Technische Hochschule Rosenheim

Introduction to AI - Part II

Machine Learning in a Nutshell

Johannes Jurgovsky

12.03.2023

Overview

Technische Hochschule Rosenheim

Input x

Can be an arbitrary numerical object (e.g. image, text, sensor readings).

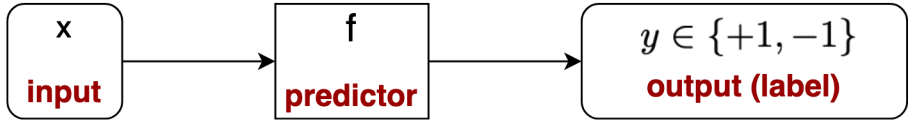
Output y

Restricted to a particular value range. Depending on the prediction task performed on x.

Predictor f

- The predictor is a function, mapping input x to output y.
- It can be a linear model, decision tree, neural network, etc.
- The predictor is a model that represents the relationship between x and y.
- The predictor is parameterized (incomplete model) and needs to be fitted (*trained*) on data to capture the desired relationship between x and y.
- Executing the predictor on an input is called inference and is usually fast compared to training it.

Supervised Learning
Binary classification



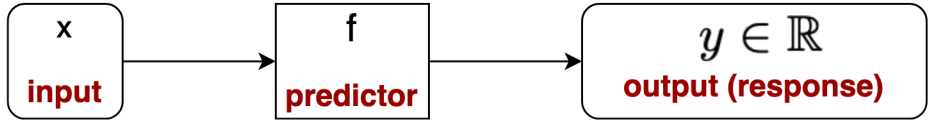
In this setting, the predictor is called a *classifier* and y is called the *label*.

Examples of classification problems:

- Fraud detection: Credit card transaction $\longrightarrow$ fraud | genuine

- Sentiment polarity: Product review $\longrightarrow$ positive sentiment | negative sentiment

Supervised Learning
Regression



In this setting, the predictor is called the *regression function* and y is called the response.

Examples of classification problems:

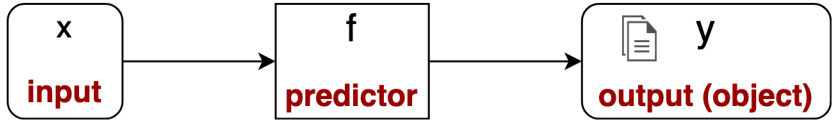
- Housing prices: Size, age, number of bedrooms, etc. $\rightarrow$ selling price
- Arrival times: Destination, vehicle, dayOfWeek $\rightarrow$ time of arrival
- Marketing: Product, typeOfAdvertisement, priceReduction $\rightarrow$ revenue

Difference between Classification and Regression:

- Classification: y is discrete
- Regression: y is continuous

Supervised Learning
Structured Prediction

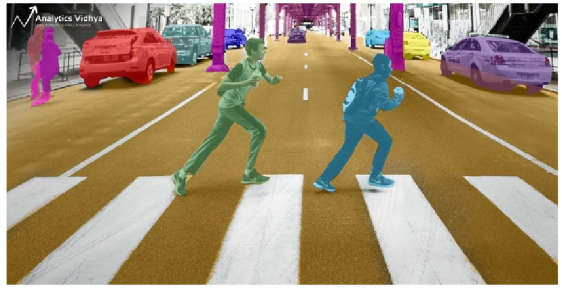
Technische Hochschule Rosenheim



In structured prediction, the output y can be any complex object.

Examples of structured prediction problems:

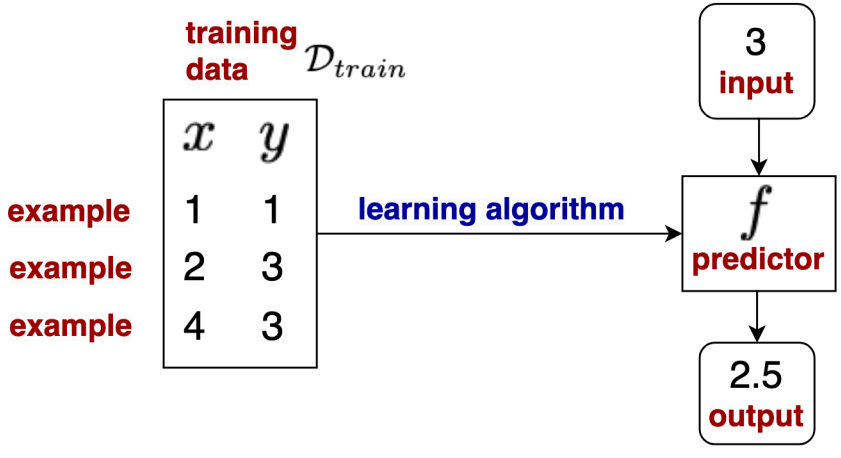
- Image captioning: Image $\longrightarrow$ textual description of scene
- Machine translation: English sentence $\longrightarrow$ German sentence
- Image segmentation: Image $\longrightarrow$ Image containing labeled pixels

