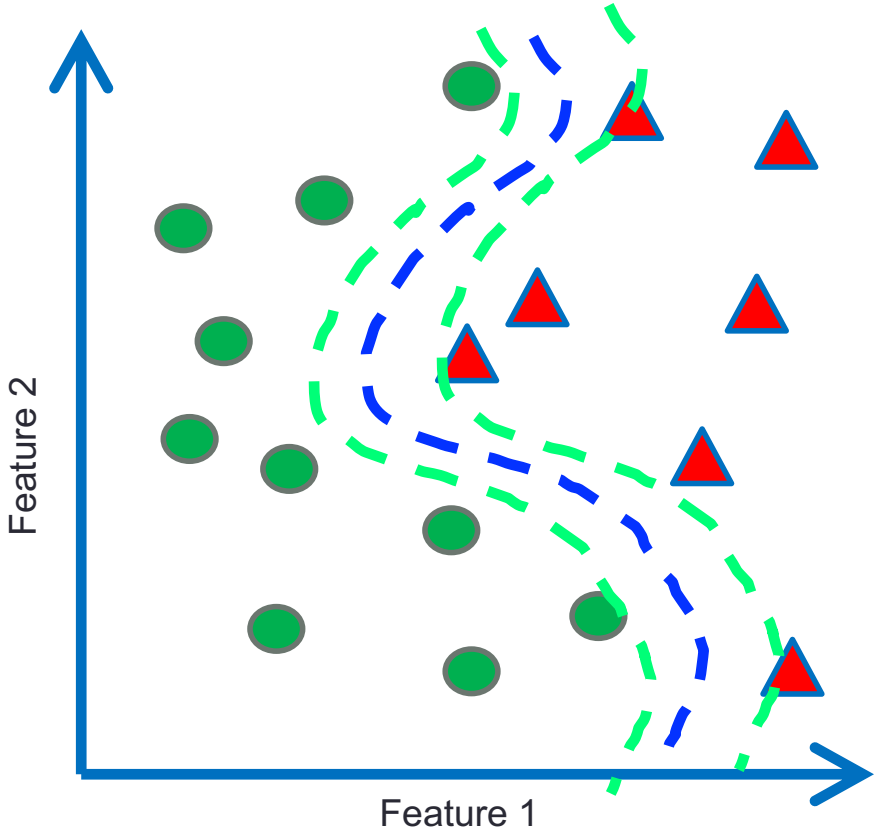
- SVMs create hyperplane between two classes that has the maximum margin between the closest data point on each side of the boundary
- The hyper-plane is defined by the parameters (**w**,b) such that points satisfying **w**.**x** – b = 0 lie on the hyperplane



- Another way to deal with non-linearly separable data is by replacing the dot product in the hyperplane equation with a non-linear kernel, to create a **non-linear SVM**
- This effectively maps the original feature space into a higher-dimensional one in which a linear hyperplane is used to split the data

Extending algorithms to multi-class

- Most classification algorithms provide a measure of “confidence” or “probability of correctness” for a given classification output based on how the query point appears in the feature space
- For example, SVMs can use the distance from the hyperplane, decision trees distance from thresholds



- To extend algorithms such as decision trees and SVM from binary to multi-class classifiers, a one vs. all classifier is typically built for each class: final class selected based on furthest hyperplane/threshold distance

5 minute break

Metrics of Classification Performance

- Classification performance metrics are an important part of understanding how the classifier might perform in a target application
- Consider a situation in which we have built a classifier and we now wish to evaluate its performance using a set of input examples for which we know the true class
- For a binary classifier (0/1) if we pass in an example input point and record the predicted class we can have one of a number of situations:
  - **True Positive (TP):** the true class was 1 and the predicted class was 1
  - **True Negative (TN):** the true class was 0 and the predicted class was 0
  - **False Positive (FP):** the true class was 0 and the predicted class was 1
  - **False Negative (FN):** the true class was 1 and the predicted class was 0
- The balance of these occurrences over the testing data can be used to measure performance

