CS412
Machine Learning
Introduction – 1b

Berrin Yanikoglu
Sabancı Üniversitesi

Different Learning Paradigms

A Loose Taxonomy of ML

- **Supervised learning**–where the algorithm learns a model that maps a set of inputs to a set of desired outputs, using labelled data.
- **Unsupervised learning**–where the algorithm discovers structure, similar patterns, sub-spaces, etc.
- **Semi-supervised or weakly supervised learning**–an intermediate supervised and unsupervised setting
- **Reinforcement learning**–where the algorithm learns a policy or a model in order to maximize a reward value without explicit feedback at intermediate steps.

Supervised Learning

- Training set of (**x**,y) pairs where y is the desired target label given by "teacher"

$$D = \{ (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots , (\mathbf{x}_N, y_N)\}$$

- Learn the unknown mapping y=f(**x**) so as to predict the target value for a future/unseen input **x**.

Classification: Label y is a category

- Gender classification ($y \in \{0,1\}$)
- Spam classification ($y \in \{0,1\}$)
- Digit classification ($y \in \{0..9\}$)

Regression: Label y is a scalar ($y \in \mathbb{R}$)

- Credit scoring
- Sales prediction

Classification

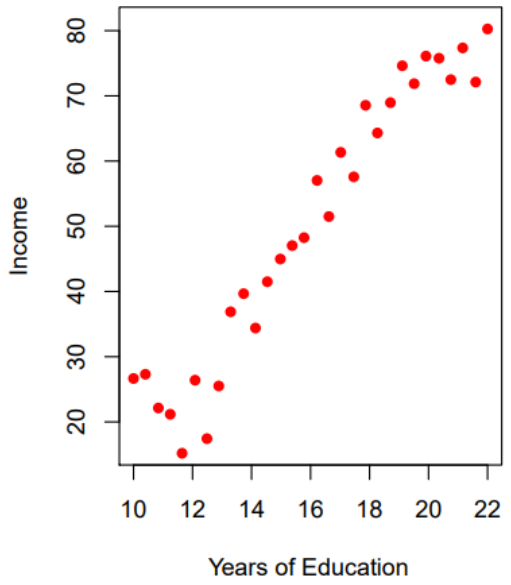
- Target to learn is a class label (binary or categorical label)

- Examples:
  - Medical diagnosis: mammogram cancerous or not?
  - Credit card applications or transactions: fraud or not?
  - Spam filtering in email: spam or not?
  - Mail categorization: inbox or junk or promotions of forum?
  - Speech understanding: which word is spoken?
  - Handwritten letters: which word/letter is this?

Regression

- Target to learn is a scalar
- Examples:
  - Credit scoring: given the bank, employment… data about a person, what should be their credit score
  - House price prediction: Given features of a house, what is a fair price?
  - Sentiment Analysis: Given a text, what is the scalar value of the sentiment in that text

How does income relate to the number of years of education?



MNIST Dataset

Imagine the **digit image classification** problem

- historically called Optical Character Recognition (OCR)

We want to **build (learn)** a machine learning **model** that can take a binary image and output its digit (**classify** it)

We will be using this very well-known dataset in our first homeworks.

Figure 1: MNIST Dataset

MNIST Dataset

Imagine the **digit image classification** problem

- historically called Optical Character Recognition (OCR)

We want to **build (learn)** a machine learning **model** that can take a binary image and output its digit (**classify** it)

We will be using this very well-known dataset in our first homeworks.

