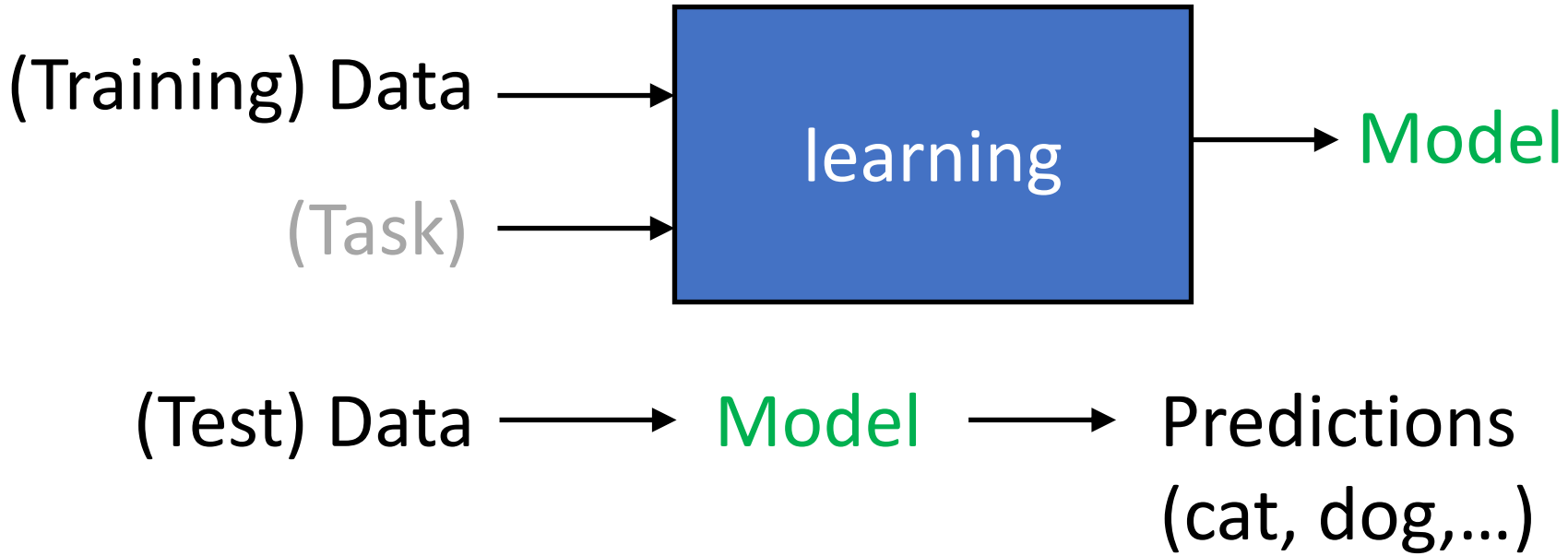
When Models Meet Data

Liang Zheng
Australian National University
liang.zheng@anu.edu.au

8.1 Data, Models, and Learning

- A machine learning system has three major components:
- Data, models, learning

- A model is obtained by learning from the training data
- A prediction is made by applying a learned model on test data



8.1 Data, Models, and Learning

- We aim to learn good models.
- How is good defined? We need to have performance metrics on the test data. Examples include
- Classification accuracy
- Distance from the ground truth
- Test time (efficiency)
- Model size
- ……
- New performance metrics are constantly being proposed by the machine learning community.

8.1.1 Data as Vectors

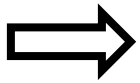
- Data, read by computers, should be in a numerical format.
- See the tabular format below

Name	Gender	Degree	Postcode	Age	Annual salary
Aditya	M	MSc	W21BG	36	89563
Bob	M	PhD	EC1A1BA	47	123543
Chloé	F	BEcon	SW1A1BH	26	23989
Daisuke	M	BSc	SE207AT	68	138769
Elisabeth	F	MBA	SE10AA	33	113888

- Row: an instance
- Column: a particular feature
- Apart from tabular format, machine learning can be applied to many types of data, e.g., genomic sequences, text and image contents of a webpage, and social media graphs, citation networks...

