COMP6733
IoT Design Studio

Machine Learning

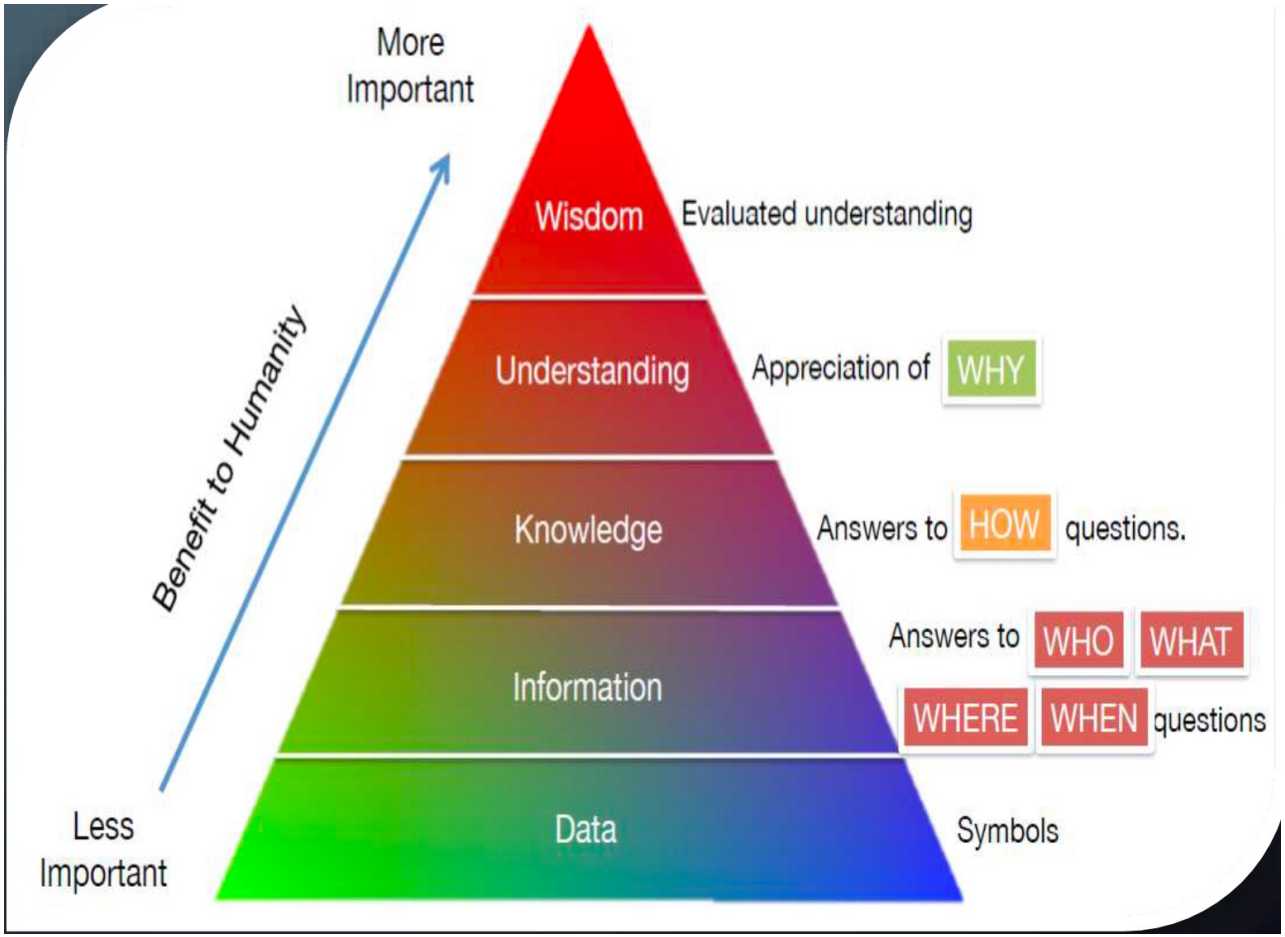
- Machine Learning
- Localisation (WiFi fingerprint-based)

Knowledge Hierarchy

Turning data into wisdom.

And the key point to IoT is that the knowledge hierarchy can be automatically summarized by machines.

Here machine learning comes into play.



"Field of study that gives computers the ability to learn without being explicitly programmed".

(Source: https://en.wikipedia.org/wiki/Machine_learning)

Overview

- Sub-field of Artificial Intelligence
- name is derived from the concept that it deals with "construction and study of systems that can learn from data"
- can be seen as building blocks to make computers learn to behave more intelligently
- It is a theoretical concept. There are various techniques with various implementations
- "A computer program is said to learn from experience (E) with some class of tasks (T) and a performance measure (P) if its performance at tasks in T as measured by P improves with E"

Terminology

- Features
  - The number of features or distinct traits that can be used to describe each item in a quantitative manner
- Samples
  - A sample is an item to process (e.g. classify). It can be a document, a picture, a sound, a video, a row in database or CSV file, or whatever you can describe with a fixed set of quantitative traits
- Feature vector
  - is an n-dimensional vector of numerical features that represent some object
- Feature extraction
  - Preparation of feature vector
  - transforms the data in the high-dimensional space to a space of fewer dimensions
- Training set
  - Set of data to discover potentially predictive relationships

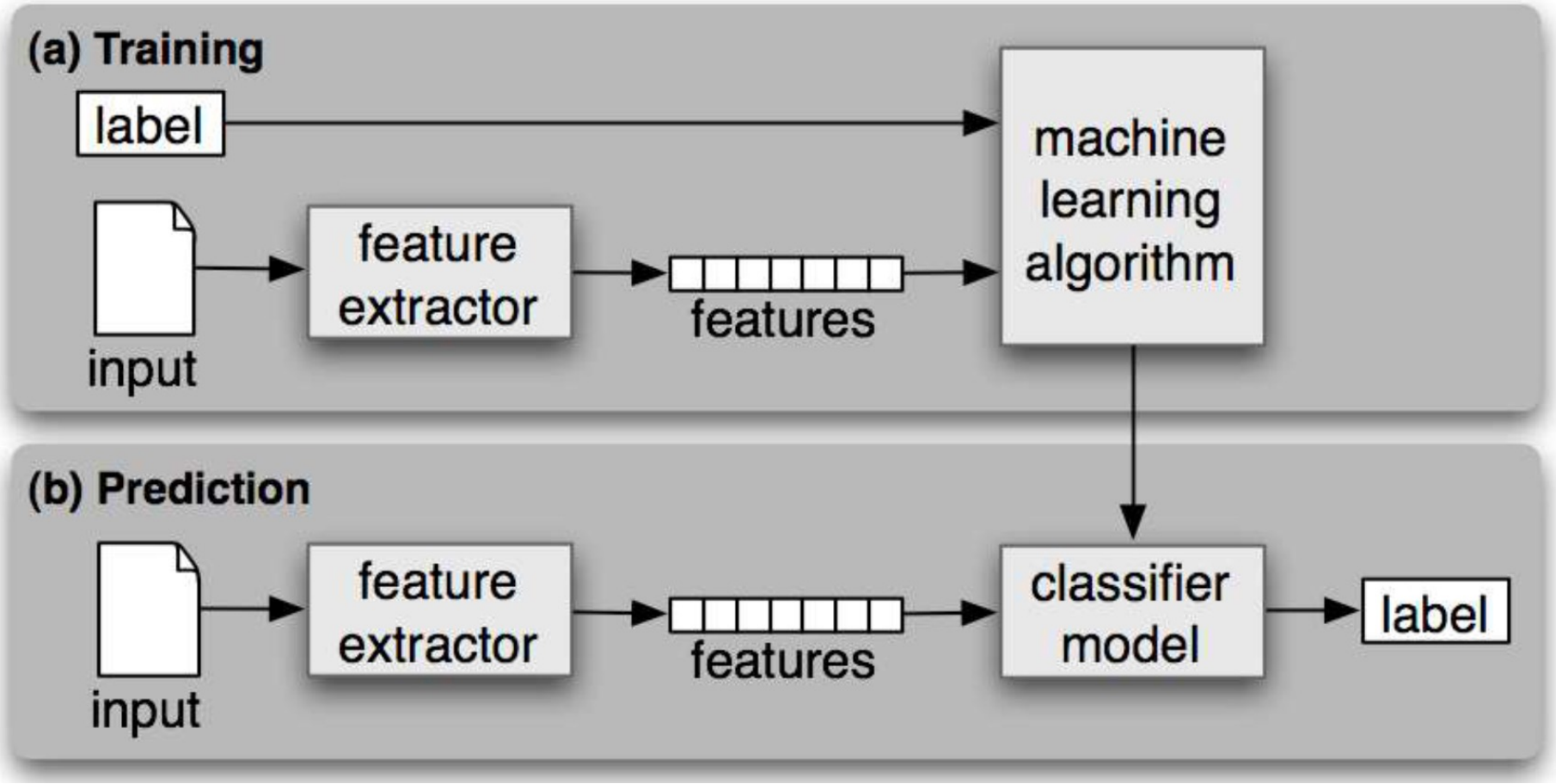
Learning

Features:
1. Color: **Radish/Red**
2. Type : **Fruit**
3. Shape
etc…

Features:
1. Sky Blue
2. **Logo**
3. Shape
etc…

Features:
1. **Yellow**
2. **Fruit**
3. Shape
etc…

Workflow


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

Workflow (Deep Learning)

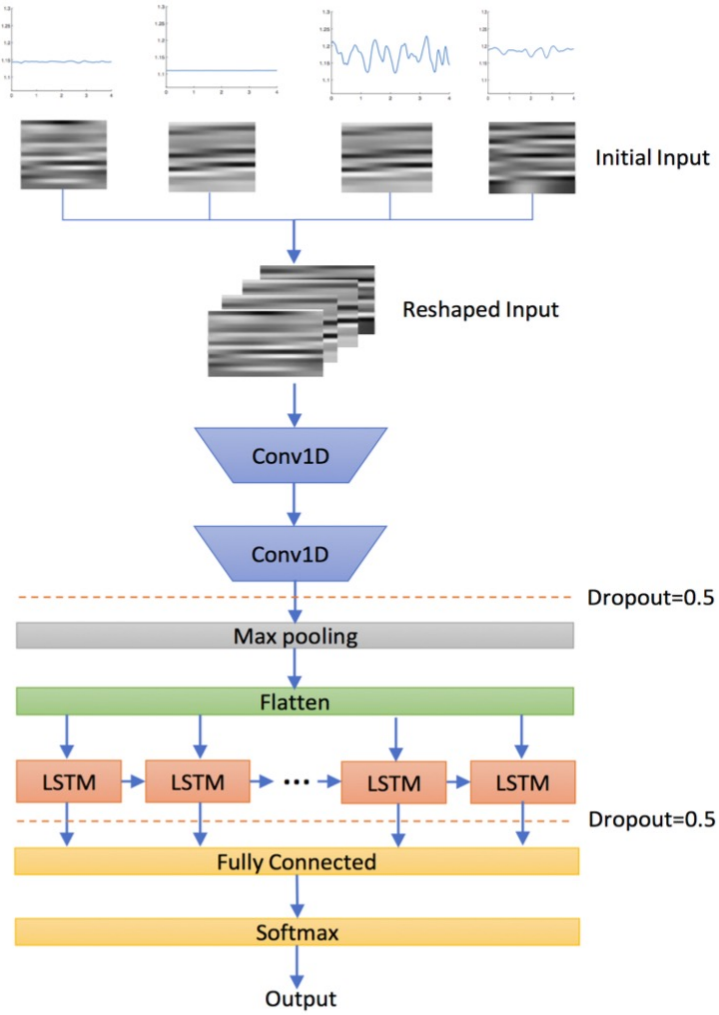
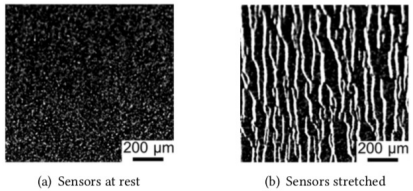
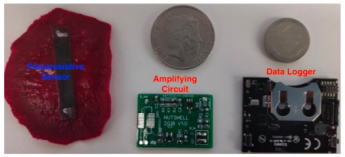


Figure 5: The CNN-LSTM model structure of E-Jacket.

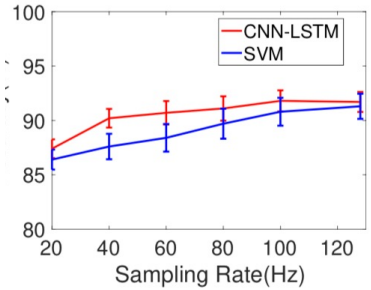
igure 7: E-Jacket prototype with 4 sensor locations (Left). ive different postures and activities to be recognized by E- cket (Right).



gure 1: Optical microscope images of strain sensor without retching (a) and under 50% strain (b).



re 6: A sensing unit comprises of a strain sensor with an lification circuit and a SensorTag as the data logger.



9: Recognition accuracy vs. sampling rates.

Table 3: Confusion matrix of 5 activities including 3 static postures without hand movements.