Figure 1: MNIST Dataset

Black pixels are 0,
White pixels are 1

or

Black pixels are 0,
White pixels are 255

 ≡

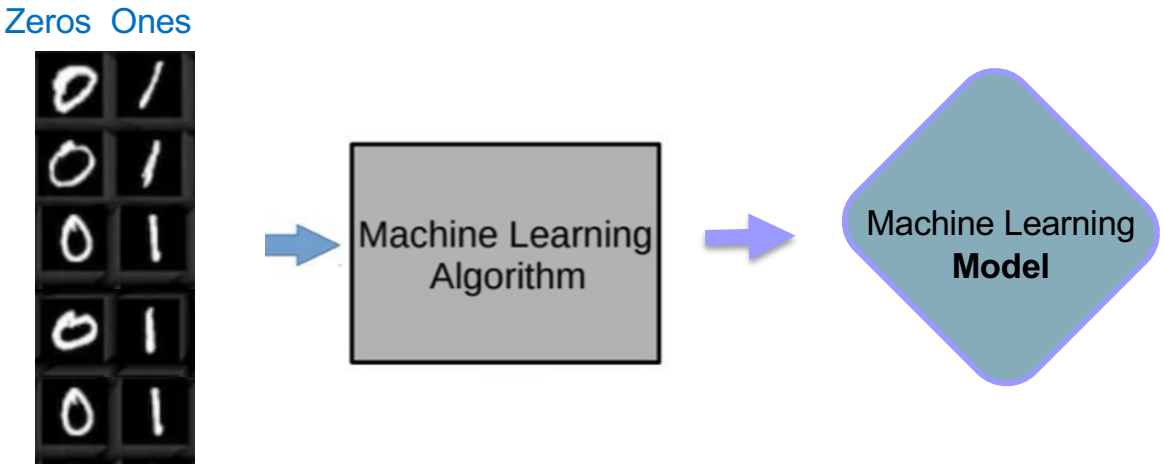
0
0
0
...
1
1
1
...
1
0
0
...

Flattened Image
(in column order)

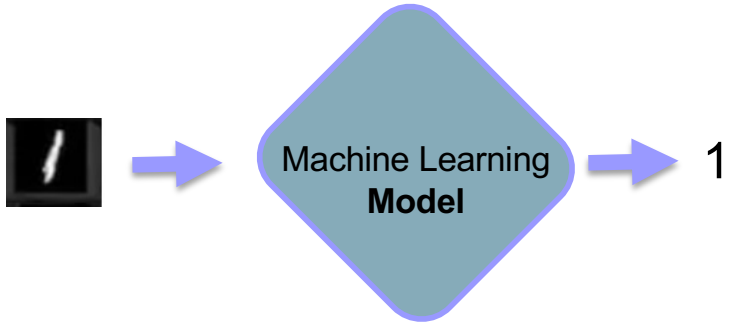
Simplified MNIST Dataset

Task: Classify a given digit image as 0 or 1

Learning: Using **labelled examples**, use a machine learning algorithm, to develop an ML model



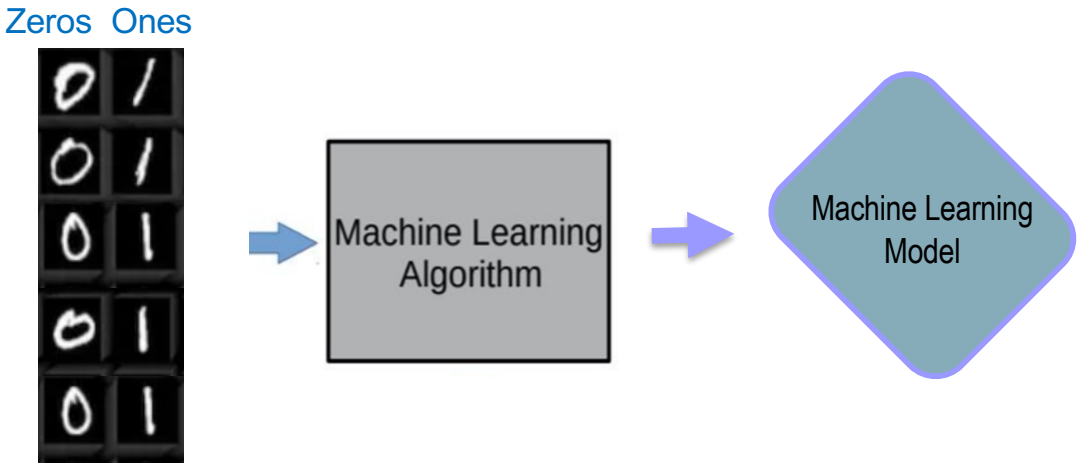
Query/Inference/Test: Given an **unlabelled example**, use the **model** to **classify** it



Simplified MNIST Dataset

Task: Classify the given digit image as 0 or 1

Learning: Considering labelled examples, use a ML algorithm, to develop an ML model