Metrics of Classification Performance

- Accuracy:

$$\frac{TP + TN}{\text{Total number of points}}$$

- Accuracy works well for balanced classes, but poorly for unbalanced ones
- Consider a test set which has 90% positive examples and 10% negative examples: a classifier which always specifies an input is positive (regardless of the actual feature values) would have an accuracy of 90%

Metrics of Classification Performance

- **Precision:** measures how correct the model is when it predicts a positive class

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

- **Recall:** measures how many of the positive classes does the model actually predict

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

- Precision and recall can be useful metrics if we wish to tune an algorithm to put more importance on false positives or false negatives: there is often a trade-off between the two

- **F1 Score** is a measure of the geometric mean of precision and recall and is maximised by having good values of both

$$\frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Precision	0	25	50	60
Recall	100	75	50	40
F1	0	37.5	50	48

Metrics of Classification Performance

- A confusion matrix is a table which shows true/false positive/negative occurrences:

		Ground Truth	
		0	1
Prediction	0	TN	FN
	1	FP	TP

Metrics of Classification Performance

- Measures can be extended to multi-class problems too
- Consider a 4-class problem:

		Ground Truth			
		1	2	3	4
Prediction	1	TN	FN	TN	TN
	2	FP	TP	FP	FP
	3	TN	FN	TN	TN
	4	TN	FN	TN	TN

- Accuracy, precision and recall are calculated as for binary problems
- Overall accuracy can also be determined by dividing the sum of the diagonals of the confusion matrix by the total number of examples

Metrics of Classification Performance

- For multi-class classification algorithms that can provide an ordered list of possible classes (based on likelihood) for each example, we can measure their top-K accuracy by the number of times the correct class is within the top-K predictions:
- Often values of top-1 (normal accuracy) or top-5 are used as competition dataset metrics for problems with very large numbers of classes

Van

1. Van
2. Car
3. Truck
4. Road
5. Building

1. Truck
2. Road
3. Van
4. Car
5. Traffic sign

1. Truck
2. Road
3. Car
4. Building
5. Traffic sign

Classifier Validation

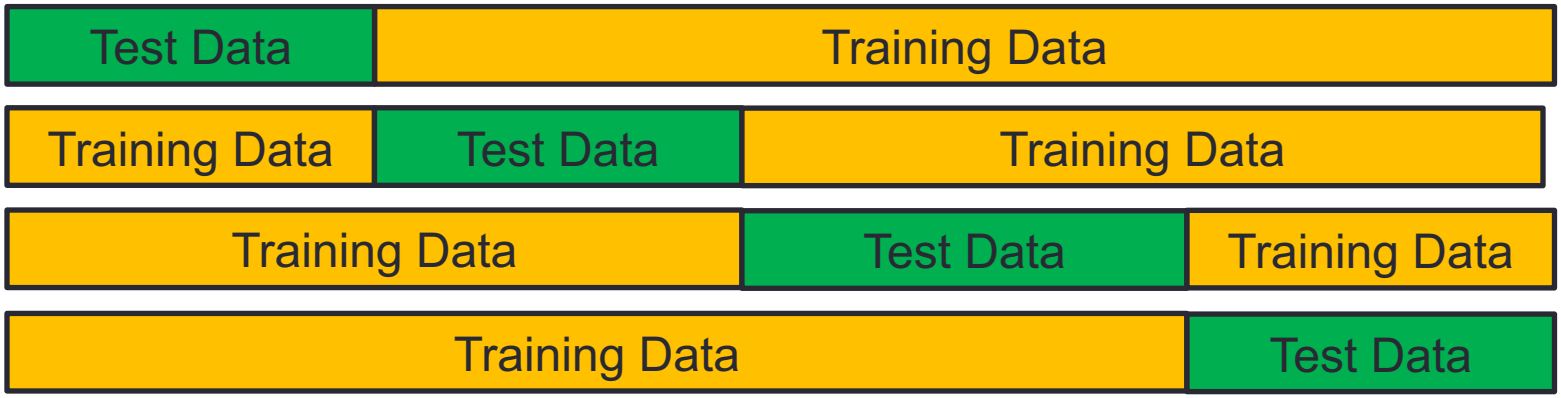
- When evaluating classifier performance, we need to be careful of which examples we used for measuring
- For example, if we use our training data used to develop the classifier to also measure accuracy, our result could be quite biases:
  - How do we know that a new input image we get won't be very different to what we have seen?
  - Imagine a situation in which our classifier just records all training examples