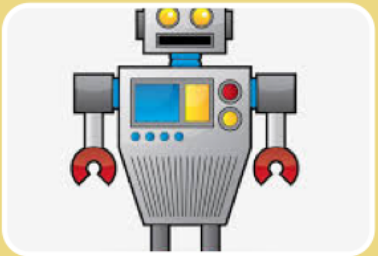
	Stand	Sit	Lie	Walk	Run	Recognition rate
Stand	87.6%	1.4%	8.1%	1.9%	0.9%	**87.6%**
Sit	0%	98.4%	1.4%	0.2%	0	**98.4%**
Lie	4.6%	2.7%	91.2%	0.9%	0.5%	**91.2%**
Walk	3.3%	1.2%	0.2%	93.4%	1.9%	**93.4%**
Run	1.6%	1.9%	1.5%	13.2%	81.7%	**81.7%**
Overall						**90.9%**

- Q. Lin *et al.*, "E-Jacket: Posture Detection with Loose-Fitting Garment using a Novel Strain Sensor," *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Sydney, Australia, 2020, pp. 49-60,.

Broad Categories

Supervised

Unsupervised

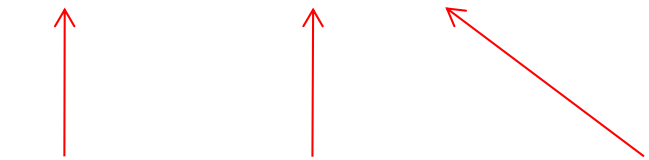


Reinforcement Learning

Supervised Learning

Framework

$$y = f(\mathbf{x})$$



output

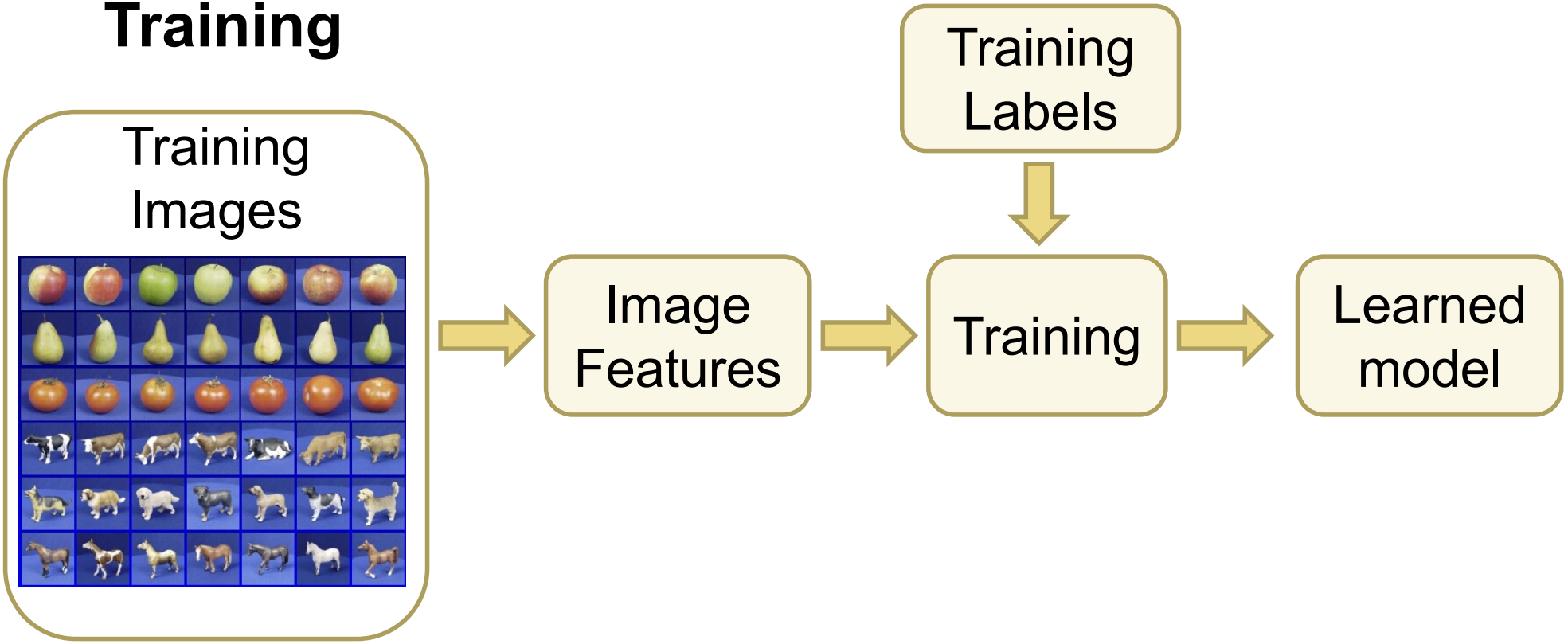
prediction function

Features extracted from raw data

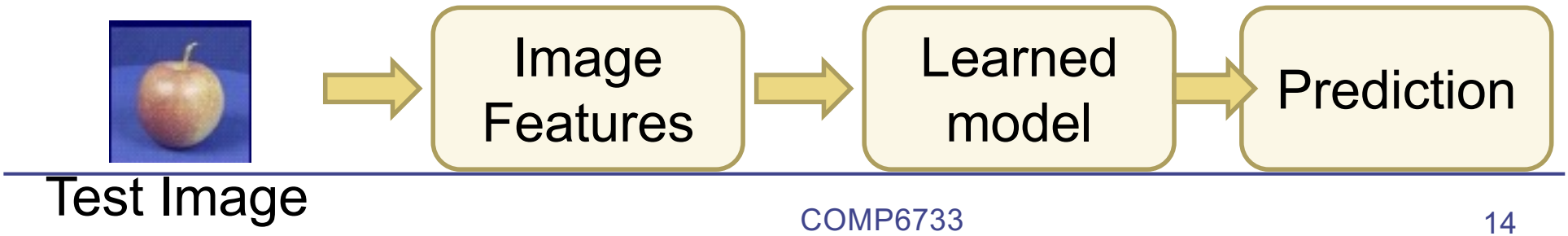
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Steps (Example where data are images)

Training



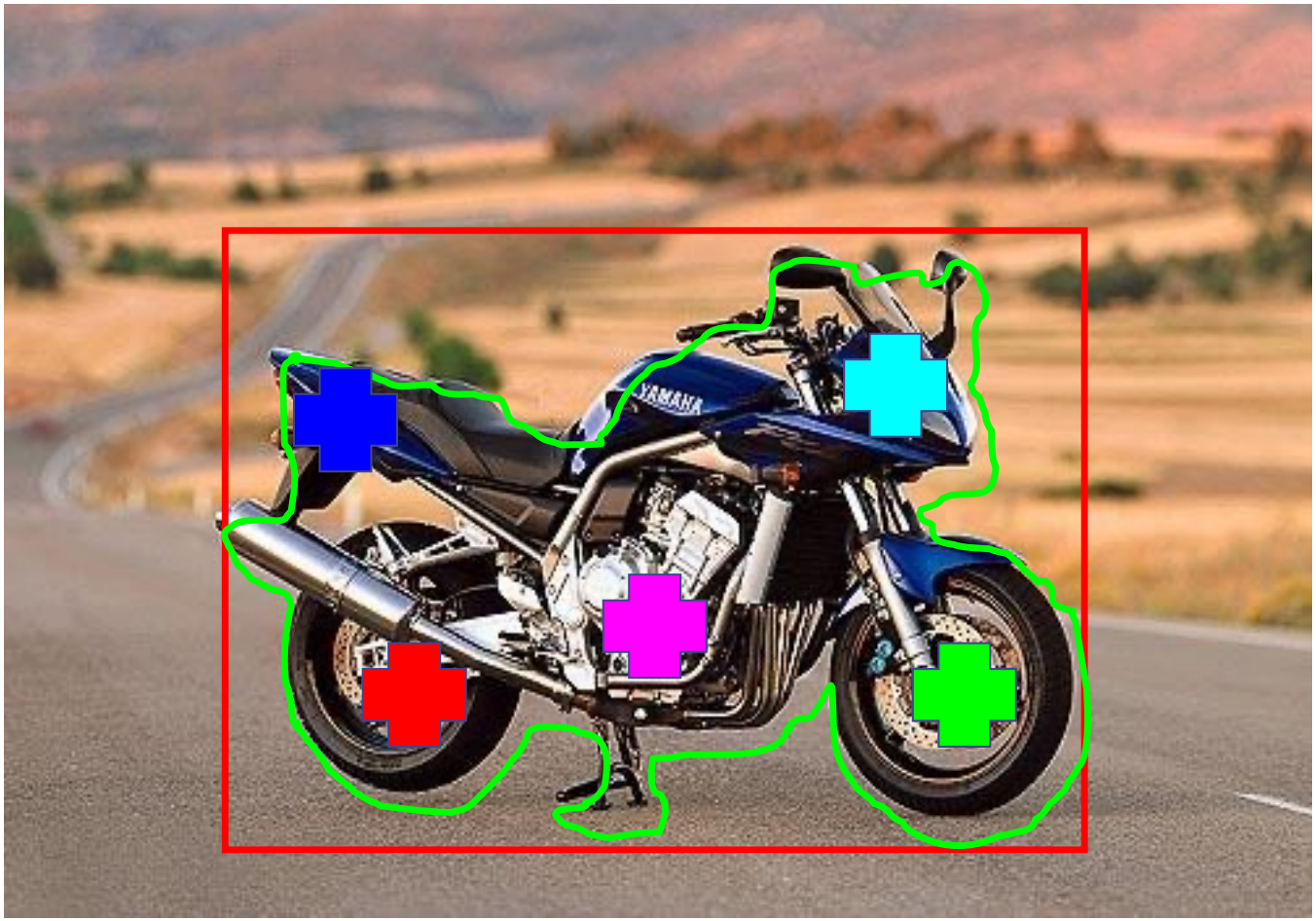
Testing



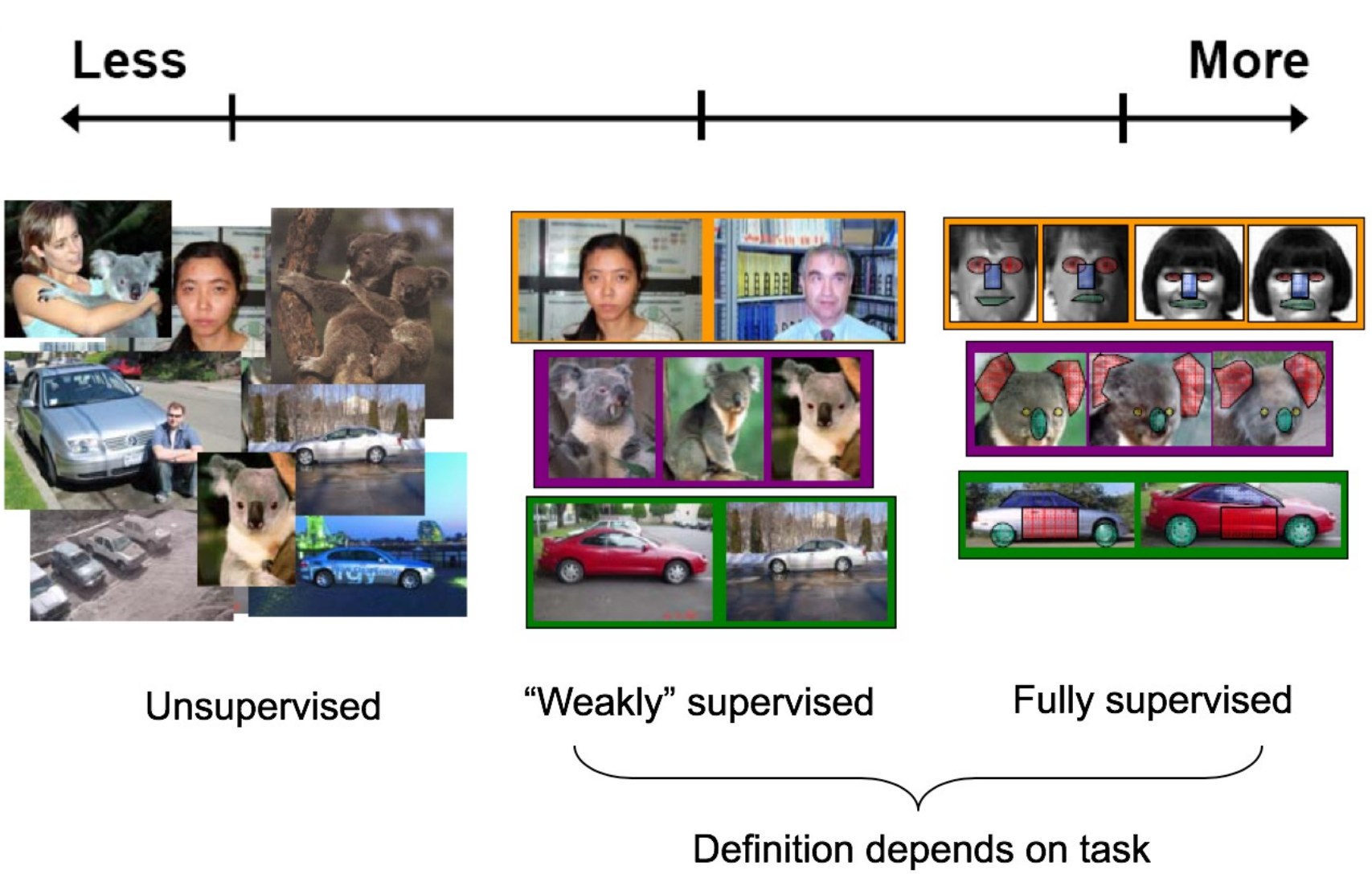
Recognition task and supervision

- Data in the training set must be annotated with the "correct answer" that the model is expected to produce

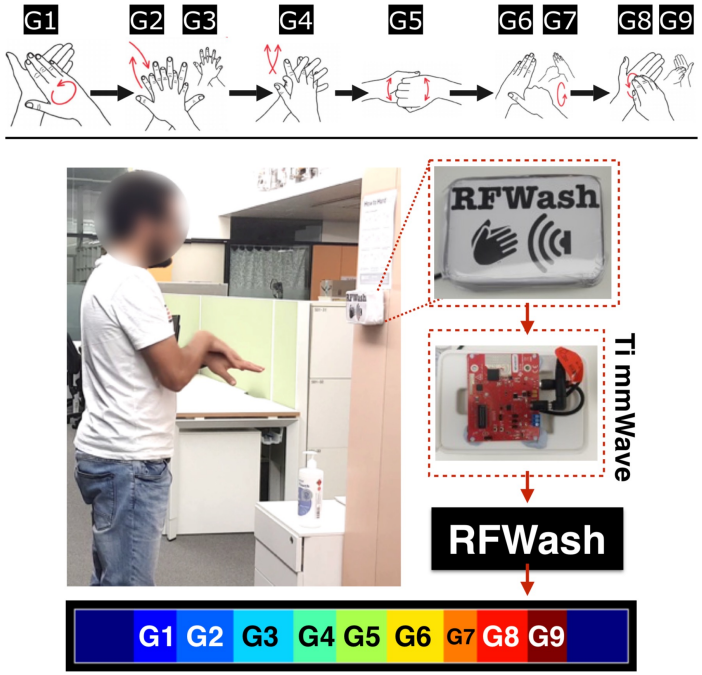
Contains a motorbike



Spectrum of Supervision



Example, Weakly learning



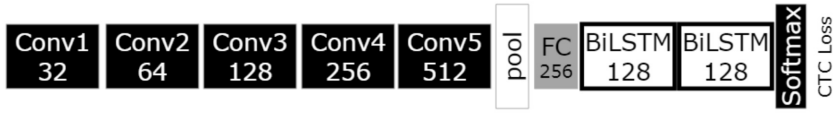


Figure 11: RFWash Network Architecture. All convolutions are 3×3 (the number of filters are denoted in each box).