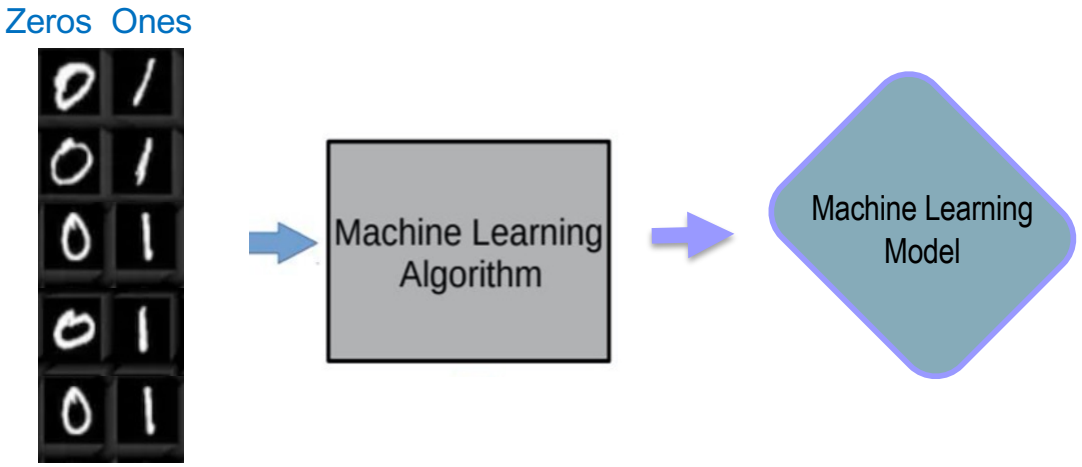
ML Model is a simple Decision Tree:

```
//If Center is Black
If sum(Pixel(54),Pixel(55), Pixel(56)) < 50
        Then y=0; //Predict zero
        Else y = 1; //Predict 1
```

Simplified MNIST Dataset

Task: Classify the given digit image as 0 or 1

Learning: Considering labelled examples, use a ML algorithm, to develop an ML model



ML Model is a simple Decision Tree:

```
//If Center is Black
If sum(Pixel(54),Pixel(55), Pixel(56)) < 50
        Then y=0; //Predict zero
        Else y = 1; //Predict 1
```

How do we know which pixels to look at?

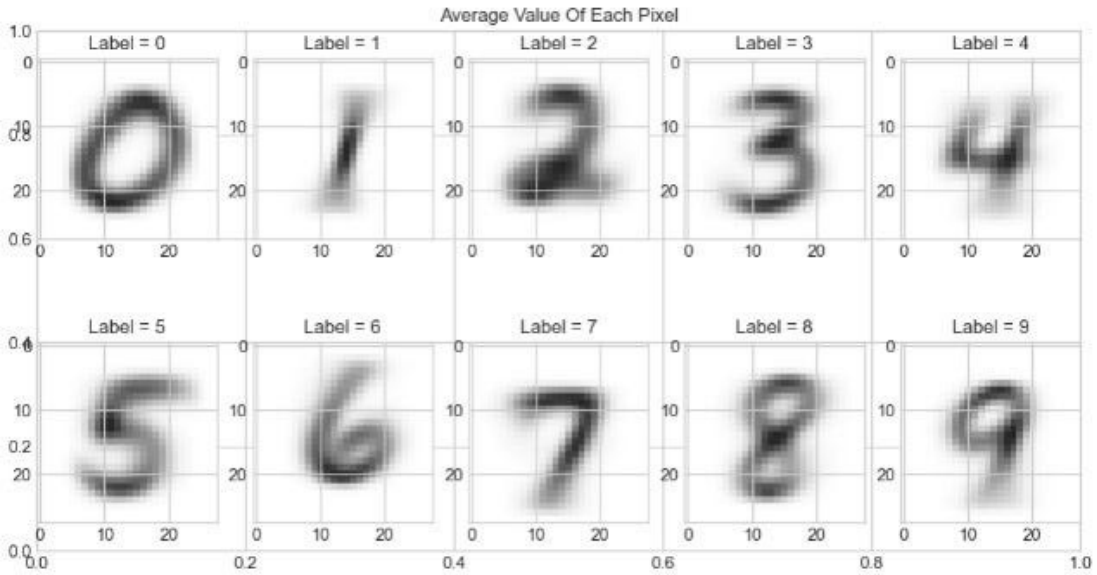
- Manuall features (mostly in the past)
- Automatically learned features