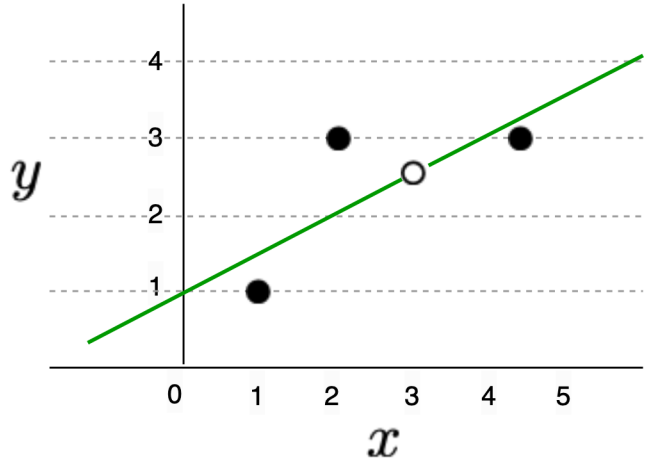


Most structured prediction tasks can be decomposed into a sequence of binary or multi-class prediction tasks.

Machine Learning: Linear Regression

Design decisions





Design decisions:

- Which predictors do we consider? → *Hypothesis class*
- How good is a predictor? → *Loss function*
- How do we find the best predictor among the ones considered? → *Optimization algorithm*

Machine Learning: Linear Regression

Hypothesis class

Technische Hochschule Rosenheim

Which predictors?

$$f(x) = 2 + 1.5 \cdot x$$
$$f(x) = 1 + 0.5 \cdot x$$
$$f(x) = w_1 + w_2 \cdot x$$

Vector notation:

- The parameters of a predictor are often called *weights*. All weights collected into a vector are called a *weight vector*: $\mathbf{w} = [w_1, w_2]$
- A *feature extractor* ϕ transforms an input x into a *feature vector*: $\phi(x) = [1, x]$
- Evaluating the predictor on an input x yields a *score*

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$
$$f_{\mathbf{w}}(3) = [1, 0.5] \cdot [1, 3] = 2.5$$

Hypothesis class:

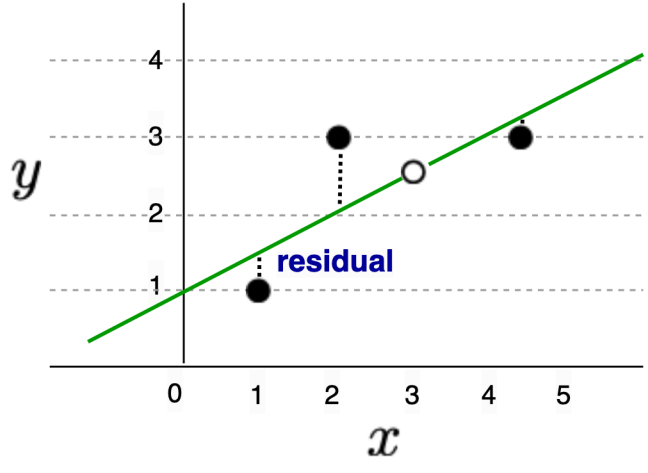
$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^2\}$$

Machine Learning: Linear Regression

Loss function

Technische Hochschule Rosenheim

How good is a predictor?