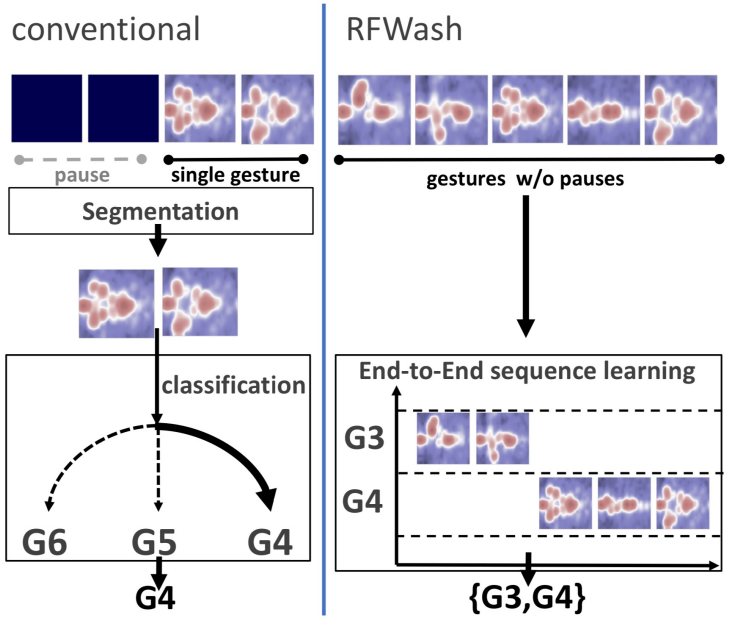


Table 5: Recognition accuracy for different deep models

Gesture	C3D	C3D(p)	DSoli	DSoli(p)	RFWash
G_{No}	92.5%	95.1%	92.5%	94.6%	**97.8%**
G_1	69.3%	72.4%	41.6%	40.9%	**92.1%**
G_2	68.3%	76.7%	88%	**89.8%**	85.4%
G_3	84.2%	**92.9%**	76.1%	77.8	87.2%
G_4	82.2%	83.9%	77.5%	80.8%	**89.7%**
G_5	84.4%	86.4%	33.1%	34.9%	**87.7%**
G_6	48.2%	43.7%	20.3%	18.8%	**71.4%**
G_7	70.4%	69.3%	74.1%	77.2%	**79.6%**
G_8	56.7%	66.3%	76.6%	**80.6%**	77.7%
G_9	66.1%	75.9%	42%	42.1%	**84.2%**
Accuracy	73.33%	77%	63.15%	64.79%	**85%**

- https://www.youtube.com/watch?v=AHIsOOI6aLI&feature=youtu.be
- https://youtu.be/hwEDIya5bx0?t=45
- "RFWash: A Weakly Supervised Tracking of Hand Hygiene Technique", A. Khamiss, B. Kusy, C.T Chou, M. Mclaws, W. Hu. The 18th ACM Conference on Embedded Networked Sensor Systems (SenSys 2020), November 2020, Yokohama, Japan.

Generalization

Training set (labels known)

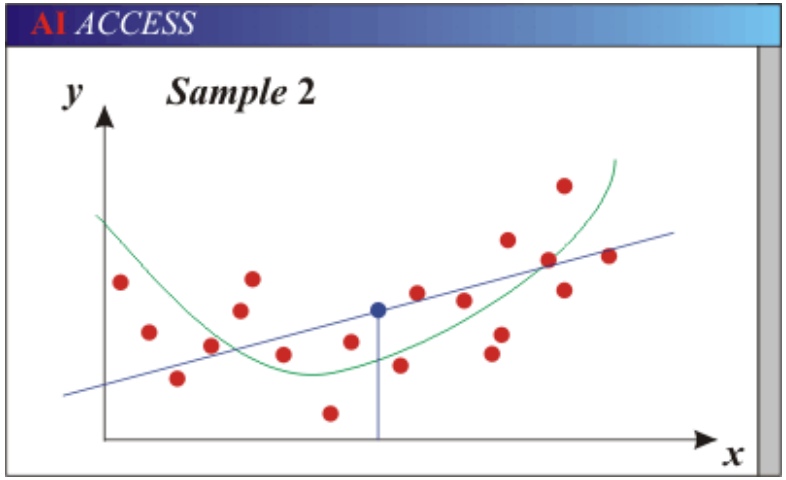
Test set (labels unknown)

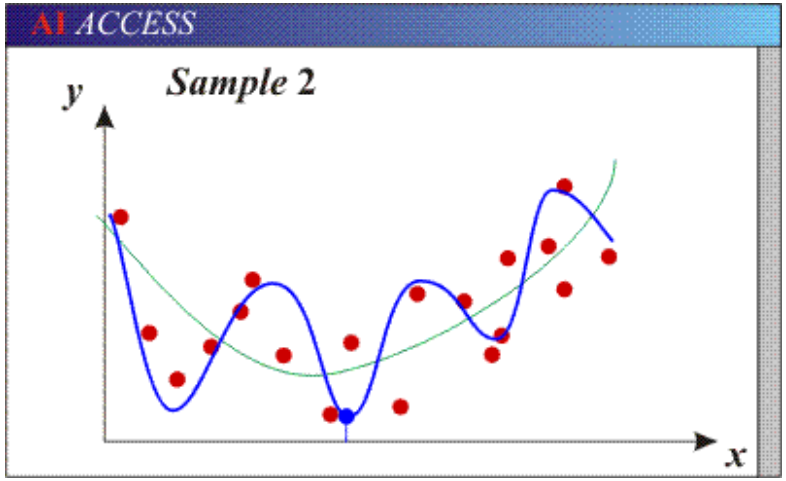
- How well does a learned model generalize from the data it was trained on to a new test set?

Generalization

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too "simple" to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Bias-Variance Trade-off

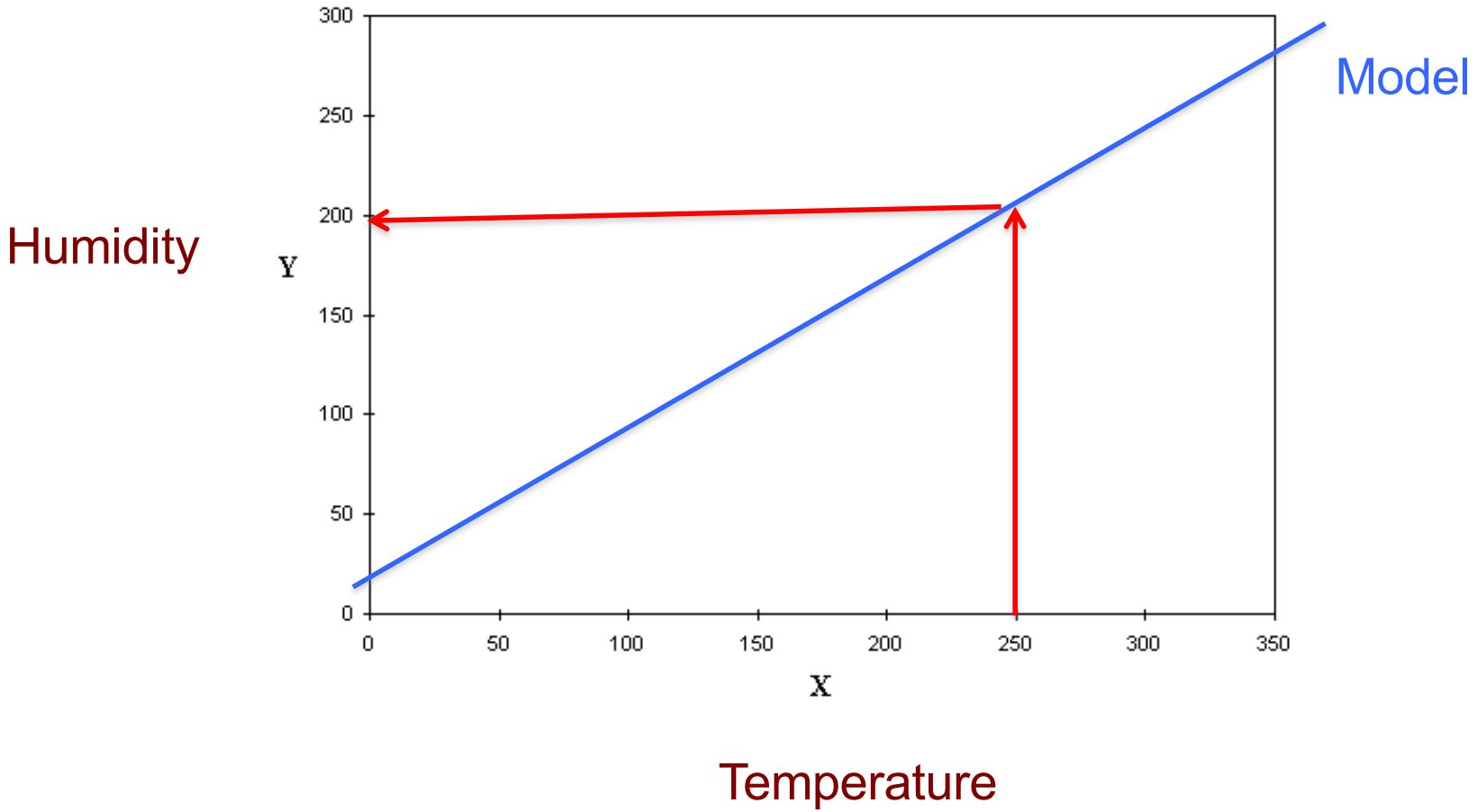




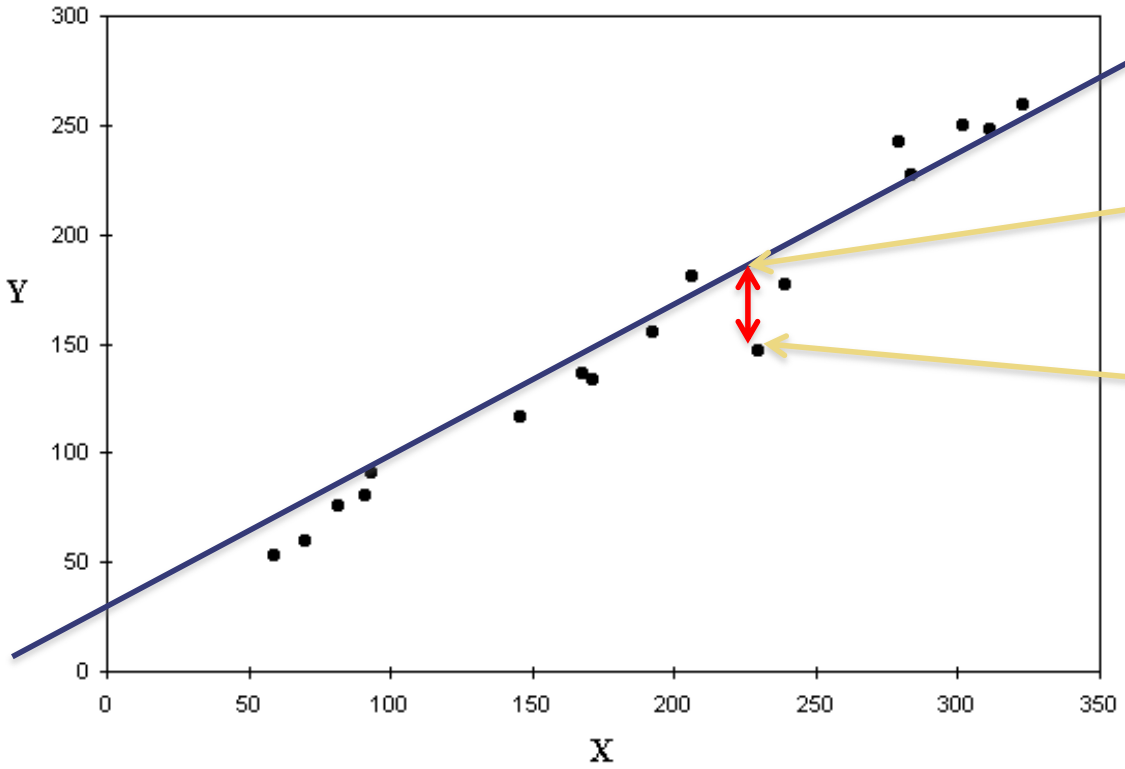
- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Least Squares Regression – 1D