Feature Extraction – Input Representation

- We could also **extract features** from the **raw image**

  Manual feature extraction

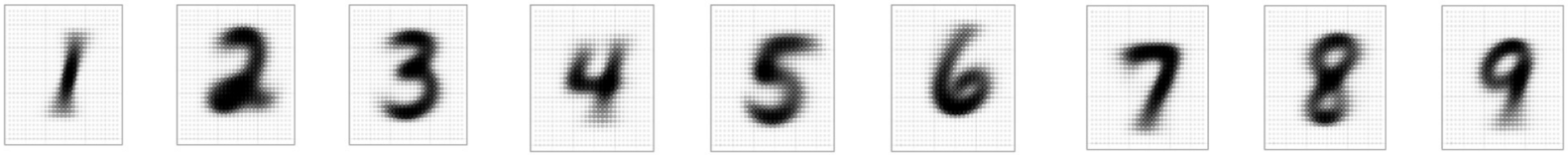
  E.g. pixel averages to gain some invariance against shifts



When we talk about **features**, it will mean whatever features are deemed useful, and thus input into the system – raw or extracted, unless specified.

Learning w. Prototypes – A ML Model for MNIST

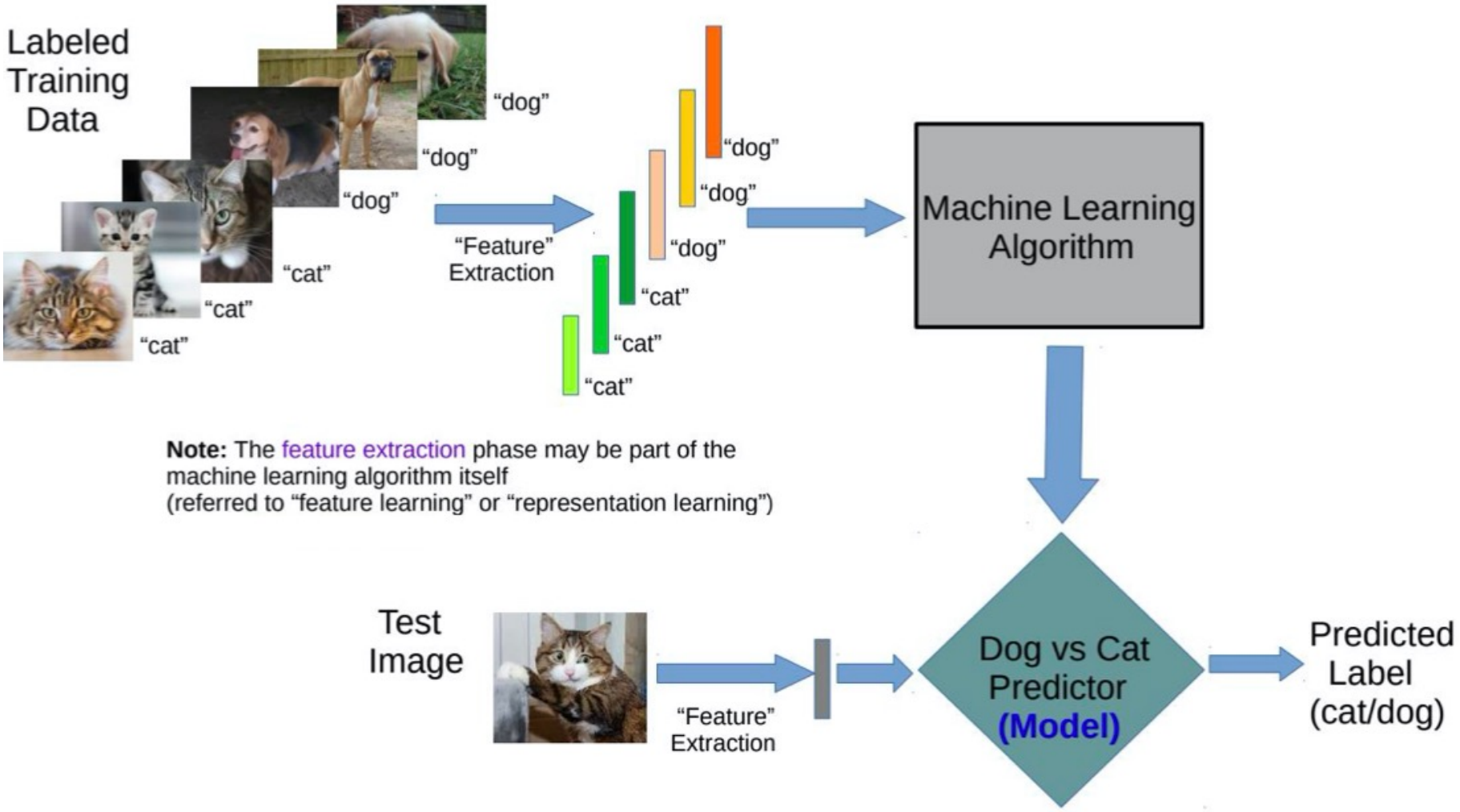
- **Basic idea:** Represent each class by a "prototype" vector
- **Class Prototype:** The "mean" or "average" of inputs from that