- Proper validation therefore requires us to use a "hold out" or "test" set of examples which were not used during training

- One issue remains: how do we know that we didn't just get lucky with our selection of training points? How did we decide which points were which?

Cross Validation

- Cross validation is a process where we produce multiple splits of the training/testing data
- In k-fold cross validation, we divide the data into k subsets and repeat the hold out method on each subset, training and testing k-different models
- We then average the performance metrics over the k-folds to measure total expected performance



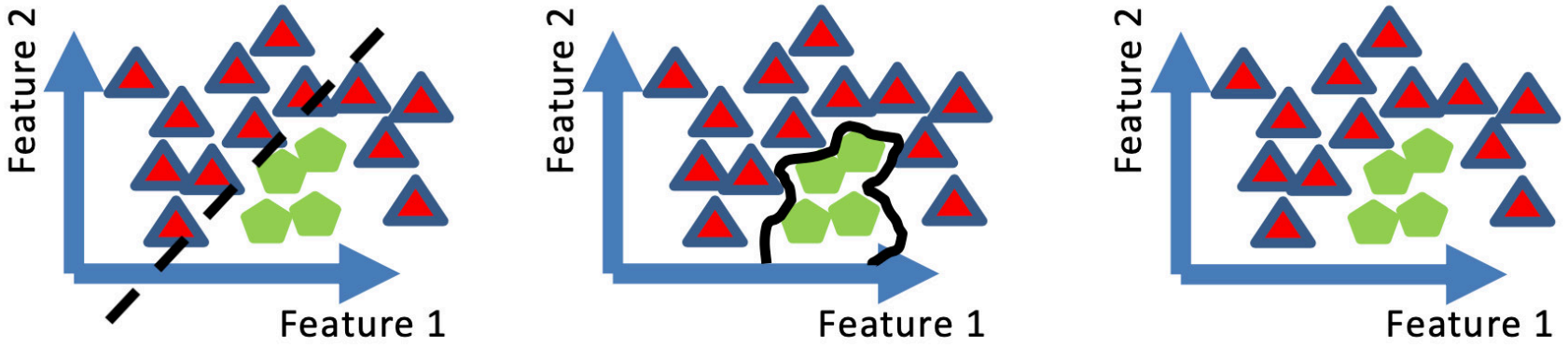
- Where possible, it is also good practice to maintain a final ”validation” dataset which has never been seen during training and testing

Hyperparameter Tuning

- Most classifiers contain tunable parameters (referred to as hyper-parameters) that control how the algorithm functions:
  - k-NN: value of k
  - SVMs: kernel type, lambda, c-value
  - Decision Trees: number of branches to keep
- Hyper-parameter tuning is the process of automatically calibrating these values to a specific problem/dataset
- Common approach is to use block search optimization: several different values for these parameters are set and separate cross-validation procedure is used to measure the performance of each
- Once the optimal parameters are determined by the cross-validation performance, the final test set performance is evaluated on the best model

Model Complexity: Bias vs. Variance trade-off

- As the size of our feature space increases, our model complexity increases: this increase in complexity demands larger training datasets to tune these parameters correctly
- For most classification tasks and the training data available, there exists an optimal model complexity:
  - Too small: model "bias" is high (under-fitting)
  - Too large: model variance is high (over-fitting)