PROBLEM: Predict ‘humidity’ given the ‘temperature’



LSR Math



$(x_j w_1 + w_0)$

y_j

$$min_{w_0,w_1} \sum_j ((x_j w_1 + w_0) - y_j)^2$$

$$min_w ||Xw - y||_2^2$$

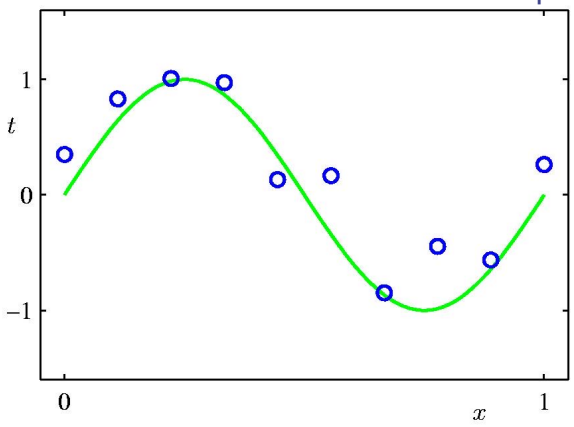
LSR Python

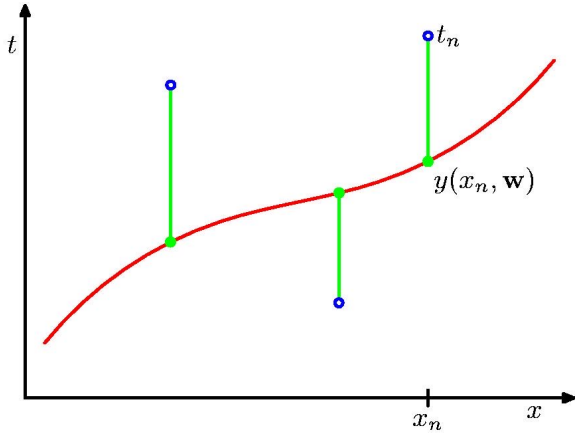
```
>>> from sklearn import linear_model
>>> clf = linear_model.LinearRegression()
>>> clf.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
>>> clf.coef_
array([ 0.5,  0.5])
```

Model Complexity: Fitting a polynomial

- The green curve is the true function (which is not a polynomial)
- The data points are uniform in x but have noise in y.
- We will use a loss function that measures the squared error in the prediction of y(x) from x. The loss for the red polynomial is the sum of the squared vertical errors.

from Bishop

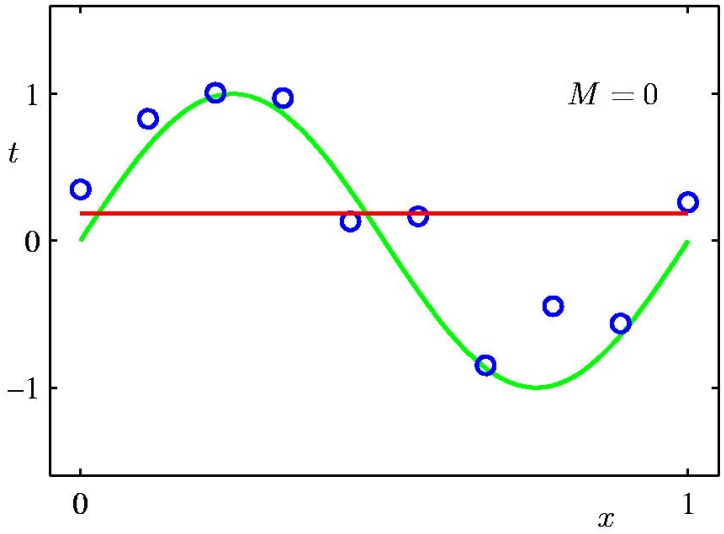


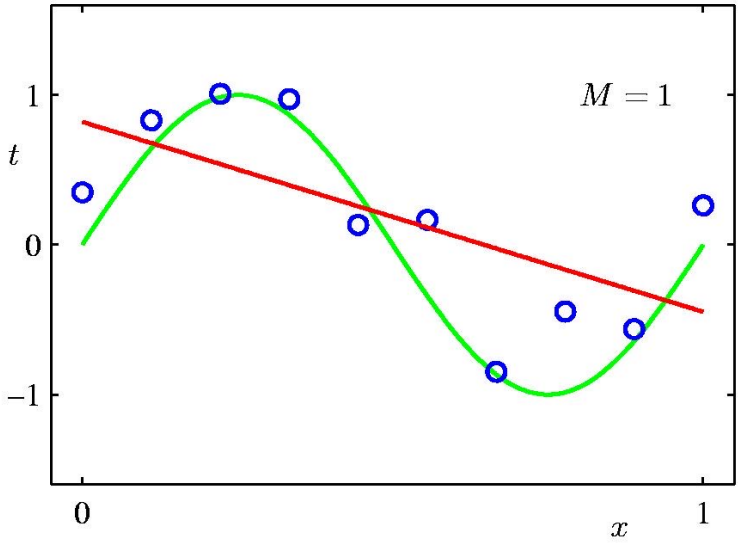


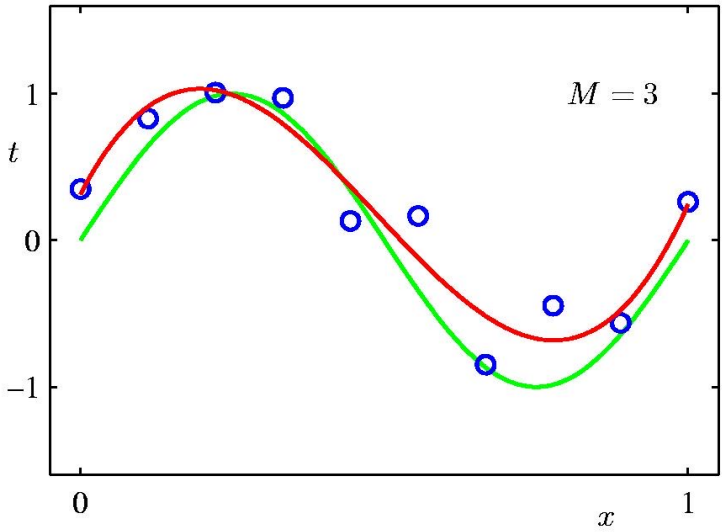
$$min_w \sum_j (w_0 + w_1 x_j + w_2 x_j^2 + \cdots + w_m x_j^m) - y_j)^2$$

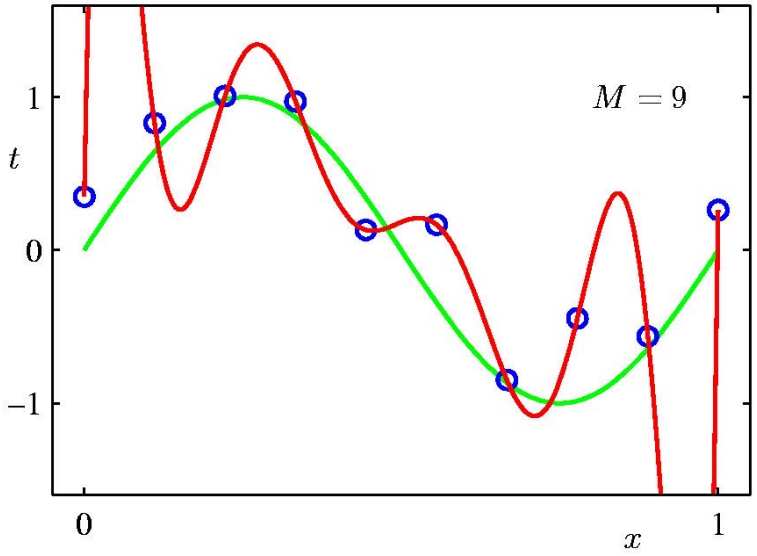
Model Complexity – Which model is best?

from Bishop









Cross validation with limited data

Select best level of complexity

$< x_j, y_j >$

x_j

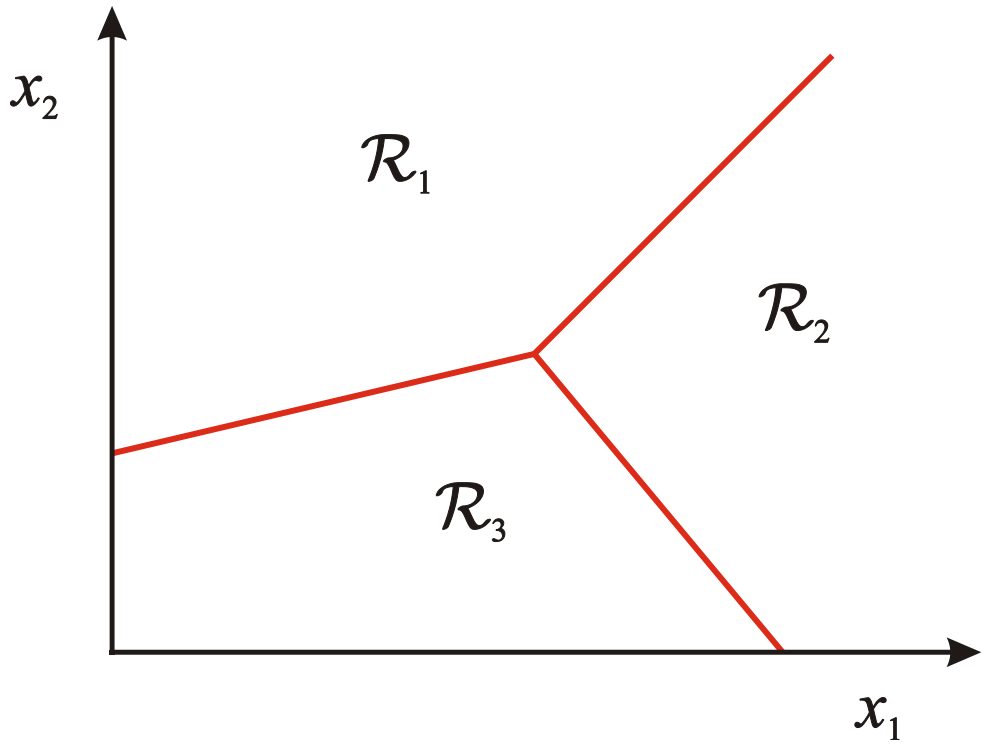
Test set

Very brief tour of some classifiers

- K-nearest neighbor
- SVM
- Boosted Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- RBMs
- Etc.

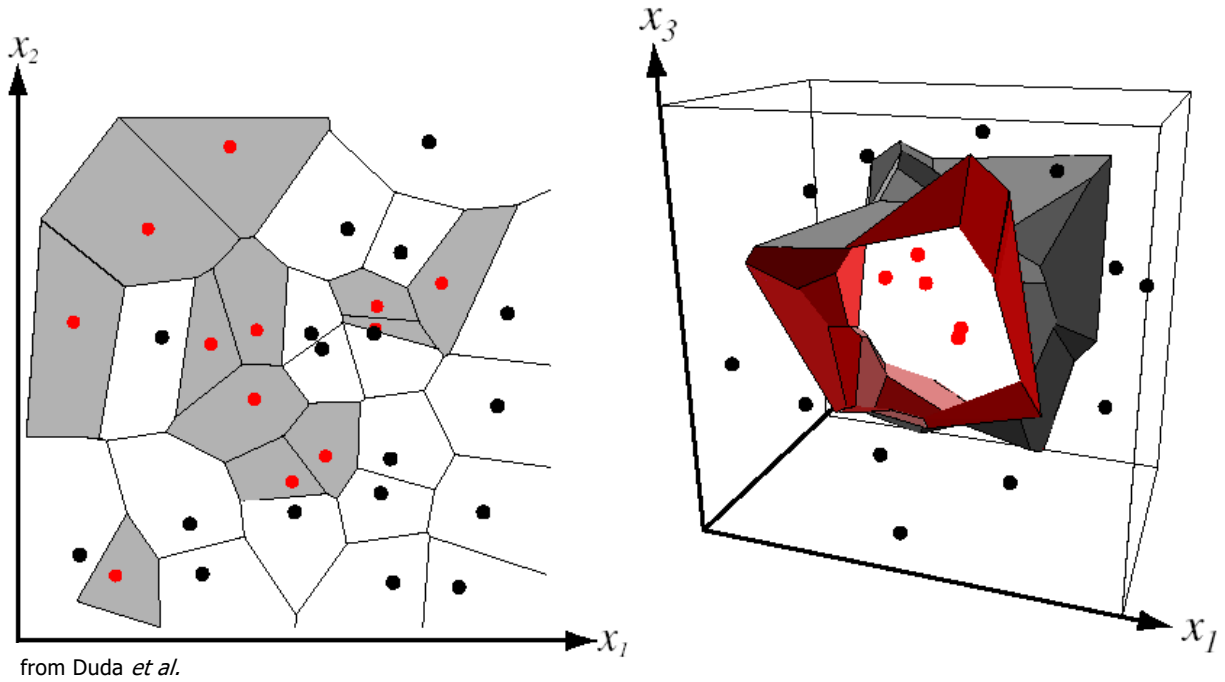
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Nearest Neighbor Classifier