We quantify the discrepancy between the predicted value $f_{\mathbf{w}}(x)$ and the corresponding target y using a loss function. In this example, we use the **Squared Loss**:

$$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$$

The loss over all examples in the training data is called the *training loss*. It is the mean loss over all training examples:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{train}} \text{Loss}(x, y, \mathbf{w})$$

Machine Learning: Linear Regression

Optimization algorithm

Technische Hochschule Rosenheim

Notice that the training loss is a function of the weight vector: $\mathrm{TrainLoss}(\mathbf{w})$.

For each choice of $\mathbf{w}$, we can evaluate this function to obtain the training loss.

Since the loss measures the sum of squared residuals, we aim to choose $\mathbf{w}$ such that $\mathrm{TrainLoss}(\mathbf{w})$ gets as small as possible.

Goal:

$$\min_{\mathbf{w}} \mathrm{TrainLoss}(\mathbf{w})$$

Gradient: The gradient $\nabla_{\mathbf{w}} TrainLoss(\mathbf{w})$ is the vector of partial derivatives, pointing in the direction in which the loss function increases the most.

Gradient descent algorithm: Start with an initial set of weights and update the weight vector repeatedly in the direction of the negative gradient (scaled by a *learning rate*):

```
Initialize w = [0, ..., 0]
For t = 1, ..., T  (epochs)
    w ← w - η · ∇TrainLoss(w)
```

Machine Learning: Linear Regression

Gradient of Squared Loss

Technische Hochschule Rosenheim

Objective function:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{train}} (\mathbf{w} \cdot \phi(x) - y)^2$$

The **gradient** of TrainLoss is the vector of partial derivatives with respect to the individual weights. After applying the chain rule, we get:

$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{train}} 2(\mathbf{w} \cdot \phi(x) - y) \cdot \phi(x)$$

If the prediction $\mathbf{w} \cdot \phi(x)$ is equal to the target y, the gradient is Zero. If the prediction is not equal to the target, the gradient will point in a direction in which the residual would increase. Thus, updating the weight vector in the opposite direction of the gradient drives the predictions closer to the target values.

Machine Learning: Linear Regression

Gradient descent applied to example data

Technische Hochschule Rosenheim

Training data:

x	y
1	1
2	3
4	3

$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{train}} 2(\mathbf{w} \cdot \phi(x) - y) \cdot \phi(x)$$

Update rule: $\mathbf{w} \leftarrow \mathbf{w} - 0.1 \cdot \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

t	$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$	**new weight vector $\mathbf{w}$**
0		$[0, 0]$
1	$\frac{1}{3}\{2(-1) \cdot [1,1] + 2(-3) \cdot [1,2] + 2(-3) \cdot [1,4]\} = [-4.67, -12.67]$	$[0.47, 1.27]$
2	$\frac{1}{3}\{2(0.74) \cdot [1,1] + 2(0.01) \cdot [1,2] + 2(2.55) \cdot [1,4]\} = [2.18, 7.24]$	$[0.25, 0.54]$
...		
T		$[1, 0.57]$

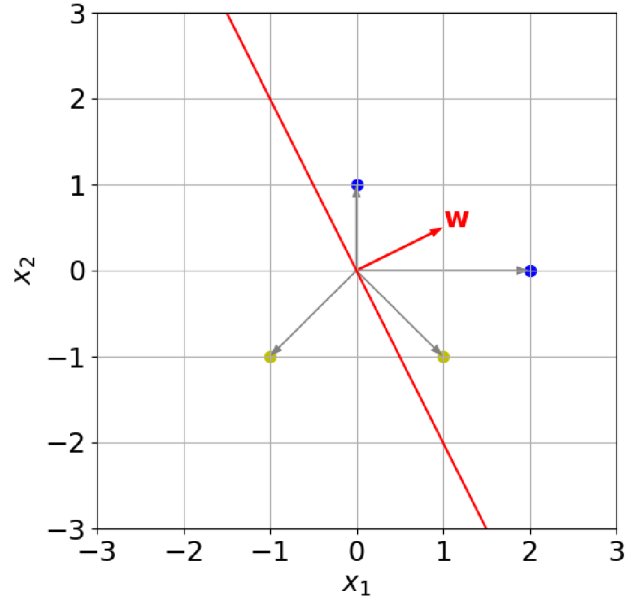
Machine Learning: Linear Classification

Decision boundary

Technische Hochschule Rosenheim

Overview

- In binary classification, the output is a label represented by a discrete variable y with values in $\{-1, 1\}$.
- The classifier is represented geometrically in *feature space* as a *decision boundary*.