- We convert the table into numerical format

Name	Gender	Degree	Postcode	Age	Annual salary
Aditya	M	MSc	W21BG	36	89563
Bob	M	PhD	EC1A1BA	47	123543
Chloé	F	BEcon	SW1A1BH	26	23989
Daisuke	M	BSc	SE207AT	68	138769
Elisabeth	F	MBA	SE10AA	33	113888



Gender ID	Degree	Latitude (in degrees)	Longitude (in degrees)	Age	Annual Salary (in thousands)
-1	2	51.5073	0.1290	36	89.563
-1	3	51.5074	0.1275	47	123.543
+1	1	51.5071	0.1278	26	23.989
-1	1	51.5075	0.1281	68	138.769
+1	2	51.5074	0.1278	33	113.888

- Gender is quantized to -1 and +1
- Degree from BS, MS to PhD: 1, 2, 3
- Postcode corresponds to Latitude and Longitude on the map
- Name is removed because of privacy and because it does not contain useful information for the machine learning system. (exceptions? See [1])

[1] Chen et al., What's in a Name? First Names as Facial Attributes. CVPR 2013

- We use N to denote the number of examples in a dataset and index the examples with lowercase $n = 1, \cdots, N$

Gender ID	Degree	Latitude (in degrees)	Longitude (in degrees)	Age	Annual Salary (in thousands)
-1	2	51.5073	0.1290	36	89.563
-1	3	51.5074	0.1275	47	123.543
+1	1	51.5071	0.1278	26	23.989
-1	1	51.5075	0.1281	68	138.769
+1	2	51.5074	0.1278	33	113.888

- Each row is a particular individual $\boldsymbol{x}_n$ referred to as an example or data point in machine learning
- The subscript n refers to the fact that this is the nth example out of a total of N examples in the dataset
- Each column represents a particular feature of interest about the example, and we index the features as $d = 1, \cdots, D$
- Each example is a D-dimensional vector

- Consider the problem of predicting annual salary from age

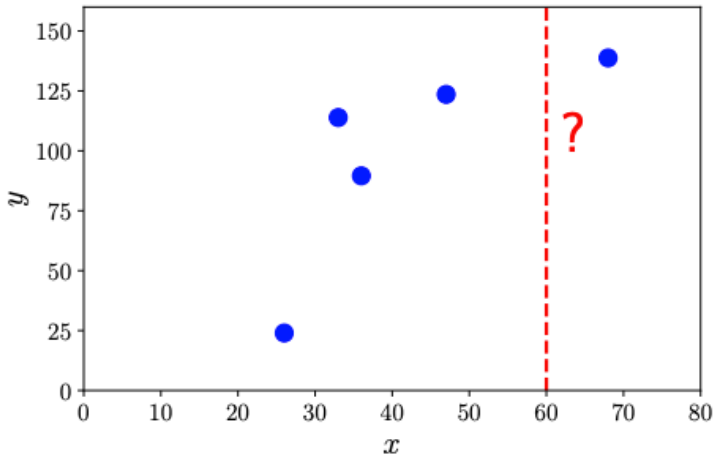
D columns

Gender ID	Degree	Latitude (in degrees)	Longitude (in degrees)	Age	Annual Salary (in thousands)
-1	2	51.5073	0.1290	36	89.563
-1	3	51.5074	0.1275	47	123.543
+1	1	51.5071	0.1278	26	23.989
-1	1	51.5075	0.1281	68	138.769
+1	2	51.5074	0.1278	33	113.888

N rows

- A supervised learning algorithm
- We have a label y_n (the salary) associated with each example $\boldsymbol{x}_n$ (age).
- A dataset is written as a set of example-label pairs $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n), \ldots, (\boldsymbol{x}_N, y_N)\}$
- The table of examples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ are concatenated and written as $X \in \mathbb{R}^{N \times D}$

We are interested in:
What is the salary (y) at age 60 ($x = 60$)?



x: age
y: salary

8.1.2 Models as Functions

- Once we have data in an appropriate vector representation, we can construct a predictive function (known as a predictor).
- Here, a model means a predictor.
- A predictor is a function that, when given a particular input example (in our case, a vector of features), produces an output.