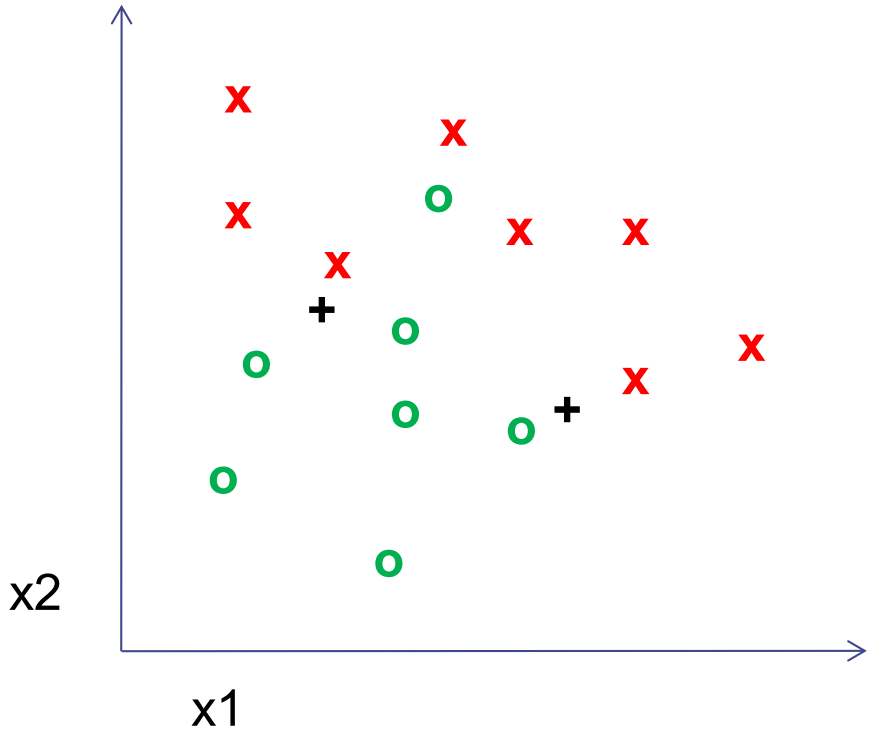
- Assign label of nearest training data point to each test data point



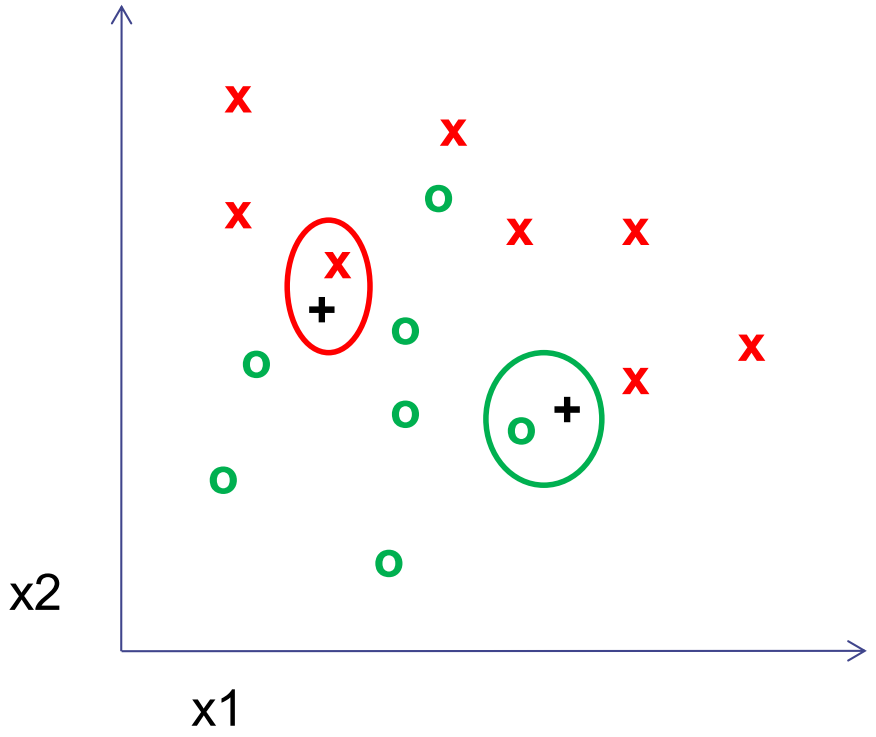
from Duda *et al.*

Voronoi partitioning of feature space
for two-category 2D and 3D data

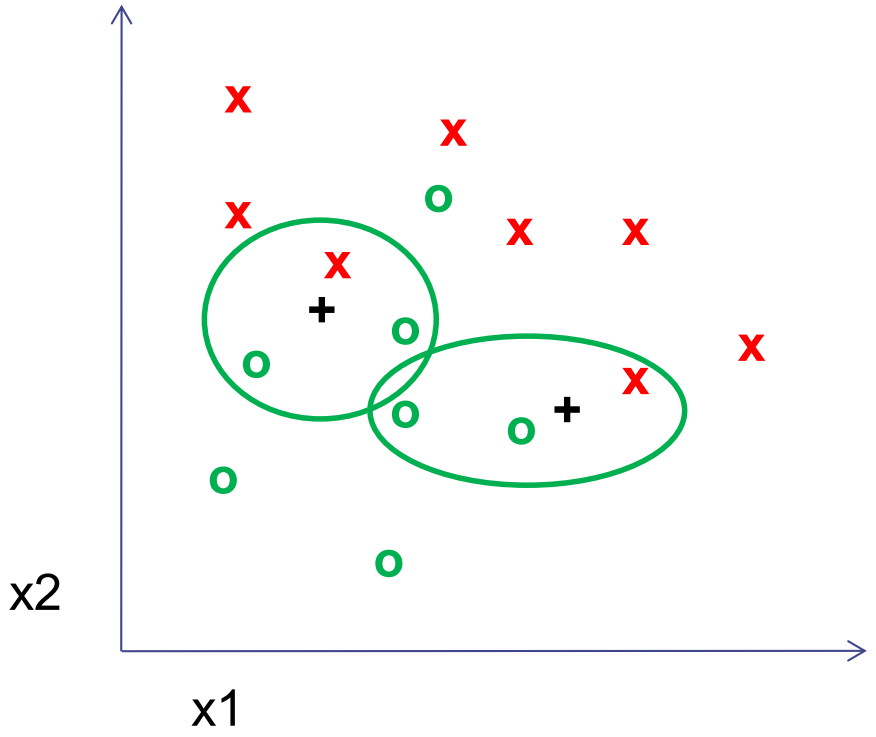
K-nearest neighbor



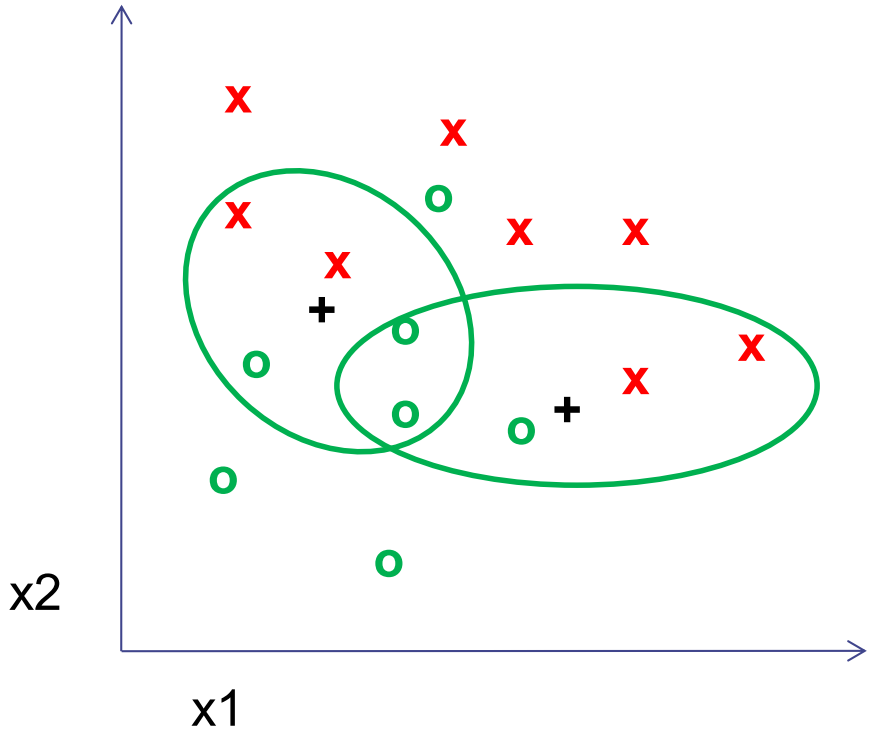
1-nearest neighbor



3-nearest neighbor

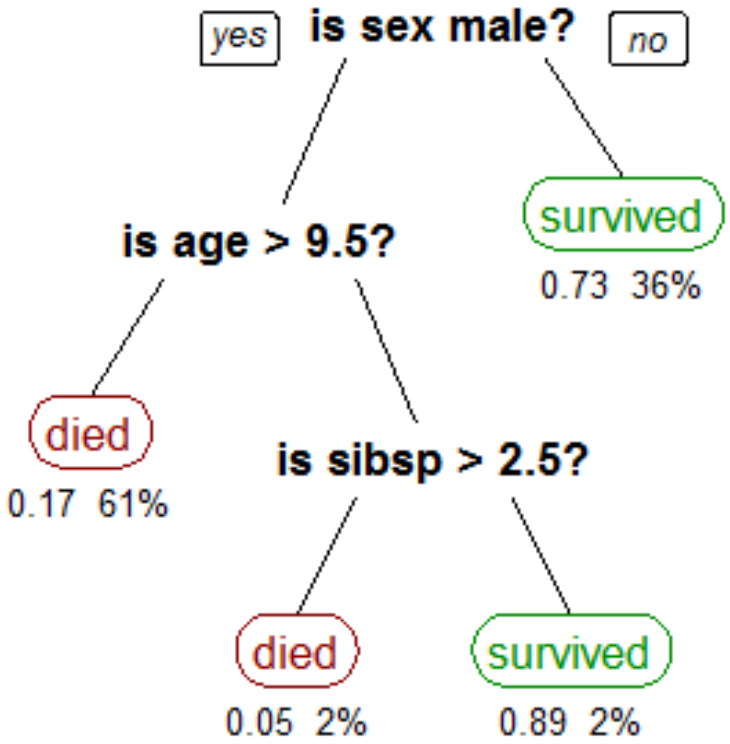


5-nearest neighbor



Classification and Regression Trees (CARTs)

- Titanic Survivors



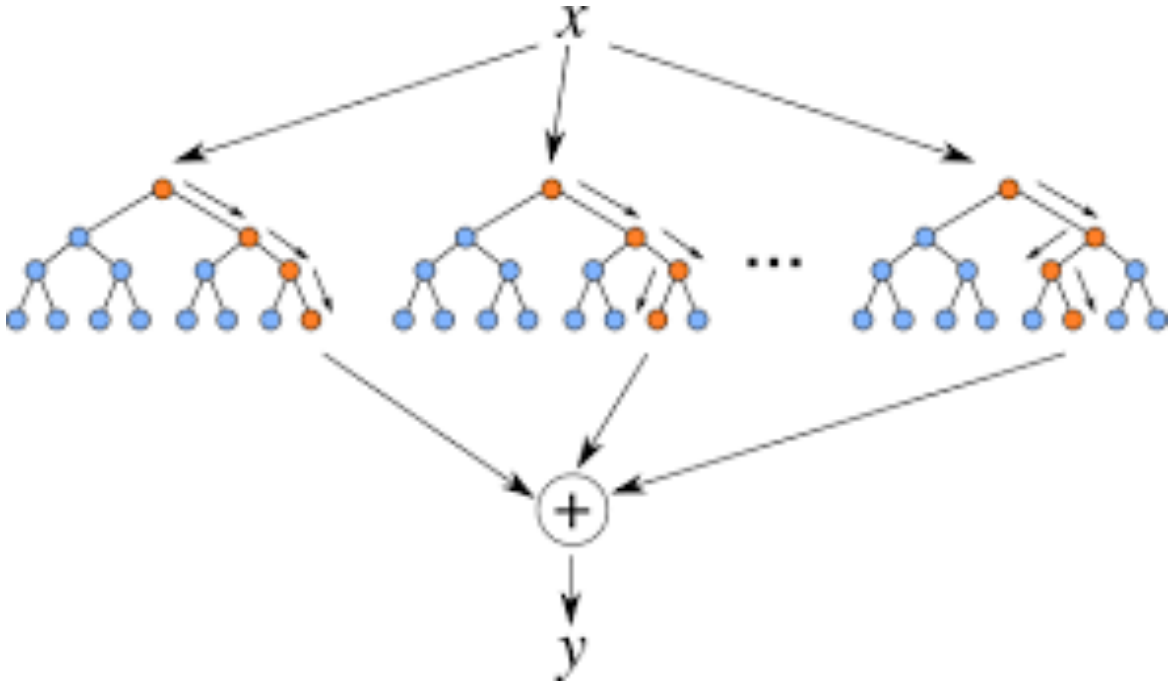
(Source: https://en.wikipedia.org/wiki/Decision_tree_learning)