Averages (prototypes) of each of the handwritten digits 1-9

- **Inference**: Predict label of a test input based on its distances from the class prototypes
  - Predicted label will be the class that is the closest to the test input
- **Distance Metric:** Euclidean, weight Euclidean, Mahalanobis, …

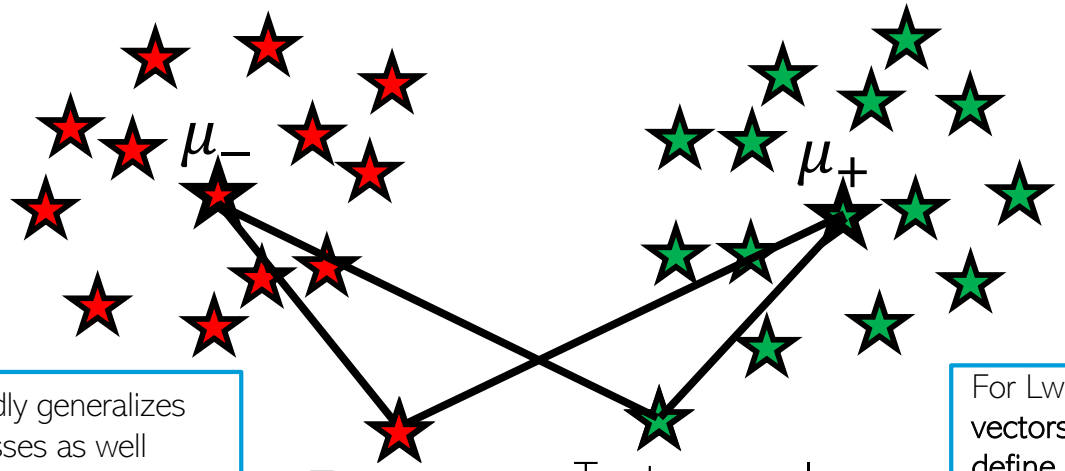
Cat vs Dog Binary Classification



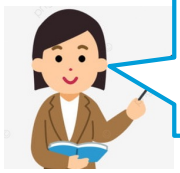
Learning with Prototypes / Template Matching

- Suppose the task is binary classification (two classes assumed pos and neg)
- Training data: N labelled examples $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^D$, $y_n \in \{-1, +1\}$
  - Assume N_+ example from positive class, N_- examples from negative class
  - Assume green is positive and red is negative

$$\mu_- = \frac{1}{N_-} \sum_{y_n=-1} \mathbf{x}_n$$

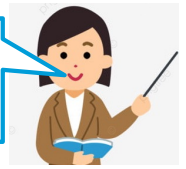


$$\mu_+ = \frac{1}{N_+} \sum_{y_n=+1} \mathbf{x}_n$$



LwP straightforwardly generalizes to more than 2 classes as well (multi-class classification) – K prototypes for K classes

For LwP, the prototype vectors ($\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$) define the "model"