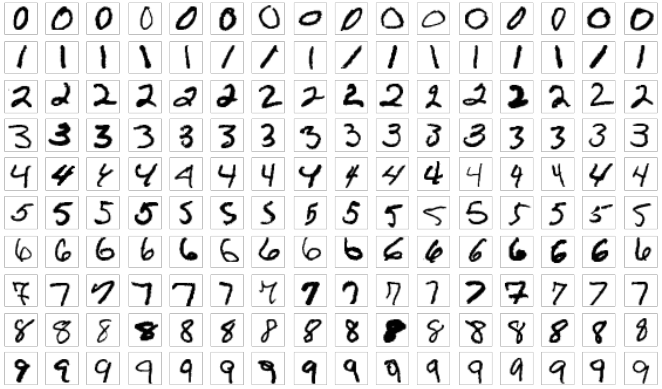
Underfitting (bias problem)
-Model too simple
-high training error
-high test error

Overfitting (variance problem)
-Model too complex
-low training error
-high test error (so useless)

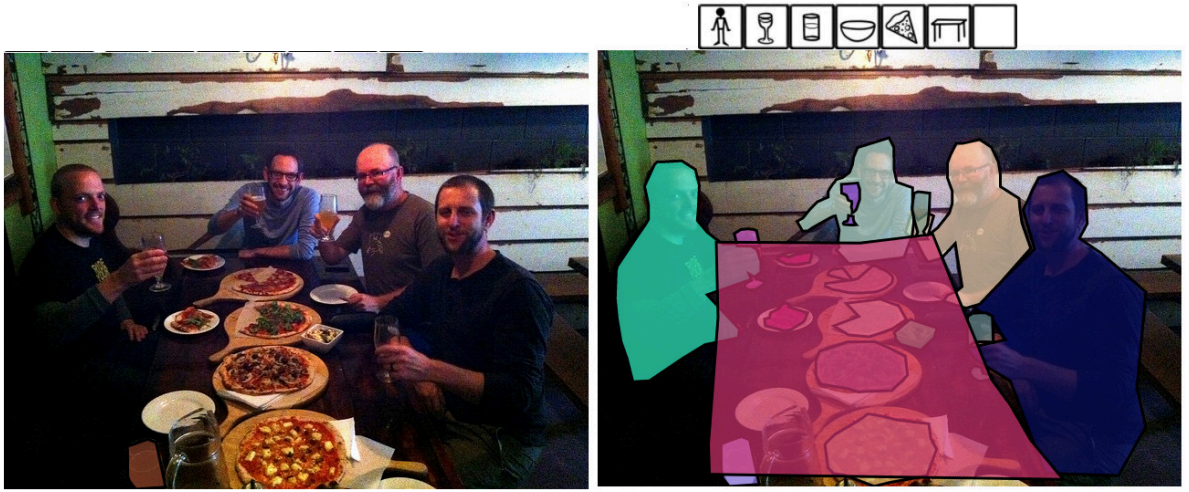
Optimal
-low training error
-low test error

Baseline image datasets used in classification

- Existing image databases can be used for developing models, supplementing training data and validating new algorithms, examples include:
  - MNIST: 60k images of 10 classes (digits)
  - Imagenet: 14 million images over 20,000 classes (http://www.image-net.org/)
  - Common Objects in Context (COCO): 80 classes, 200k images (https://cocodataset.org/#explore)
  - Pascal VOC: (http://host.robots.ox.ac.uk/pascal/VOC/)



MNIST Hand-written Digits



COCO Example: People, wine glass, table, pizza, etc.

Further Reading and Next Week

- References:
  - D. A. Forsyth and J. Ponce, “Computer Vision - A Modern Approach”, Prentice Hall, 2002
  - R. Szeliski, “Computer Vision: Algorithms and Applications”, Springer, 2010

- Next Week:
  - Introduction to Neural Networks, Deep Learning and Image Classification