- For example,

$$f: \mathbb{R}^D \rightarrow \mathbb{R}$$

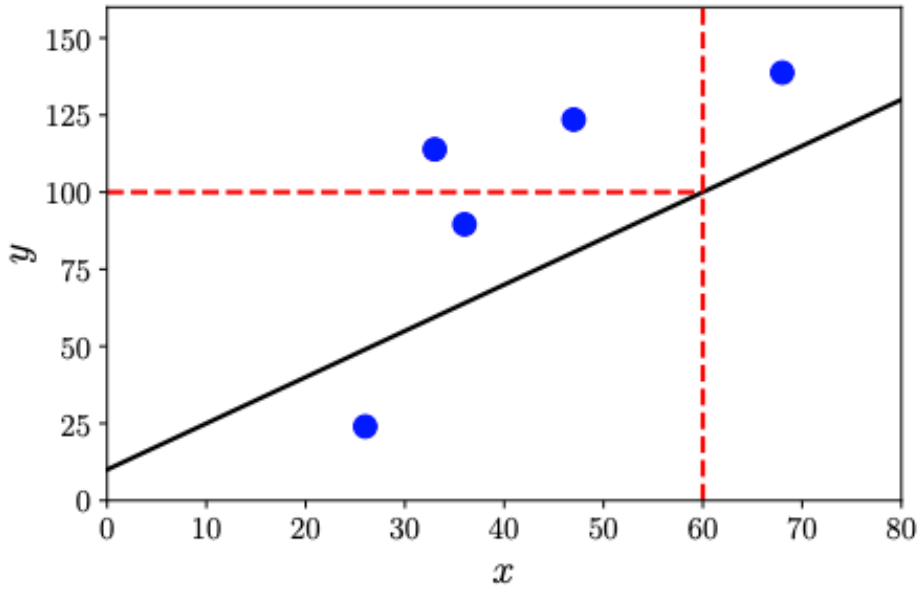
where the input $\boldsymbol{x}$ is a D-dimensional vector, and the output is a real-valued scalar. That is, the function f is applied to $\boldsymbol{x}$, written as $f(\boldsymbol{x})$ and returns a real number.

8.1.2 Models as Functions

- We mainly consider the special case of linear functions

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + \theta_0$$

- Example: predicting salary $f(\boldsymbol{x})$ from age $\boldsymbol{x}$.

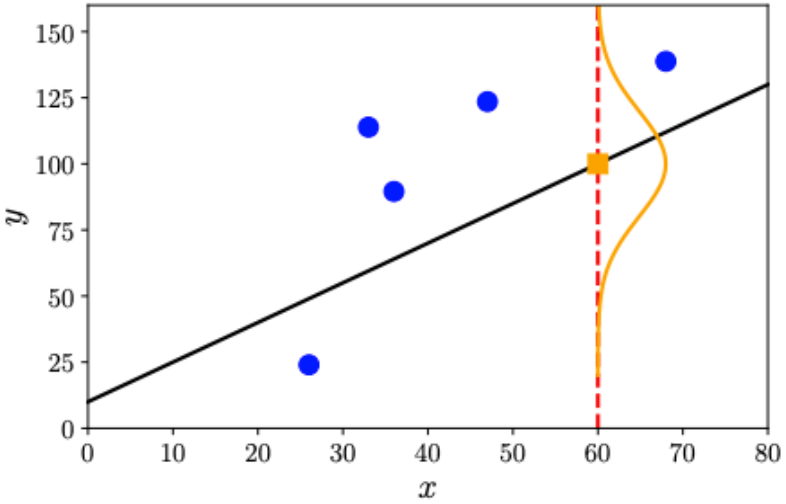


Black and solid diagonal line is an example predictor.

$$f(60) = 100$$

8.1.3 Models as Probability Distributions

- The observed data is usually a combination of the true underlying data and noise, i.e., $\tilde{x} = x + n$
- We wish to reveal x from $\tilde{x}$
- So We would like to have predictors that express some sort of uncertainty, e.g., to quantify the confidence we have about the value of the prediction for a particular test data point.



Example function (black solid diagonal line) and its predictive uncertainty at $x = 60$ (drawn as a Gaussian).

- Instead of considering a predictor as a single function, we could consider predictors to be probabilistic models.
- We will learn probability in later lectures

8.1.4 Learning is Finding Parameters

- The goal of learning is to find a model and its corresponding parameters such that the resulting predictor will perform well on unseen data.
- 3 algorithmic phases when discussing machine learning algorithms
- Prediction or inference
- Training or parameter estimation
- Hyperparameter tuning or model selection

- Prediction phase: we use a trained predictor on previously unseen test data
- The training or parameter estimation phase: we adjust our predictive model based on training data. We will introduce the empirical risk minimization for finding good parameters.
- We use cross-validation to assess predictor performance on unseen data.
- We also need to balance between fitting well on training data and finding "simple" explanations of the phenomenon. This trade-off is often achieved using regularization.