sklearn.tree.DecisionTreeClassifier

sklearn.tree.DecisionTreeRegressor

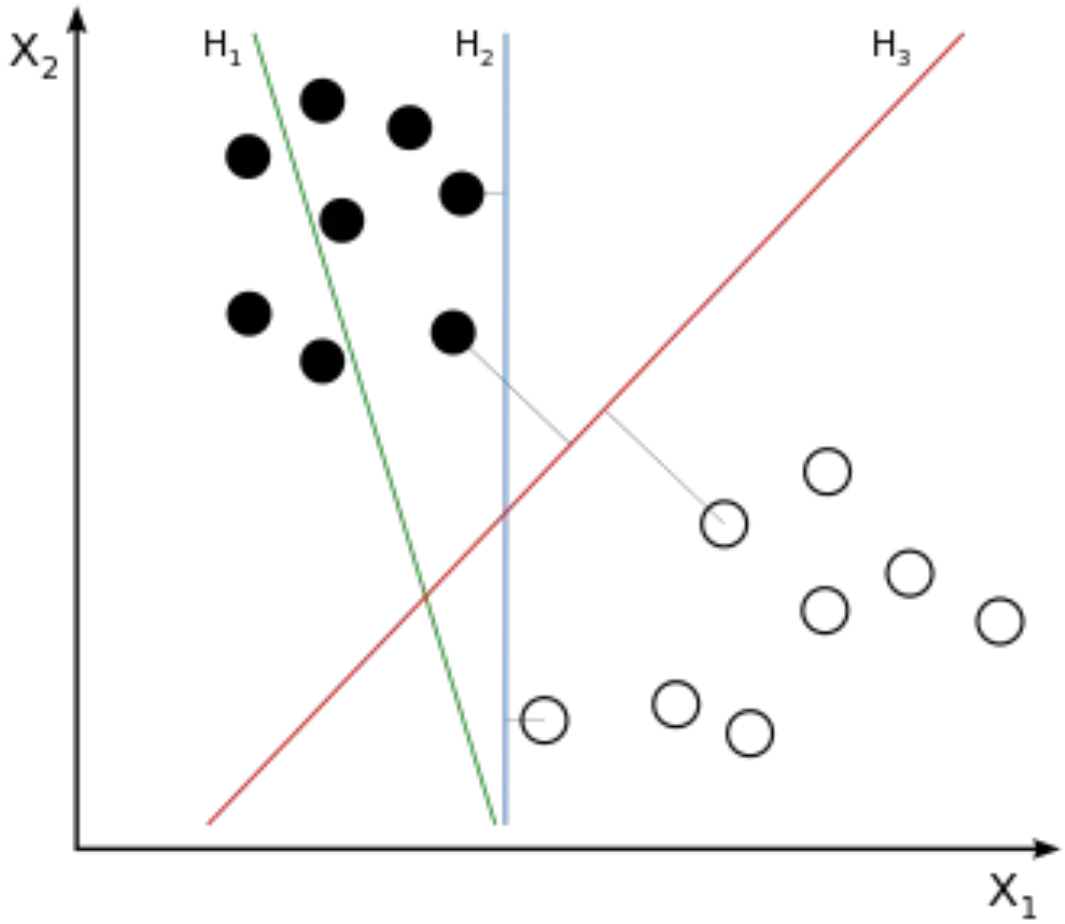
Random Forests (Ensemble Learning)

Ensembles perform better than a single model



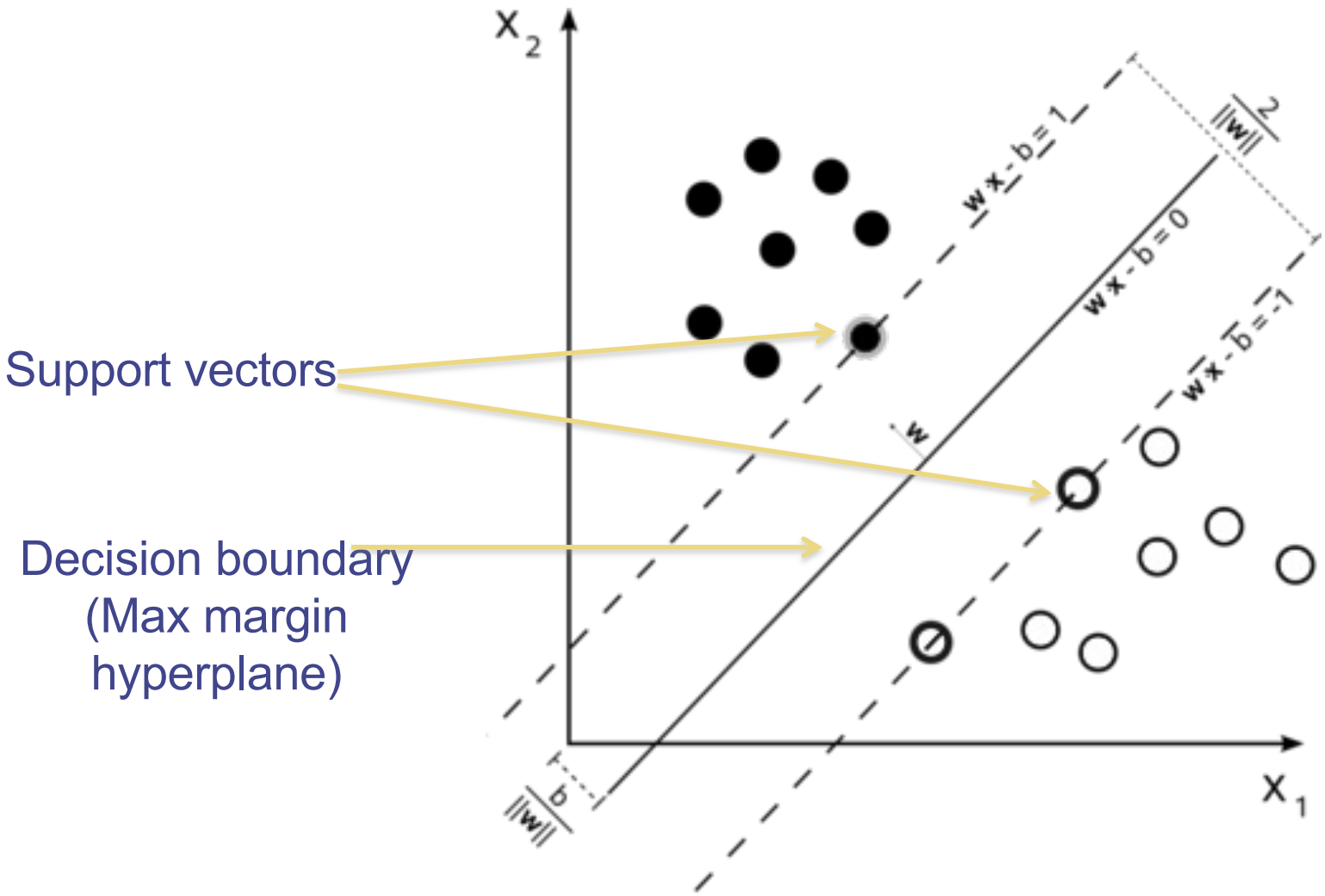
sklearn.ensemble.RandomForestClassifier

Support Vector Machine (SVM) Classification



Source: https://en.wikipedia.org/wiki/Support_vector_machine

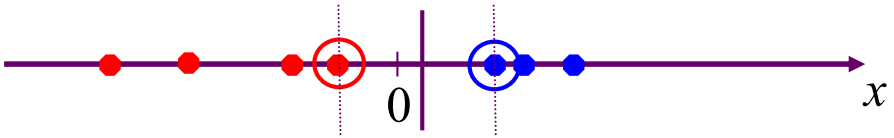
Support Vector Machines (SVM)



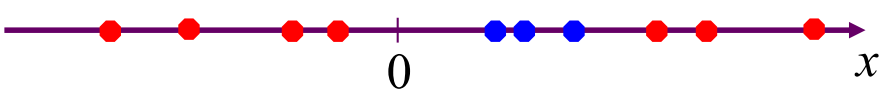
Source: https://en.wikipedia.org/wiki/Support_vector_machine

Nonlinear SVMs

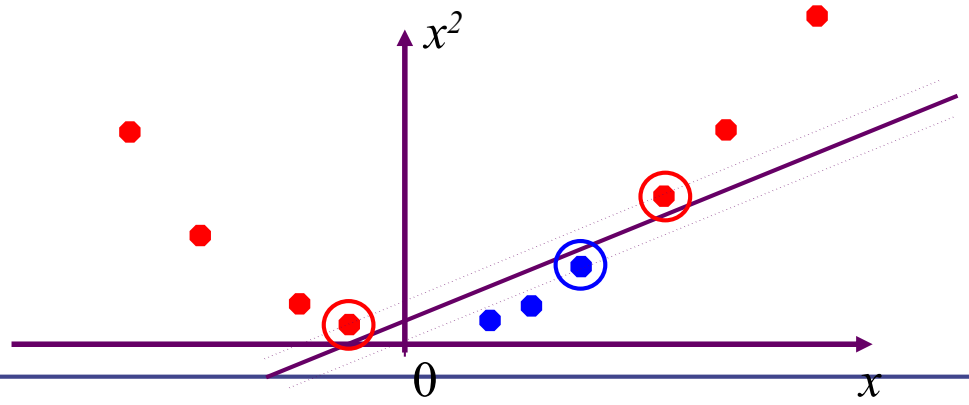
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

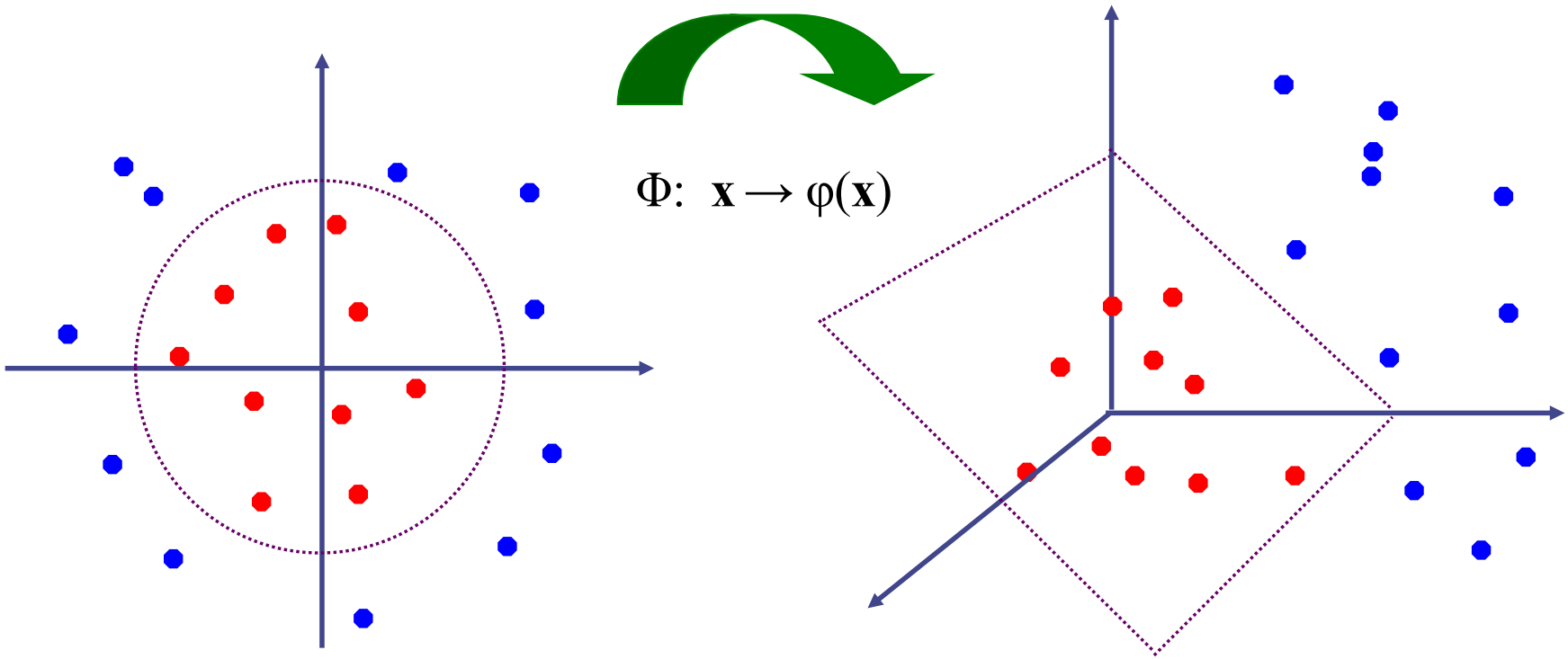


- We can map it to a higher-dimensional space:



Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



What to remember about classifiers

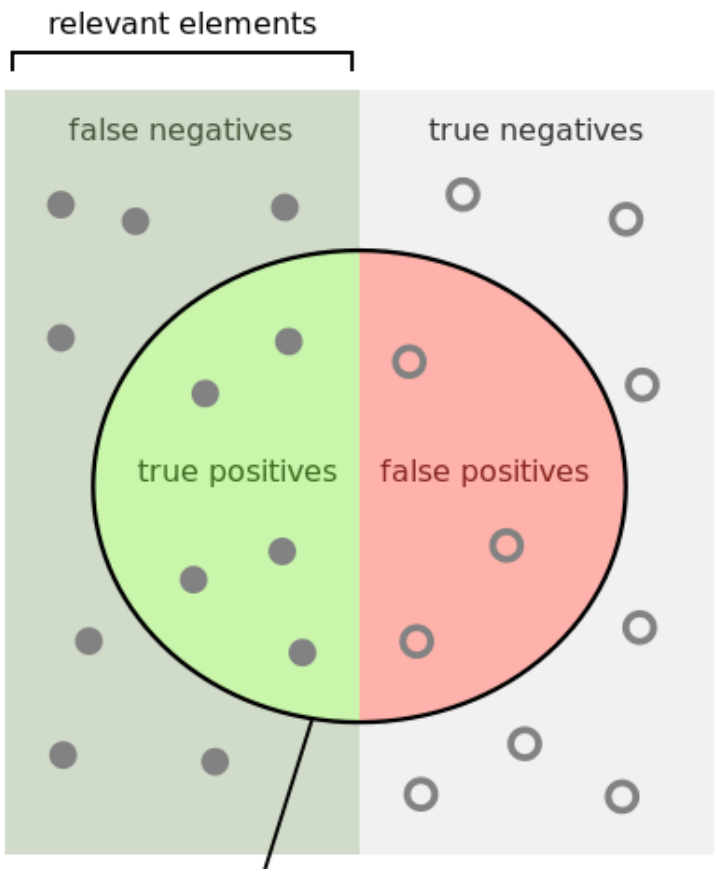
- No free lunch: machine learning algorithms are tools, not dogmas

- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Metrics: Confusion Matrix

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Precision / Recall



How many selected items are relevant?

