Lecture 1

PHYS 568

[Kevin P. Murphy "Prob. ML"]

Introduction

Machine learning (ML): automated discovery of patterns & regularities in data, using these patterns (learned) to make new predictions.

(A.) Supervised learning

The task is to learn a mapping

f: inputs $\vec{x}$ → outputs $\vec{y}$

$\vec{x}$ vectors are often called features, while $\vec{y}$ vectors are often called targets.

In many cases, $\vec{x} \in \mathbb{R}^D$ or some subset of $\mathbb{Z}^D$

The mapping is learned using training data: $\mathcal{D} = \{(\vec{x}_n, \vec{y}_n)\}_{n=1}^{N}$, where

N = sample size

(1.) Classification task

$\tilde{y} \in Y = \{1, 2, \ldots, C\}$ ← # classes

↑ class labels

Binary classification: $y \in \{0, 1\}$, or $y \in \{-1, 1\}$.

Ex. [Iris flower dataset]

class labels:

C	Name
1	Setosa
2	Versicolor
3	Virginica

Data:

i	sl (cm)	sw	pl	pw	C
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.5	0.2	1
⋮					
→ 149					

N = 150, 50 examples per class

4 features (expert-identified):

$\tilde{x} \in \mathbb{R}^4$.

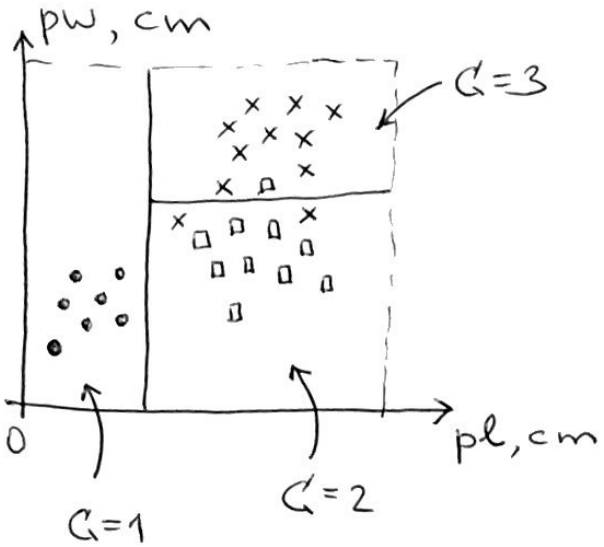
In contrast, in classification from raw images we have: $\tilde{x} \in \tilde{Z}^{3 \times D_1 \times D_2}$

$\tilde{Z} = \{0, 1, \ldots, 255\}$

↑ RGB channels

$D_1 \times D_2$ in pixels

linear classifier:



Misclassification rate: (n^{th} datapoint)

$$R(\vec{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(1-\delta_{y_n, f(\vec{x}_n, \vec{\theta})})$$

($\vec{\theta}$: model prms; y_n: true label)

$$\hat{\vec{\theta}} = \underset{\vec{\theta}}{\operatorname{argmin}}\, \mathcal{I}(\vec{\theta})$$ model fitting

---

Uncertainty in predictions:

$$p(y=c \mid \vec{x}, \vec{\theta}) \equiv f_c(\vec{x}, \vec{\theta})$$ ← prob. of label c

$$0 \le f_c \le 1 \;, \quad \sum_{c=1}^{C} f_c = 1 .$$

max. likelihood estimation:

$$\log \mathcal{I}(\vec{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \log p(y_n \mid \vec{x}_n, \vec{\theta})$$

$$\hat{\vec{\theta}} = \underset{\vec{\theta}}{\operatorname{argmax}}\, \mathcal{I}(\vec{\theta})$$ ML estimate

(2.) Regression task

Predict $y \in R^D$ instead of class labels;

often, $D = 1$.

Minimize mean-squared error (MSE):

$$MSE(\vec{\theta}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\vec{x}_n, \vec{\theta}))^2$$

Prob. modeling: assume that

$$p(y | \vec{x}, \vec{\theta}) = \mathcal{N}(y | f(\vec{x}, \vec{\theta}), \sigma^2),$$

where $$\mathcal{N}(y | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

Note that (assume that σ is known)

$$-\log \mathcal{L}(\vec{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_n - f(\vec{x}_n, \vec{\theta}))^2}{2\sigma^2}} \right] \quad (=)$$

$\uparrow$

$$\mathcal{L} = \prod_{n=1}^{N} \mathcal{N}(y_n | f(\vec{x}_n, \vec{\theta}), \sigma^2)$$

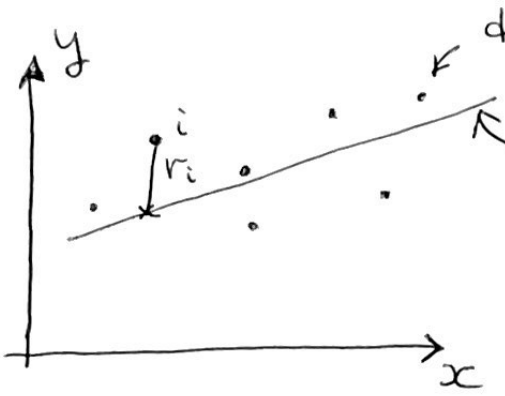
$$(=) \frac{1}{2\sigma^2} MSE(\vec{\theta}) + \text{const}.$$