- Hyperparameter tuning or model selection
- We need to make high-level modeling decisions about the structure of the predictor. For example
- Number of layers to be used in deep learning
- Number of components in a Gaussian Mixture Model
- Weight of regularization terms

} Hyperparameter

- The problem of choosing among different models/hyperparameters is called model selection

- Difference between parameters and hyperparameters
- Parameters are to be numerically optimized ($\sim 10^6$ weights in a deep network)
- Hyperparameters need to use search techniques (neural architecture search [2])

[2] Zoph et al., Neural architecture search with reinforcement learning, Arxiv 2016

8.2 Empirical Risk Minimization

- What does it mean to learn?
- Estimating parameters based on training data.
- Four questions will be answered
- What is the set of functions we allow the predictor to take? – Hypothesis class of functions
- How do we measure how well the predictor performs on the training data? -- Loss functions for training
- How do we construct predictors from only training data that performs well on unseen test data? -- regularization
- What is the procedure for searching over the space of models? -- Cross-Validation

8.2.1 Hypothesis Class of Functions

- We are given N examples $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding scalar labels $y_n \in \mathbb{R}$.
- Supervised learning: we have pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
- We want to estimate a predictor $f(\cdot, \boldsymbol{\theta}): \mathbb{R}^D \to \mathbb{R}$, parametrized by $\boldsymbol{\theta}$
- We hope to be able to find a good parameter $\boldsymbol{\theta}^*$ such that we fit the data well, that is

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}^*) \approx y_n \text{ for all } n = 1, \ldots, N$$

- We use $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta}^*)$ to represent the output of the predictor

Example (least-squares regression)

- When the label y_n is real-valued, a popular choice of function class for predictors is affine functions (linear functions).

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + \theta_0$$

- For more compact representations, we concatenate an additional unit feature $x^{(0)} = 1$ to $\boldsymbol{x}_n$, i.e.,

$$\boldsymbol{x}_n = \left[x_n^{(0)}, x_n^{(1)}, x_n^{(2)}, \dots, x_n^{(D)}\right]^{\mathrm{T}} = \left[1, x_n^{(1)}, x_n^{(2)}, \dots, x_n^{(D)}\right]^{\mathrm{T}}$$

- The parameter vector is $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \dots, \theta_D]^T$
- We can write the predictor as follows

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n$$

which is equivalent to the affine model

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \theta_0 + \sum_{d=1}^{D} \theta_d x_n^{(d)} = \theta_0 x_n^{(0)} + \sum_{d=1}^{D} \theta_d x_n^{(d)} = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n$$

Example (least-squares regression)

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n$$

- The predictor takes the vector of features representing a single example $\boldsymbol{x}_n$ as input and produces a real-valued output,

$$f\colon \mathbb{R}^{D+1} \to \mathbb{R}$$

- $f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n$ is a linear predictor
- There are many non-linear predictors, such as the neural networks

8.2.2 Loss Function for Training

- In training, we aim to learn a model that fits the data well.
- To define "fits the data well", we specify a loss function

$$\ell(y_n, \hat{y}_n)$$

- Input: ground truth label y_n of a training example

  the prediction $\hat{y}_n$ of this training example
- Output: a non-negative number, called loss. It represents how much error we have made on this particular prediction
- To find good parameters $\boldsymbol{\theta}^*$, we need to minimize the average loss on the set of N training examples

- We usually assume training examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ are independent and identically distributed (i.i.d).

- Under the i.i.d assumption, the empirical mean is a good estimate of the population mean.
- We can use the empirical mean of the loss on the training data
- Given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, we use the notation of an example matrix

$$\boldsymbol{X} \coloneqq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times D}$$

and a label vector

$$\boldsymbol{y} = [y_1, \ldots, y_N]^{\mathrm{T}} \in \mathbb{R}^N$$

- The average loss is given by

$$\mathbf{R}_{emp}(f, \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n)$$

where $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta})$. The above equation is called the empirical risk. The learning strategy is called empirical risk minimization.

Example - Least-Squares Loss

- We use the squared loss function

$$\ell(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2$$

- We aim to minimize the empirical risk, which is the average of the losses over the training data.

$$\min_{\theta \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n) = \min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} \left(y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta})\right)^2$$

Using the linear predictor $f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n$, we obtain the optimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} \left(y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta})\right)^2$$

- This equation can be equivalently expressed in matrix form

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2$$

- This is known as the least-squares problem. There exists a closed-form analytic solution for this by solving the normal equations. We will discuss it in later lectures

- We actually want to find a predictor f that minimizes the expected risk (or the population risk)

$$\mathbf{R}_{\text{true}}(f) = \mathbb{E}_{x,y}\left[\ell\left(y, f(\boldsymbol{x})\right)\right]$$

where y is the ground truth label and $f(\boldsymbol{x})$ is the prediction based on the example $\boldsymbol{x}$.