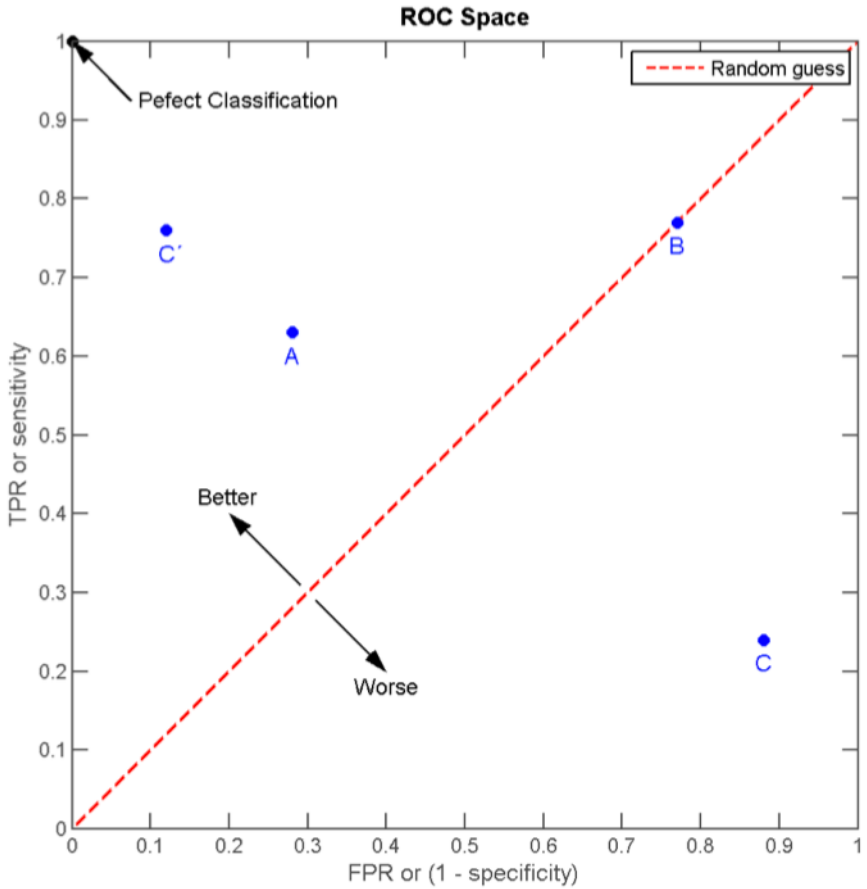
$$\text{Precision} = \frac{tp}{tp + fp}$$

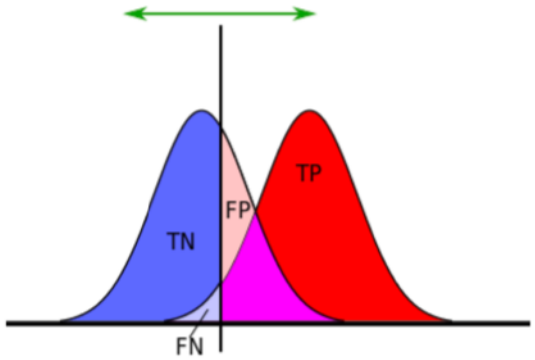
How many relevant items are selected?

$$\text{Recall} = \frac{tp}{tp + fn}$$

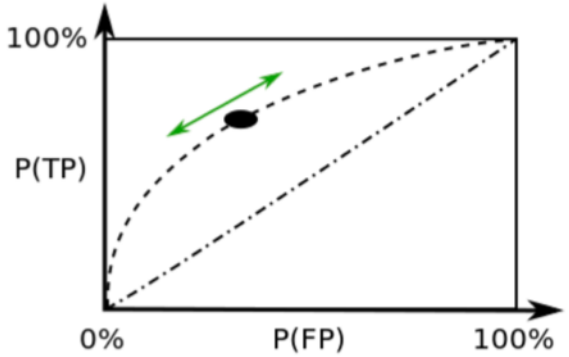
- 100% precision at the expense of recall?
- 100% recall at the expense of precision?

ROC Curves





TP	FP
FN	TN



Generative Adversarial Network (GAN) Example

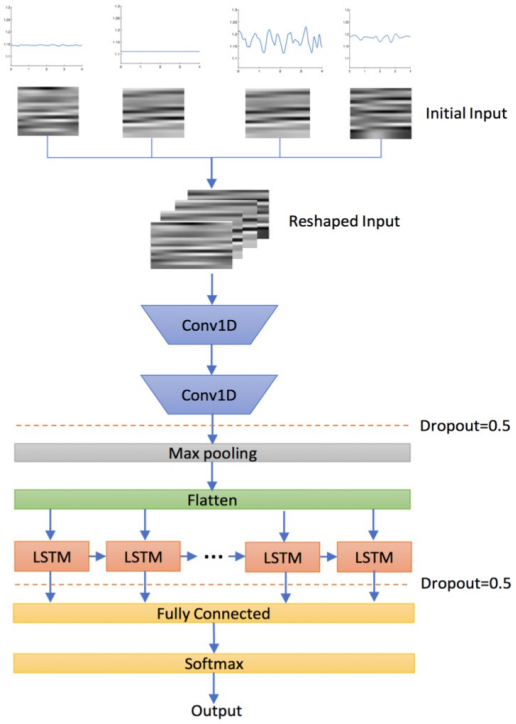


Figure 5: The CNN-LSTM model structure of E-Jacket.

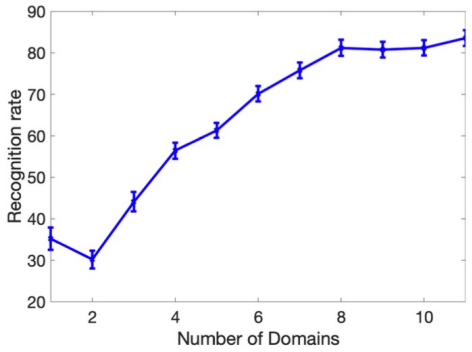


Fig. 11. Impact of number of domains on recognition accuracy.

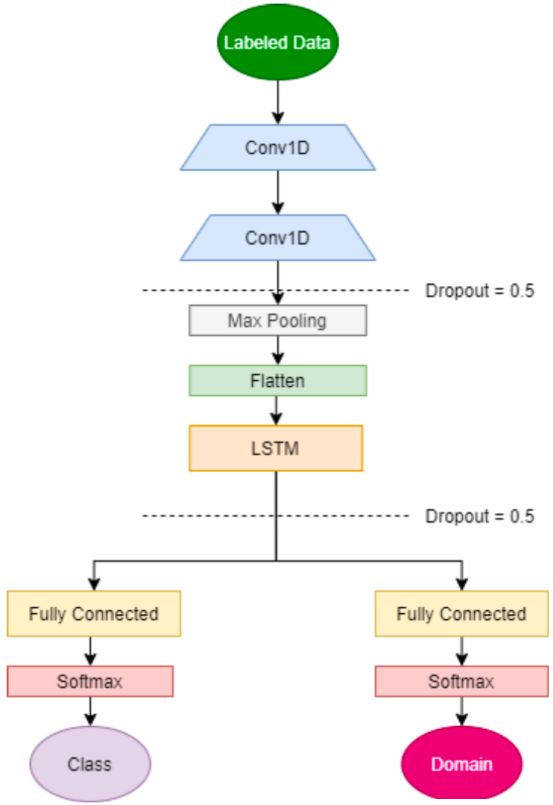
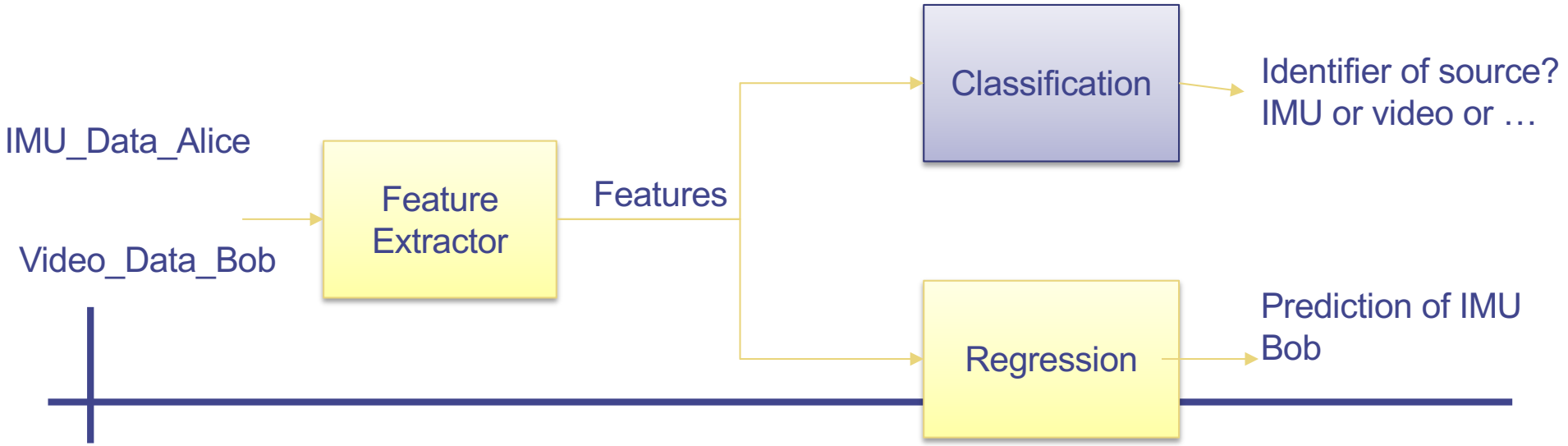


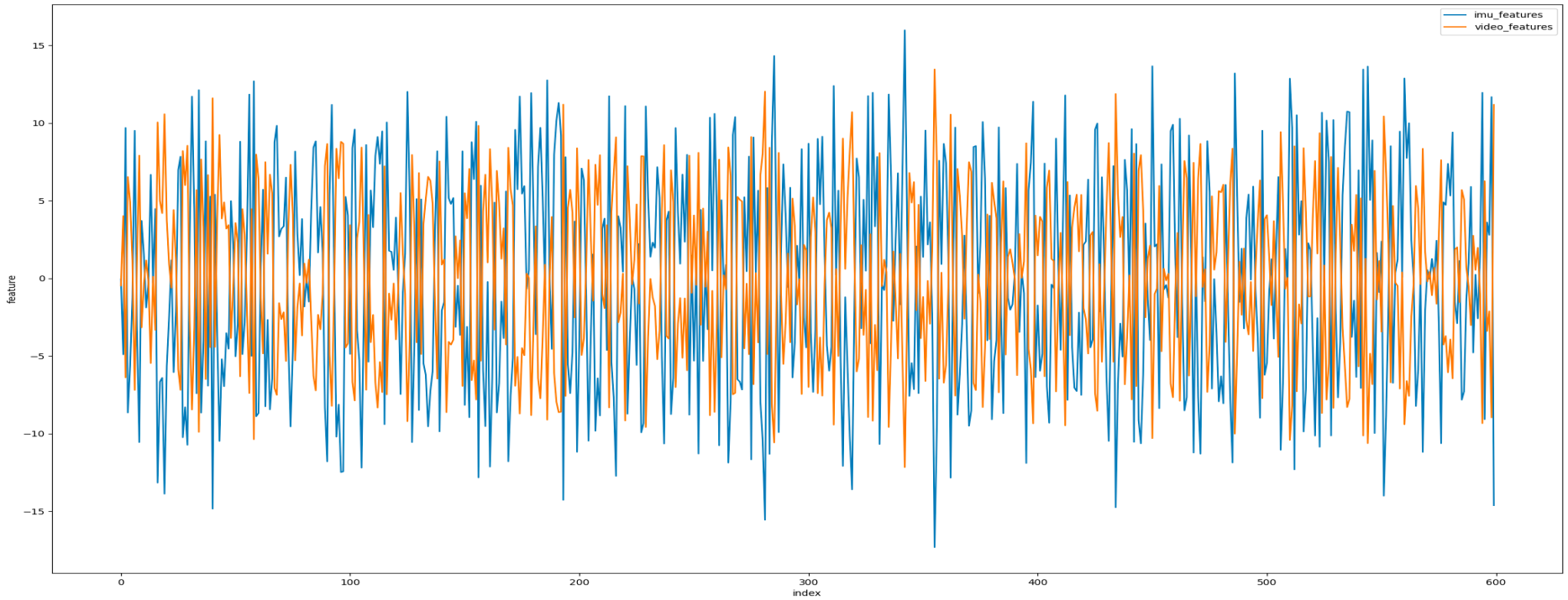
Fig. 9. The adaptive CNN-LSTM model structu

Fig. 2. Prototype with 4 sensor locations (Left). Five different postures and activities to be recognized by the smart garment system (Right).