Thus, maximizing $\log \mathcal{L}$ = minimizing MSE

$$\hat{\vec{\theta}} = \arg\max_{\vec{\theta}} [\log \mathcal{L}(\vec{\theta})] \leftarrow \text{ML estimate}$$

Linear regression:

$f(x, \vec{\theta}) = b + wx$ (1D input data)



$\vec{\theta} = (b, w)$

Residual $r_i = y_i - f(x_i, \vec{\theta})$

The task is to minimize $\sum_{i=1}^{N} r_i^2$

For multi-D $\vec{x}$,

$f(\vec{x}, \vec{\theta}) = b + \vec{w}^T \cdot \vec{x}$ ← multiple linear regression

$\vec{\theta} = (b, \vec{w})$

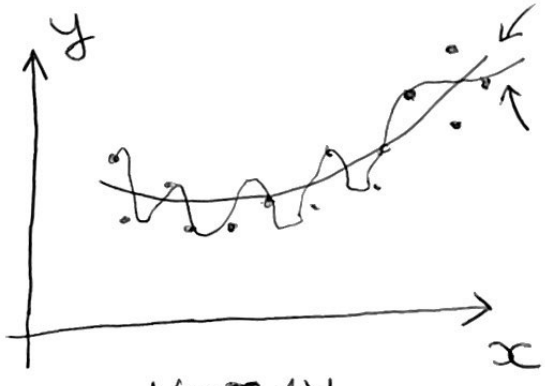
Polynomial regression:

1D input data: $f(x, \vec{w}) = \vec{w}^T \cdot \vec{\varphi}(x)$,

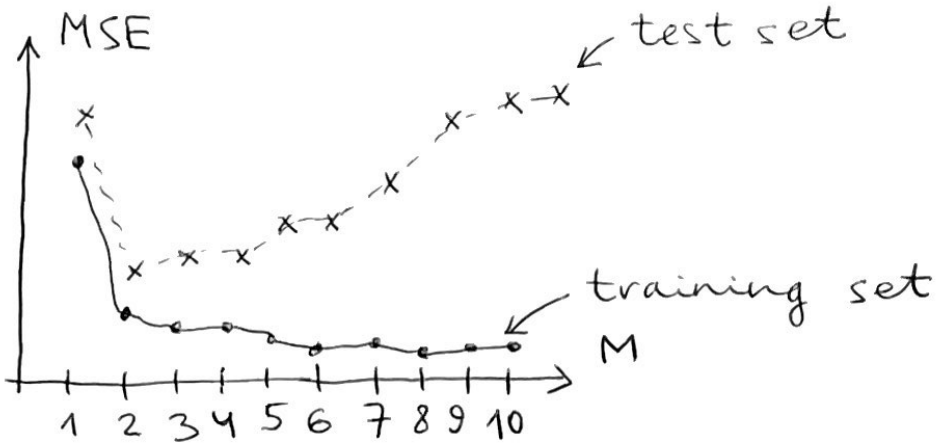
where $\vec{\varphi}(x) = (1, x, x^2, \ldots, x^M)$

↑ feature vector

Ex.



N = 14



Neural networks:

Key idea: instead of specifying features manually, do feature extraction automatically: $f(\vec{x}; \vec{w}, \vec{V}) = \vec{w}^T \cdot \vec{\varphi}(\vec{x}; \vec{V})$

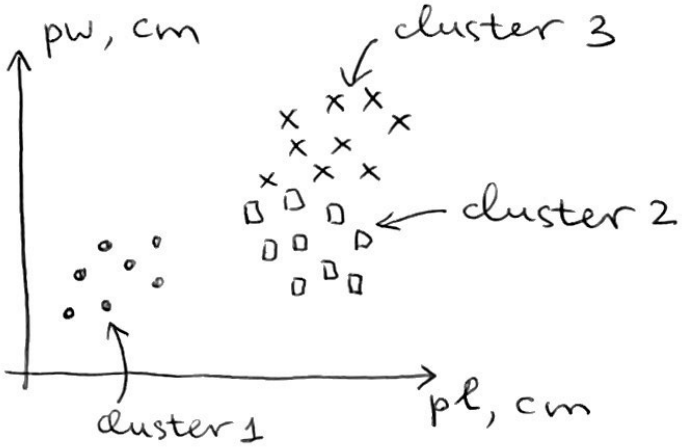
↑ feature prms

(B) Unsupervised learning

Observe input data $\mathcal{D} = \{\vec{x}_n\}_{n=1}^N$, but no target variables.

Clustering:

K=3
↑
clusters
(fixed or learned)



Dimensionality reduction:

project data to lower-D subspace, while preserving the "essence" of the data.

For example, we can assume that each observed $\vec{x}_n \in R^D$ was generated by latent factors $\vec{z}_n \in R^K$, $K \ll D$.

Probabilistic Principal Components Analysis (PCA):

$$p(\vec{x}_n | \vec{z}_n, \vec{\theta}) = \mathcal{N}(\vec{x}_n | \underbrace{W\vec{z}_n + \vec{\mu}}_{\text{linear assumption}}, \sigma^2 \mathbb{I})$$

(c.) Reinforcement learning (RL)

Agent learns to interact with its environment to maximize a reward.

Key idea: reward is only given at the end of a sequence of steps, not at every step.

The agent needs to learn a policy

$$\vec{a} = \pi(\vec{x})$$

↑ current state of the agent