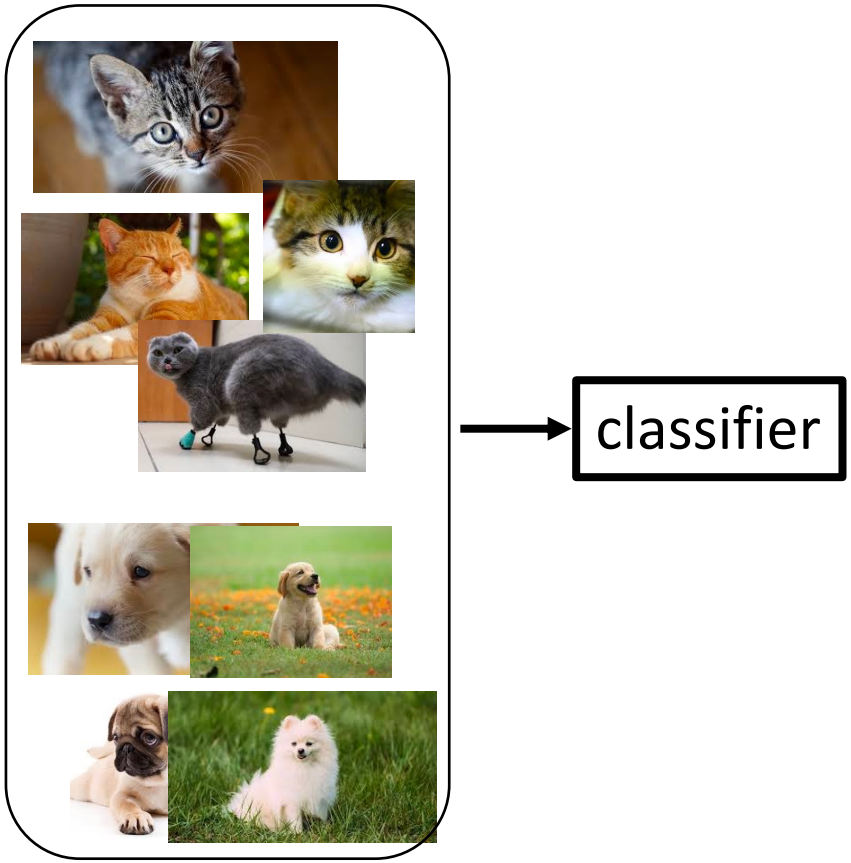
- $\mathbf{R}_{\text{true}}(f)$ is the true risk, if we can access an infinite amount of data
- The expectation $\mathbb{E}$ is over the infinite set of all possible data and labels.

- Machine learning applications have different types of performance measure.
  - For classification: accuracy, AUC, F1 score, etc.
  - For detection: mean average precision, mIoU, etc.
  - For image denoise/super resolution: SSIM, PSNR, etc.
- In principle, the loss function should correspond to the measure.
- However, there are often mismatches between loss functions and the measures – due to implementation/optimization considerations

Check your understanding

- A machine learning model may contain as few as a couple of parameters
- When we use a linear regression modeling, $f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + \theta_0$, we don't have hyperparameters.
- Hyperparameters are usually learned through the same way as normal parameters.
- It's very hard to know the expected risk, but easier to know the empirical risk
- Given a fixed task, we can only use a fixed set of evaluation metrics.

We start with training a classifier



Training data

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

We do a bit testing….

Correct prediction

Testing image → classifier → Dog 0.05, Cat 0.95 → Cat

Prediction result = Ground truth

Wrong prediction

Testing image → classifier → Dog 0.85, Cat 0.15 → Cat

Prediction result ≠ Ground truth

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

We now evaluate a model

Training data

classifier

Testing data

Evaluation

Accuracy:
90%

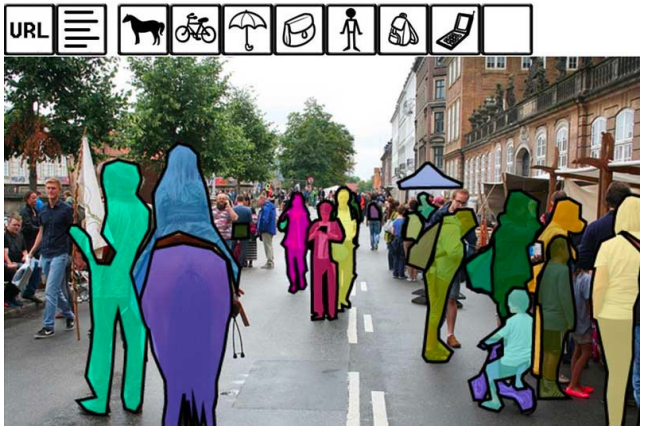
Ground truths provided

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

Is this way of evaluation feasible?