No.	Without domain discriminator path	CNN-LSTM with domain discriminator path
1	76	96.9
2	72.8	82.8
3	73	85.9
4	68.6	85.7
5	66.9	74.7
6	75.7	72.1
7	73.9	87.9
8	77.3	85.8
9	68.2	80.8
10	73.8	71.2
11	78.6	88.1
12	76.6	82.3
Overall	73.5 ± 3.8	82.9 ± 7.4

Video side channel attack. IMU and video are two classes

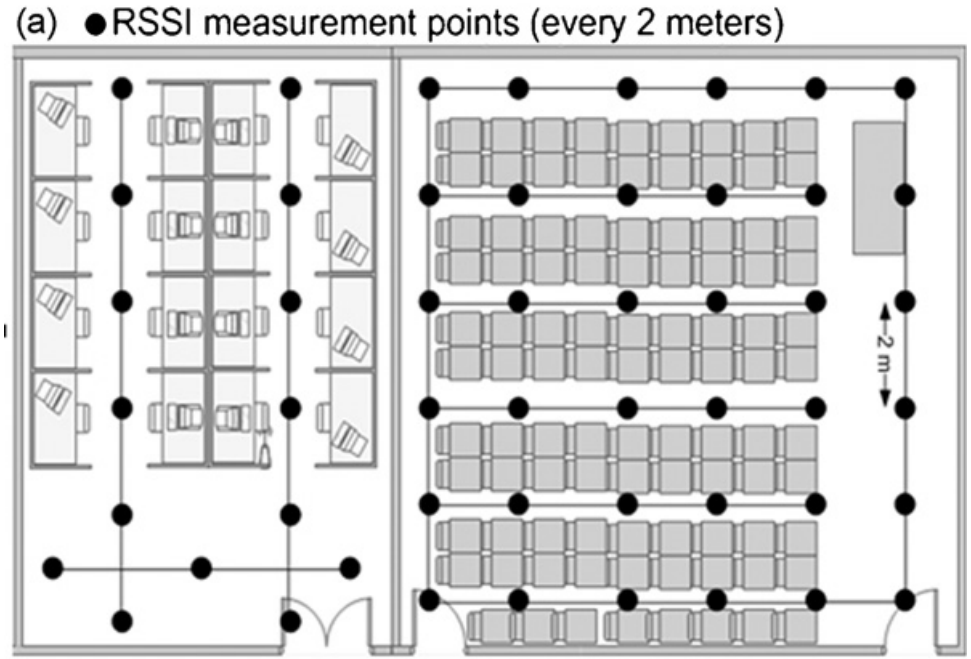


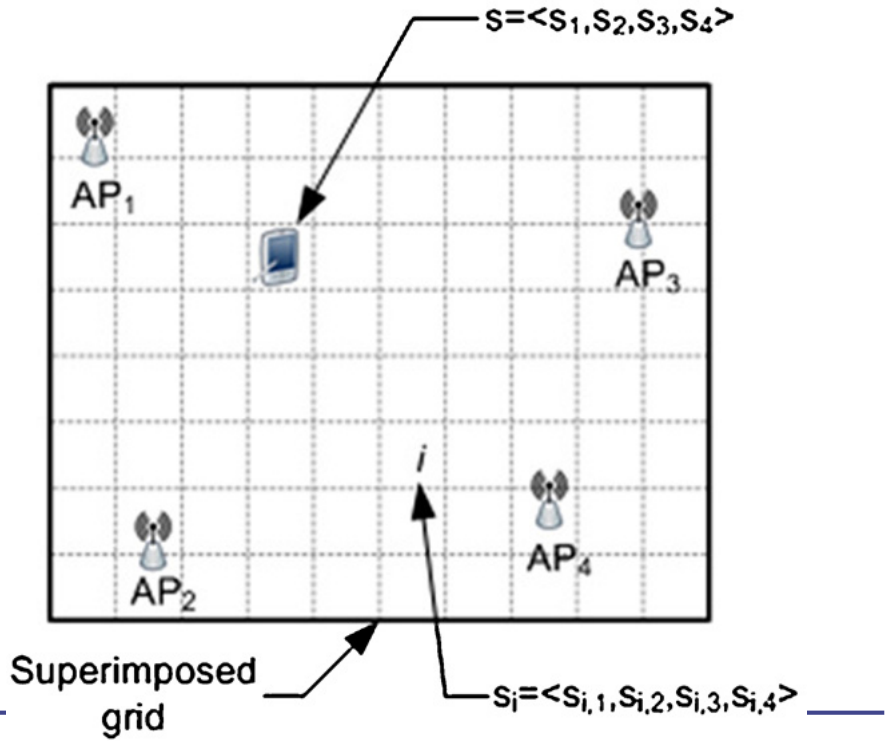


- Machine Learning
- Localisation (WiFi fingerprint-based)

RSSI fingerprinting

- Basic ideas
  - Divide the indoor space into small cells
  - Number of anchors = n
  - Within each cell, measure the RSSI from each anchor
  - Let R(c,j) = RSSI of anchor j at cell c
  - Fingerprint of cell c = [R(c,1) R(c,2) …. R(c,n)]



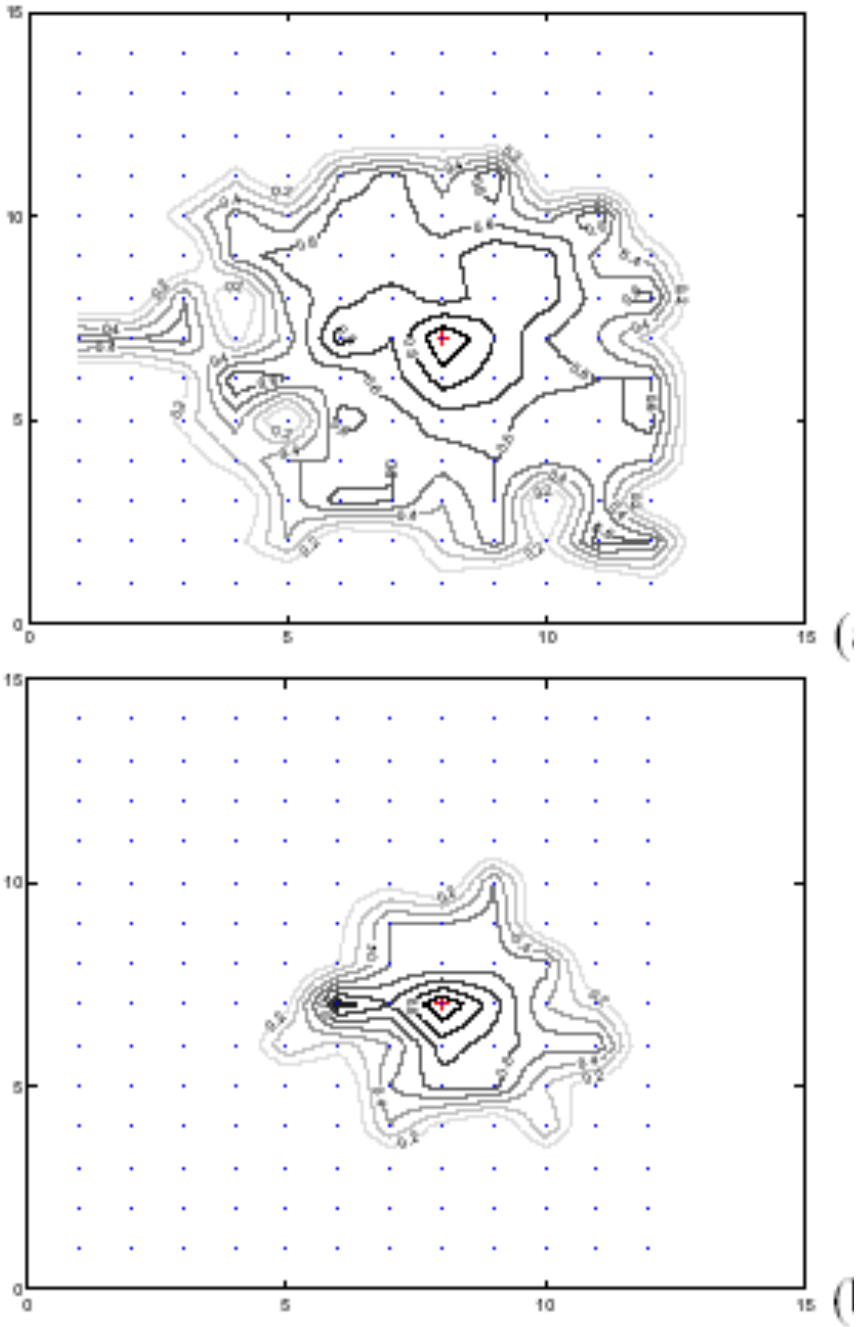


Naïve method

- If each cell has a unique RSSI fingerprint and its fingerprint doesn't change with time
- We can use the fingerprint to identify a cell
- Unfortunately, the RSSI fingerprint at each cell changes over time
  - Same problem that we have seen before, the RSSI measurements at a given position are not the same

Experimental link quality (2)

- Simple, circular shape of “region of communication” – not realistic
- Instead:
  - Correlation between distance and loss rate is weak; iso-loss-lines are not circular but irregular
  - Asymmetric links are relatively frequent (up to 15%)
    - Asymmetric link: Node A can hear Node B but not the other way
  - Significant short-term PER variations even for stationary nodes



Use multiple measurements

- Within each cell, measure the RSSI fingerprint m times
- Let R(c,j,p) = RSSI of anchor j at cell c during the p-th measurements
- Let f(c,p) = [R(c,1,p) R(c,2,p) …. R(c,n,p)]
- Fingerprint of cell c = [f(c,1) f(c,2) f(c,3) f(c,4) … f(c,m)]

How to determine which cell?

- Let f(c,p) = [R(c,1,p) R(c,2,p) …. R(c,n,p)]
- Fingerprint of cell c = [f(c,1) f(c,2) f(c,3) f(c,4) … f(c,m)]
- Put sensor in some cell and let it measure the RSSI from the n anchors, let us say the measurements are

  r = [r(1) r(2) … r(n)]
- If the vector r is exactly the same as f(x,y), then we can say that the sensor is in cell x
- However, we seldom get an exact match

Nearest neighbour (NN) matching

- Fingerprint of cell c = [f(c,1) f(c,2) f(c,3) f(c,4) … f(c,m)]
  - c = 1, …, C
- Closest match of r
  - Note: Both r and f(c,p) are vectors
- Error e(c,p) = distance between r and f(c,p)
- Error e(x,y) is the least among all e(c,p) (c = 1,…,C; p = 1,…,m) then we say the sensor is at cell x

k-Nearest neighbour (k-NN) matching

- Fingerprint of cell c = [f(c,1) f(c,2) f(c,3) f(c,4) … f(c,m)]
  - c = 1, …, C
- Vector to match = r
- Error e(c,k) = distance between r and f(c,k)
- In k-NN matching where k is a parameter, we consider the k closest matches to r
- Example: k = 5
  - The closest 5 matches of r are f(7,2) f(3,1) f(4,10) f(7,5) f(7,3)
  - Most number of matches come from cell 7, so we say the sensor is cell 7

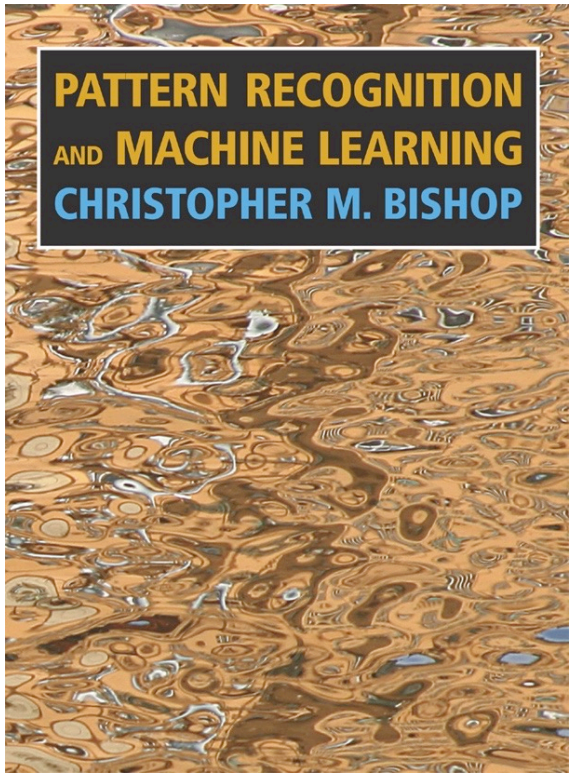
RSSI fingerprinting

- Many refinements have been proposed
  - Use statistical methods
  - Deep learning

- What do you think is the most labour intensive part of RSSI fingerprinting?

References

[bishop]



[tibshirani]