Training data

x_1	x_2	y
0	1	1
2	0	1
-1	-1	-1
1	-1	-1

Design decisions

- Which classifiers are considered? $\longrightarrow$ Hypothesis class
- Given a classifier, how well does it perform? $\longrightarrow$ Loss function
- How do we find the best classifier? $\longrightarrow$ Optimization algorithm

Machine Learning: Linear Classification

Decision Boundary

Technische Hochschule Rosenheim

Training data

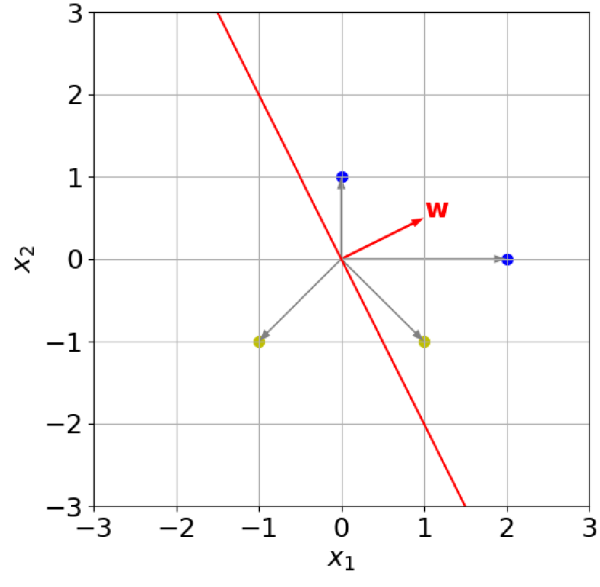
x_1	x_2	y
0	1	1
2	0	1
-1	-1	-1
1	-1	-1

Consider the example predictor

$$f_{\mathbf{w}}(x) = \text{sign}(\underbrace{[1, 0.5]}_{\mathbf{w}} \cdot \underbrace{[x_1, x_2]}_{\phi(x)})$$

The sign-function maps from a score to a label

$$\text{sign}(z) = \begin{cases} +1, & \text{if } z > 0 \\ -1, & \text{if } z < 0 \\ 0, & \text{if } z = 0. \end{cases}$$



✎ Inner product

The inner product between two vectors is proportional to the cosine of the angle between them.

- If the angle is acute, the inner product is positive.
- If the angle is obtuse, the inner product is negative.

⊙ Decision Boundary

The decision boundary consists of all points x such that $\mathbf{w} \cdot \phi(x) = 0$.

Machine Learning: Linear Classification

Hypothesis class

Technische Hochschule Rosenheim

General binary classifier

$$f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$$

Hypothesis class

$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^d\}$$

Machine Learning: Linear Classification

Loss function: Zero-One loss

Technische Hochschule Rosenheim

In contrast to loss functions in regression, a loss function for a classifier must quantify the effect of incorrect decisions.

A simple loss function is called the *Zero-One Loss*:

- If the predicted label and target label agree, the classifier incurs no loss
- If the predicted label and target label *dis* agree, the classifier incurs a loss of 1.

$$\text{Loss}_{0/1}(x, y, \mathbf{w}) = 1_{[f_{\mathbf{w}}(x) \neq y]}$$

Example:

$$f_{\mathbf{w}}(x) = \text{sign}(\underbrace{[1, 0.5]}_{\mathbf{w}} \cdot \underbrace{[x_1, x_2]}_{\phi(x)})$$

$$\text{Loss}_{0/1}([0, 1], 1, [1, 0.5]) = 1_{[\text{sign}([1,0.5]\cdot[0,1]) \neq 1]} = 0$$

$$\text{Loss}_{0/1}([1, -1], -1, [1, 0.5]) = 1_{[\text{sign}([1,0.5]\cdot[1,-1]) \neq -1]} = 1$$

...

x_1	x_2	y
0	1	1
2	0	1
-1	-1	-1
1	-1	-1

Machine Learning: Linear Classification

Score and Margin

Technische Hochschule Rosenheim

Regardless of the output function (e.g. sign), its argument is called the *score*:

⊙ Score

The score on an example (x, y) is $\mathbf{w} \cdot \phi(x)$.
It expresses how *confident* we are in predicting $+1$.