- Yes

ImageNet



MSCOCO

Ground truths provided

LFW

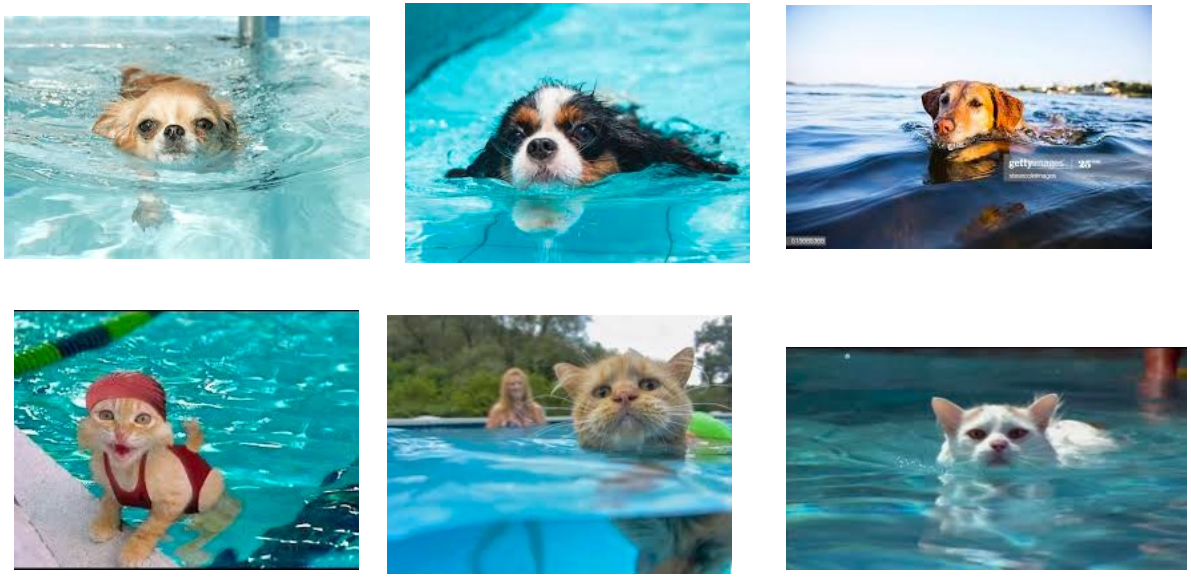
Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

Is this way of evaluation feasible?

- No….

We can’t calculate a classifier accuracy!!

Suppose we deploy our cat-dog classifier to a swimming pool



Ground truths not provided

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

We encounter this problem too many times in CV applications….

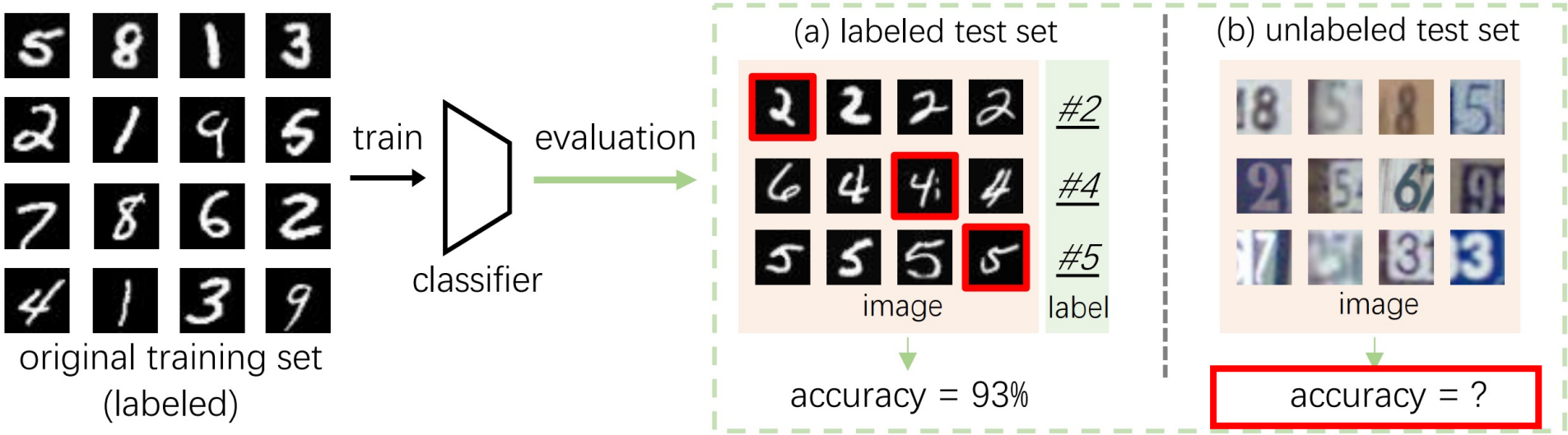
- Deploy a ReID model to a new community
- Deploy face recognition in an airport
- Deploy a 3D object detection system to a new city
- ……

We can't quantitatively measure the performance of our model like we usually do!!