Slide credit: Piyush Rai

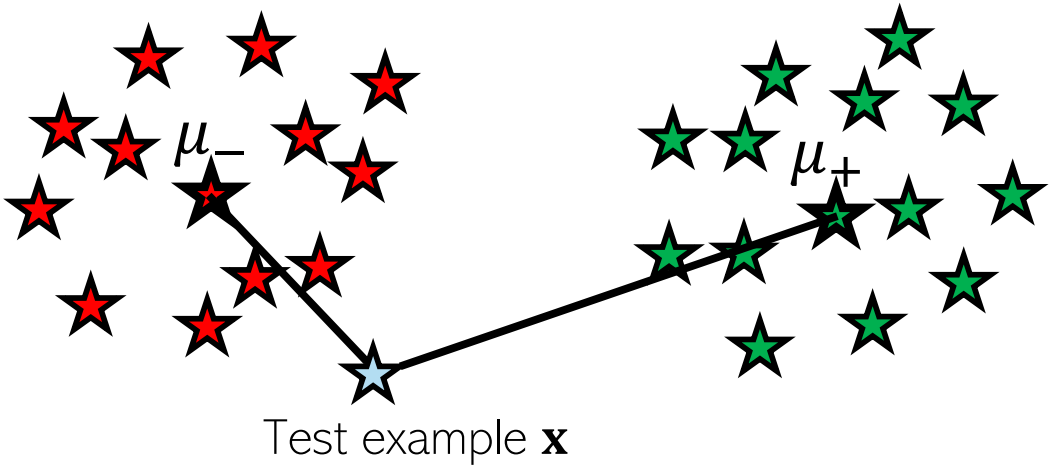
LwP: The Prediction Rule, Mathematically

- What does the prediction rule for LwP look like mathematically?

- Assume we are using Euclidean distances here

$$\left|\left|\boldsymbol{\mu}_- - \mathbf{x}\right|\right|^2 = \left|\left|\boldsymbol{\mu}_-\right|\right|^2 + \left|\left|\mathbf{x}\right|\right|^2 - 2\langle\boldsymbol{\mu}_-, \mathbf{x}\rangle$$

$$\left|\left|\boldsymbol{\mu}_+ - \mathbf{x}\right|\right|^2 = \left|\left|\boldsymbol{\mu}_+\right|\right|^2 + \left|\left|\mathbf{x}\right|\right|^2 - 2\langle\boldsymbol{\mu}_+, \mathbf{x}\rangle$$



Prediction Rule: Predict label as +1 if $\left|\left|\boldsymbol{\mu}_- - \mathbf{x}\right|\right|^2 > \left|\left|\boldsymbol{\mu}_+ - \mathbf{x}\right|\right|^2$
otherwise as -1

Template Matching Model - Key Aspects

- Very simple, interpretable, and lightweight model
  - Just requires computing and storing the class prototype vectors
- Works with any number of classes (thus for multi-class classification as well)
- Can be generalized in various ways to improve it further, e.g.,
  - Modeling each class by a probability distribution rather than just a prototype vector
  - Using distances other than the standard Euclidean distance (e.g., Mahalanobis)
- How well template matching works depends crucially on the distance metric. With a learned distance function, can work very well even with very few examples from each class (used in some "few-shot learning" models …)

Performance Evaluation

How do we evaluate a trained model?

- We are not really interested in **how well a system does on the training data** but how well it will do in **general** (for **unseen** data)

- **Training set performance is mostly irrelevant**, but:

  If too low, it tells us that our model's complexity may not be sufficient.

  if too high, it doesn't tell us that we are doing well. The model may have even memorized all the labels

Empirical Risk Minimization

Assume we have learned a function f(**x**), mapping the input **x** to a label y

Formally, we want to **minimize the true risk** (also called **loss)**:

$$R_{\text{true}}(f) = \mathop{\mathbb{E}}_{p}\big[L(f(\mathbf{x}), \mathbf{y})\big] = \int\int p(\mathbf{x}, \mathbf{y}) L(f(\mathbf{x}), \mathbf{y}) d\mathbf{x} d\mathbf{y}.$$

where *L(f(**x**),y)* is the loss incurred predicting *f(**x**)* instead of the target value y; and p(**x**,y) is the joint distribution of the input and labels.

Empirical Risk Minimization

$$R_{\text{true}}(f) = \mathop{\mathbb{E}}_{p}\big[L(f(\mathbf{x}), \mathbf{y})\big] = \int\int p(\mathbf{x}, \mathbf{y}) L(f(\mathbf{x}), \mathbf{y}) d\mathbf{x} d\mathbf{y}.$$

- Since we don't know the distribution p(x,y), we will use **empirical risk minimization.**

  We will minimize the error on an **unseen test set** hoping that it is a good enough substitute for the true risk.

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x^{(i)}), y^{(i)})$$

Loss Function / Performance Measure

Performance of the learner can be measured in one of the following ways, as suitable for the application:

Accuracy:

- Misclassification rate (in classification problems)
- Mean Squared Error (in regression problems)
- General Loss functions (taking into account different costs for different mistakes)
- Precision, Recall, Area under the ROC Curve (will see later)
- ...