⊙ Margin

The margin on an example (x, y) is $(\mathbf{w} \cdot \phi(x))y$.
It expresses how *correct* we are (regardless of the label).

Based on the margin, we can define the Zero-One Loss:

⊙ Zero-One Loss

$$\begin{aligned} \text{Loss}_{0/1}(x, y, \mathbf{w}) &= 1_{[f_{\mathbf{w}}(x) \neq y]} \\ &= 1_{[\underbrace{(\mathbf{w} \cdot \phi(x))y}_{\text{margin}} \leq 0]} \end{aligned}$$

→ However, the gradient of the Zero-One Loss w.r.t. the margin is Zero everywhere. Therefore, gradient descent won't work on this loss function.

Machine Learning: Linear Classification

Loss function: Hinge Loss

A convenient generalization of the Zero-One Loss is called the *Hinge Loss*.

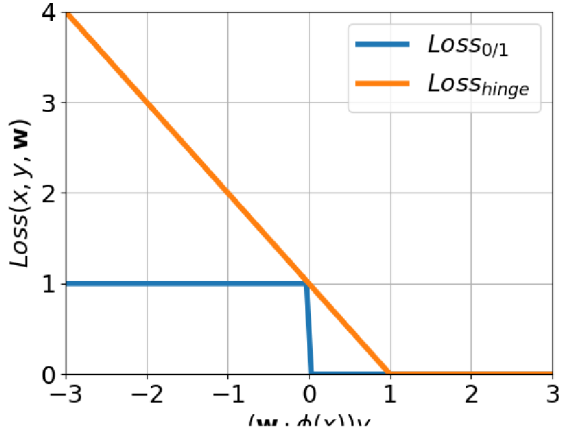
It exhibits the following intuition:

- Instead of incurring a constant loss of 1, incur a loss proportional to how incorrect the classification was.
- The hinge loss incurs a loss based on the margin.

⊙ Hinge Loss

$$\text{Loss}_{hinge}(x, y, \mathbf{w}) = \max\{1 - \underbrace{\underbrace{(\mathbf{w} \cdot \phi(x))}_{\text{score}} y}_{\text{margin}}, 0\}$$

$\rightarrow$ The hinge loss encourages the predictor to classify examples confidently $\underbrace{|\mathbf{w} \cdot \phi(x)|}_{\text{score}} \geq 1$ and correctly.



The gradient is non-zero for a margin < 1.

$$\nabla \text{Loss}_{hinge}(x, y, \mathbf{w}) = \begin{cases} -\phi(x)y, & \text{if } 1 - (\mathbf{w} \cdot \phi(x))y > 0 \\ 0, & \text{otherwise.} \end{cases}$$

$\rightarrow$ Optimization can be performed using gradient descent.

Machine Learning
Stochastic Gradient Descent

Technische Hochschule Rosenheim

Problem

Each update to the weights requires a full iteration over all training examples.
This is a serious issue when training on larger datasets with $|\mathcal{D}_{train}| >> 10^4$ training examples.

Solution

⊙ Stochastic Gradient Descent

Instead of updating the weights with the training loss, we calculate the loss for each example and update the weights with just the gradient obtained on this example.

```
Initialize w = [0, ..., 0]
For t = 1, ..., T  (epochs)
  For (x,y) ∈ D[train]
    w ← w - η · ∇Loss(x, y, w)
```

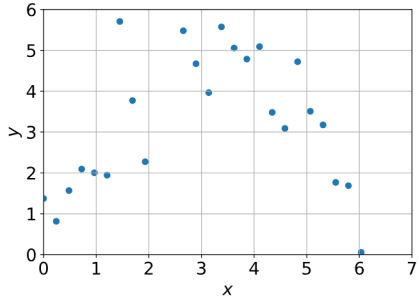
Since the loss induced by a single training example is only a very crude estimate of the actual training loss, the procedure is stochastic.
Thus, the losses might also increase temporarily while iterating over the training examples.

Machine Learning
Non-Linear Predictors

Technische Hochschule Rosenheim

Reminder: Hypothesis class of Linear Regression

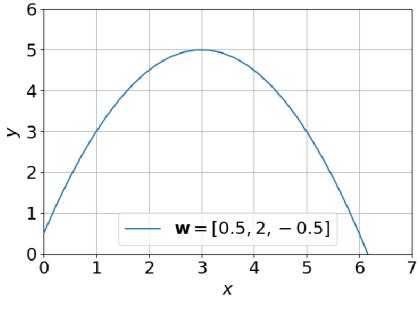
$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^d\}$$



How can we fit a non-linear predictor?