Unless we annotate the test data…, but environment will change over time…. We need to annotate test data again

Deng and Zheng, "Are labels always necessary for classifier accuracy evaluation?" CVPR 2021.

Formally:

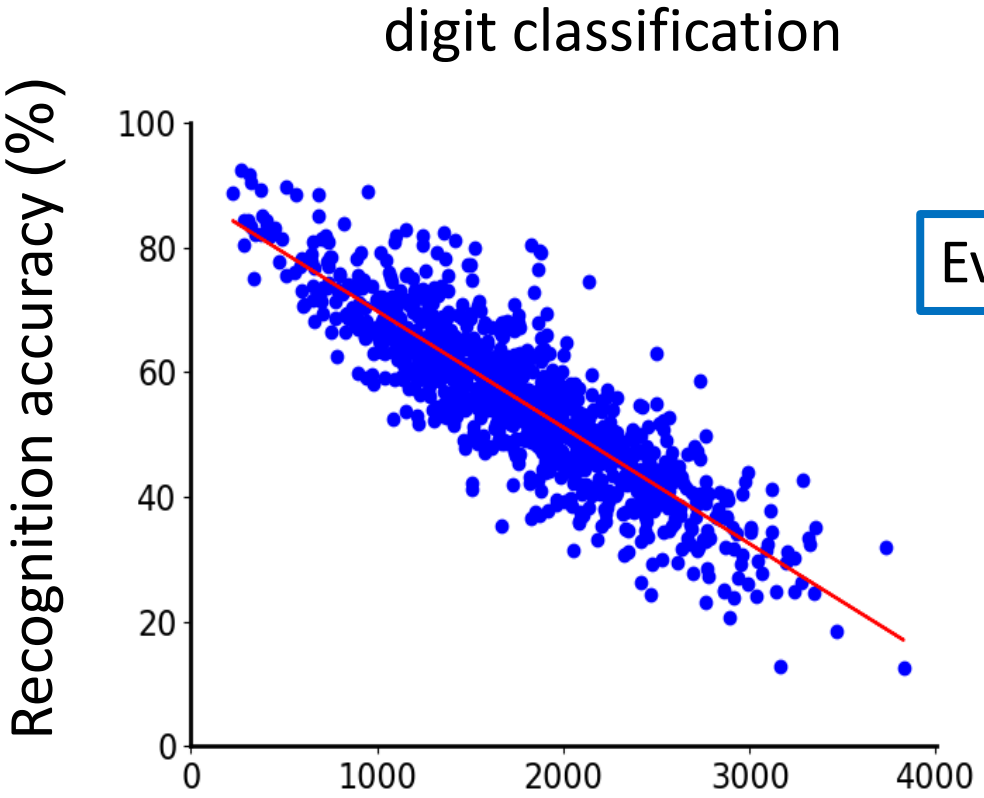


Given
- A training dataset
- A classifier trained on this dataset
- A test set without labels

We want to estimate:
Classification accuracy on the test set

Deng and Zheng, "Are labels always necessary for classifier accuracy evaluation?" CVPR 2021.