Training duration

Test duration

Model size

Performance Metrics

- Misclassification rate for Classification

  # mistakes/ # test samples

$$\frac{1}{N}\sum_{i=1}^{N} I(t^i \neq y(x^i))$$

- Mean Squared Error for Regression

  Average squared difference between the predicted value and the target values (we square so that errors don't cancel each other)

$$\frac{1}{N}\sum_{i=1}^{N} (y(x_i) - t_i)^2$$

Classification

- Confusion matrix:

		Predicted class	
		Cancer	NotCancer
Actual class	Cancer	*120*	*40*
	NotCancer	*50*	*450*

- Tells us where the errors are happening
- Applicable in multi-class problems as well

- Assume the rate of cancer is 5% among all people tested at a certain hospital.
- What happens if an ML model predicts all cases as non-cancer?

		Predicted class	
		Cancer	NotCancer
Actual class	Cancer	*0*	*5*
	NotCancer	*0*	*95*

Validation and Cross-Validation

Train - Validation - Test

- If there is a need for parameter tuning (e.g. deciding how complex the decision rule/tree should be), we can use a **validation set:**

  Split the training data into two:

  - validation (hold-out set)
  - development set

  Train different models with different parameters using samples from the development set

  Measure performance them on the validation set to select the best parameters

- The test set refers to the sequestered (kept away) data that is used as the indicator of generalization performance (e.g. Kaggle).

Cross-Validation

- If training set is not very large;
  - If the development (training portion) is kept larger, the learning improves
  - If the validation part is kept larger, generalization error estimate is more reliable

- For this reason, we often use cross-validation for model selection.

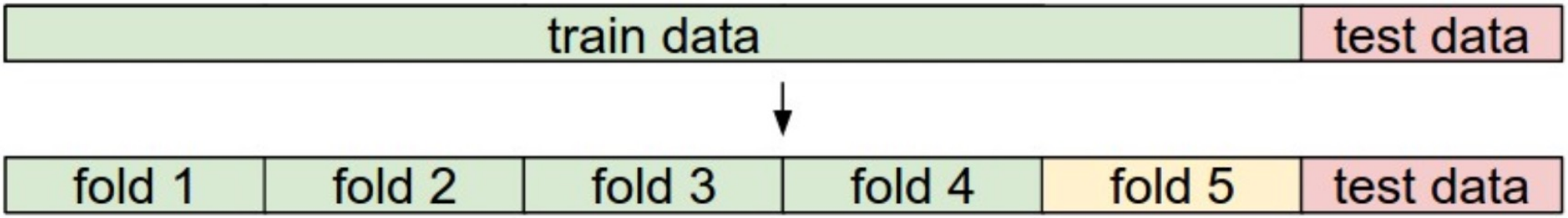
- You need to separate a subset as test, and another for validation.
  - Choose your model by testing on validation set
  - Test on test set.

k-fold cross-validation

Split the training set into k folds (random or stratified)

For each fold f
Train the learner with the other folds
Test the learner on the f fold, giving err_f

Average error = $\underset{\text{all } f}{Average}(err_f)$



Test Set

- If you have access to the test data during training, you are not supposed to use it or even peek at it during your model building**; or your test performance becomes biased!**

Stock Prediction

- Will NVIDIA's stock rise or fall tomorrow?

- It is impossible to reliably predict tomorrow`s stock price
  - Yet, investment companies do exactly that
    - We can detect patterns and predict the future probabilistically

- We need relevant factors as input
  - What if some relevant features are missing??
  - What if some irrelevant features are included??
- Yet, there could be a war tomorrow that changes everything
- Machine learning aims to detect historical patterns and behaviour so as to predict the future
  - Better than random? better than others? …

!What to Know!

- Classification vs Regression
- Features vs Label
- Learning, inference…
- Machine learning algorithm vs ML model
- Raw input vs feature representation
- The MNIST and toy-MNIST dataset/problem
- Supervised learning terminology: $\{(\mathbf{x}_i, y_i)\}$
- Basic performance metrics, true vs empirical risk, misclassification rate, MSE, confusion matrix
- Prototype learning

  >> Later: k-nearest neighbor model
- Correct use of validation, development and test sets