Use machinery of linear predictors but transform inputs non-linearly using the feature map ϕ.

- The the predictor is still linear in $\mathbf{w}$ and $\phi(x)$
- But, it is non-linear in x.



Example: Quadratic predictor

Consider a dataset with three examples and just a single input $x \in \{1, 3, 4\}$.
We are free to choose any *feature map* ϕ. For instance:

$$\phi(x) = [1, x, x^2]$$

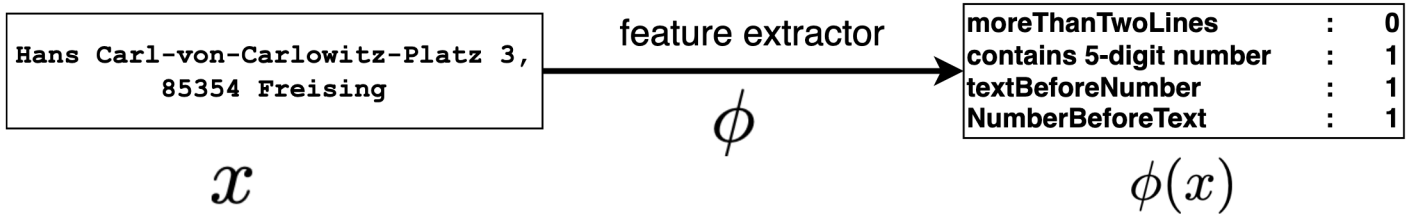
The *feature vector* for input $x = 4$ is:

$$\phi(4) = [1, 4, 16]$$

Machine learning
Feature Extraction

Technische Hochschule Rosenheim

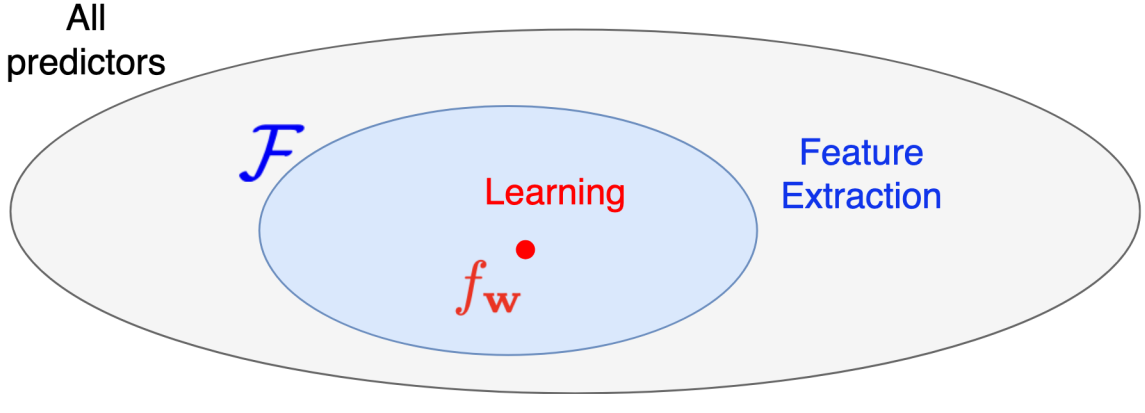
Example: Address detection



Which properties of x are relevant for predicting y?

- Requires domain knowledge
- Design of ϕ is manual effort (feature engineering)
- No learning algorithm can overcome bad feature engineering

By choosing features, we commit to a hypothesis class:



Machine Learning

Summary

Technische Hochschule Rosenheim

A *learning algorithm* takes in *training data* and produces a *predictor*.

The set of predictors we consider is called the *hypothesis class*.

→ In our example: Linear functions

A *loss function* measures the quality of a predictor.

→ Squared loss, Zero-One Loss, Hinge Loss

An *optimization algorithm* is used to find the best predictor within the hypothesis class.

→ Gradient descent, Stochastic Gradient Descent

A *linear classifier* maps the *score* to either -1 or +1 using the sign-function.

References

Technische Hochschule Rosenheim

- Material based on Stanford's CS221. Check out the Course.