Method - regression



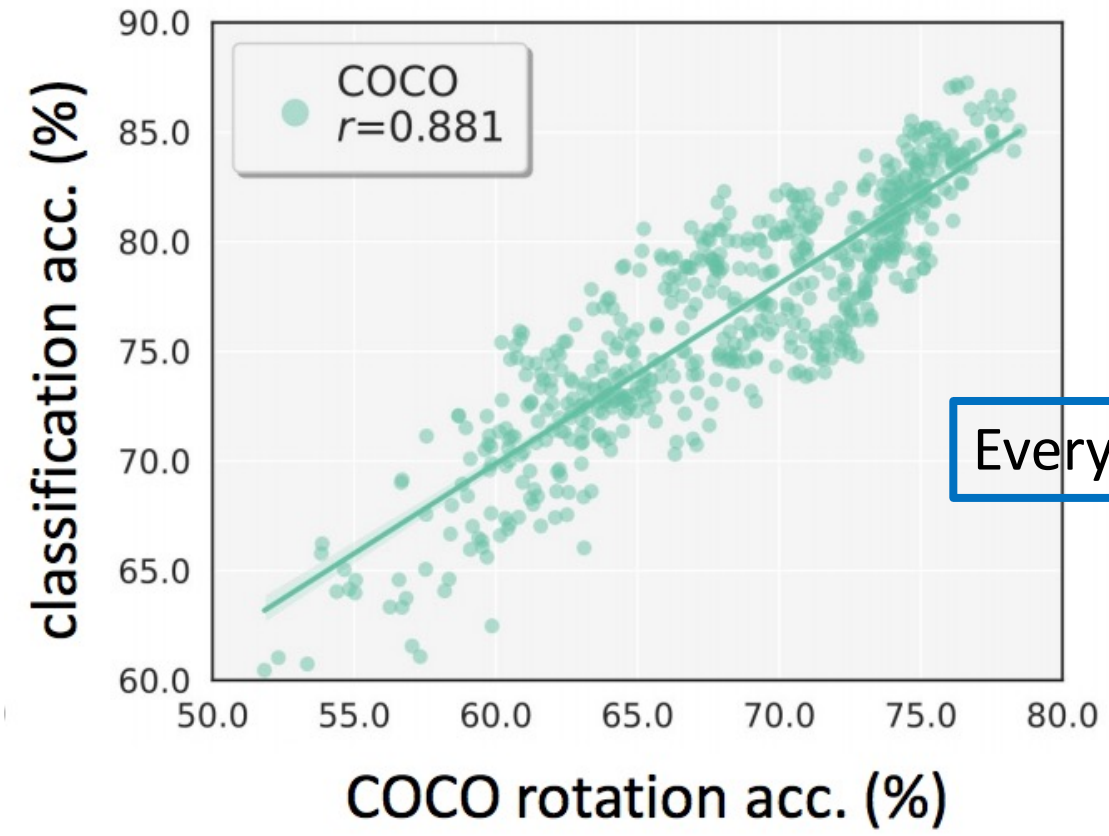
Every point is a dataset

Fréchet distance

Domain gap between a training set and test sets

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

Method - regression



Every point is a dataset

Rotation prediction accuracy

Deng, Gould and Zheng, "What does rotation prediction tell us about classifier accuracy under varying testing environments?" ICML 2021

Experiment

	Digits			Natural images			
Method	SVHN	USPS	MSE↓	Pascal	Caltech	ImageNet	MSE↓
Ground-truth accuracy	25.46	64.08	0	86.13	93.40	88.83	0

Deng and Zheng, “Are labels always necessary for classifier accuracy evaluation?” CVPR 2021.

Experiment

Method	Digits			Natural images			
	SVHN	USPS	MSE↓	Pascal	Caltech	ImageNet	MSE↓
Ground-truth accuracy	25.46	64.08	0	86.13	93.40	88.83	0
Confidence ($\tau = 0.8$)	7.97	5.88	16.03	84.32	90.78	86.50	**1.32**
Confidence ($\tau = 0.9$)	37.22	27.95	20.55	78.61	87.71	87.71	4.02

"Confidence": a simple pseudo label method.

If the maximum value of the softmax vector is greater than τ, we view this sample as correctly classified.

Deng and Zheng, "Are labels always necessary for classifier accuracy evaluation?" CVPR 2021.

Experiment

Method	Digits			Natural images			
	SVHN	USPS	MSE↓	Pascal	Caltech	ImageNet	MSE↓
Ground-truth accuracy	25.46	64.08	0	86.13	93.40	88.83	0
Confidence ($\tau = 0.8$)	7.97	5.88	16.03	84.32	90.78	86.50	**1.32**
Confidence ($\tau = 0.9$)	37.22	27.95	20.55	78.61	87.71	87.71	4.02
Linear reg.	26.28	50.14	6.98	83.87	79.11	83.19	4.98

Deng and Zheng, "Are labels always necessary for classifier accuracy evaluation?" CVPR 2021.

Experiment

Method	Digits			Natural images			
	SVHN	USPS	MSE↓	Pascal	Caltech	ImageNet	MSE↓
Ground-truth accuracy	25.46	64.08	0	86.13	93.40	88.83	0
Confidence ($\tau = 0.8$)	7.97	5.88	16.03	84.32	90.78	86.50	**1.32**
Confidence ($\tau = 0.9$)	37.22	27.95	20.55	78.61	87.71	87.71	4.02
Linear reg.	26.28	50.14	6.98	83.87	79.11	83.19	4.98
Neural network reg.	27.52	64.11	**1.03**	87.76	89.39	91.82	**1.75**

The two regression methods are stable and quite accurate.

Deng and Zheng, "Are labels always necessary for classifier accuracy evaluation?" CVPR 2021.