Machine Learning and Data Mining

Christos Dimitrakakis

August 6, 2023

Outline

This course gives an introduction to the algorithms and theory of machine learning. Application is in the form of a course project. During the course, you will be able to:

- Formulate machine learning problems in terms of opimisation or probabilistic inference.
- Understand the fundamental machine learning algorithms.
- Be able to implement some of the simplest algorithms.
- Apply off-the-shelf algorithms to problems.
- Develop custom models using algorithms from TensorFlow python library.

Here is a summary of the scheduled topics for this course, together with the theory and practice focus.

Week	Date	Topic	Theory
1	09.19	Course Introduction	
2	09.26	kNN	Generalisation
3	10.03	Perceptron	Convergence
4	10.10	Linear Regression	SGD, Least-Squares
5	10.17	Multi-Layer Neural Network	Backpropagation
6	10.24	TensorFlow Lab	Network Architectures
7	10.31	Discriminative Models	Logistic Regression
8	11.07	Generative Models	Bayes Classifier
9	11.14	Bayesian Networks	Conditional Independence
10	11.21	Regularisation	Non-linear programming
11	11.28	Bayesian Inference	Conjugate priors
12	12.05	Approximate Bayesian Inference	Monte-Carlo Methods
13	12.12	Bayesian Neural Networks	Stochastic Variational Infere
14	12.19	Project Presentations	

Primary

- Introduction to Statistical Learning with Python

https://hastie.su.domains/ISLP/ISLP_website.pdf

- Elements of Statistical Learning

https://hastie.su.domains/Papers/ESLII.pdf

Secondary

- Probabilistic Machine Learning: An Introduction

https://probml.github.io/pml-book/book1.html
https://github.com/probml/pml-book/releases/latest/download/book1.pdf

- Probabilistic Machine Learning: Advanced Topics

https://probml.github.io/pml-book/book2.html
https://github.com/probml/pml2-book/releases/latest/download/book2.pdf

Class data

Fill in your data (does not have to be true)

The main problems in machine learning and statistics

Prediction

- Will it rain tomorrow?
- How much will bitcoin be worth next year?

Inference

- Does my poker opponent have two aces?
- What is the mass of the moon?
- What is the law of gravitation?

Decision Making

- Should I go hiking tomorrow?
- Should I buy some bitcoins?
- Should I fold, call, or raise in my poker game?
- How can I get a spaceship to orbit the moon?

The need to learn from data

Problem definition

- What problem do we need to solve?
- How can we formalise it?
- What properties of the problem can we learn from data?

Data collection

- Why do we need data?
- What data do we need?
- How much data do we want?
- How will we collect the data?

Modelling and decision making

- How will we compute something useful?

Learning from data

Unsupervised learning

- Given data $x_1, \ldots, x_T$.
- Learn about the data-generating process.

Supervised learning

- Given data $(x_1, y_1), \ldots, (x_T, y_T)$
- Learn about the relationship between x_t and y_t.
- Example: Classification, Regression

Online learning

- Sequence prediction: At each step t, predict x_{t+1} from $x_1, \ldots, x_t$.
- Conditional prediction: At each step t, predict y_{t+1} from $x_1, y_1 \ldots, x_t, y_t, \color{red}{x_{t+1}}$

Reinforcement learning

Learn to act in an unknown world through interaction and rewards

Unsupervised learning

Image compression

- Learn two mappings c, d
- $c(x)$ compresses an image x to a small representation z.
- $d(z)$ decompresses to an approximate image $\hat{x}$.

Supervised learning

Image classification

Unsupervised learning

Density estimation

Compression

Generative modelling

Pitfalls

Reproducibility

- Modelling assumptions
- Distribution shift
- Interactions and feedback

Fairness

- Implicit biases in training data
- Fair decision rules and meritocracy

Privacy

- Accidental data disclosure
- Re-identification risk

Schedule

Material

The problems of Machine Learning (1 week)

Learning as Optimisation (4 weeks)

- Objective functions
- k Nearest Neighbours
- Learning and generalisation
- Linear neural networks
- Multi-layer neural networks

Learning as Probabilistic Inference (4 weeks)

Sequence modelling (2 weeks)

Reinforcement Learning (2 weeks)

Supervised learning objectives

- Data (x_t, y_t), $x_t \in X$, $y_t \in Y$, $t \in [T]$.
- i.i.d assumption: $(x_t, y_t) \sim P$ for all t.
- Supervised decision rule $\pi(a_t | x_t)$

Classification

- Predict the labels correctly, i.e. $a_t = y_t$.
- Have an appropriate confidence level

Regression

- Predict the mean correctly
- Have an appropriate variance around the mean

Unsupervised learning objectives

- Reconstruct the data well
- Model the data-generating distribution
- Be able to generate data

Reinforcement learning objectives

- Maximise total expected reward, either
- during learning, or
- after learning is finished.

A simple classification problem

Income distribution data:

- $x \in \{\mathrm{M}, \mathrm{F}\}$, gender.
- $y \in \mathbb{R}$, income.

Problem

- Can we model the income distribution?

The Nearest Neighbour algorithm

Pseudocode

- Input: Data $(x_t, y_t)_{t=1}^{T}$, test point x, distance d
- $t^* = \arg\min_t d(x_t, x)$
- Return $y^* = y_{t^*}$

Classification

$y_t \in [m] \equiv \{1, \ldots, m\}$ See example code

Regression

$y_t \in \mathbb{R}^m$

The k-Nearest Neighbour algorithm

Pseudocode

- Input: Data $(x_t, y_t)_{t=1}^{T}$, test point x, distance d, neighbours k
- Calculate $h_t = d(x_t, x)$ for all t.
- Get sorted indices $s = \texttt{argsort}(h)$ so that $d(x_{s_i}, x) \leq d(x_{s_{i+1}}, x)$ for all i.
- Return $\sum_{i=1}^{k} y_{s_i} / k$.

Classification

- It is not convenient to work with discrete labels
- We use a one-hot encoding vector representation $(0, \ldots, 0, 1, 0, \ldots, 0)$.
- $y_t \in \{0, 1\}^m$ with $\|y_t\|_1 = 1$, so that the class of the t-th example is j iff $y_{t,j} = 1$.

Regression

$y_t \in \mathbb{R}^m$
Code:

The Train/Test methodology

Training data $D = ((x_t, y_t) : t = 1, \ldots, T)$.

- $x_t \in X$
- $y_t \in \mathbb{R}^m$.

Assumption: The data is generated i.i.d.

- $(x_t, y_t) \sim P$ for all t (identical)
- $D \sim P^T$ (independent)

The optimal decision rule for P

$$\max_{\pi} U(\pi, P) = \max_{\pi} \int_{X \times Y} dP(x, y) \sum_{a} \pi(a|x) U(a, y)$$

The optimal decision rule for D

$$\max_{\pi} U(\pi, D) = \max_{\pi} \sum_{(x,y) \in D} \sum_{a} \pi(a|x) U(a, y)$$

Generalisation

Error due to mismatched objectives

The π^* maximising $U(\pi, P)$ is not the $\hat{\pi}$ maximising $U(\pi, D)$.

Lemma

If $|U(\pi, P) - U(\pi, D)| \leq \epsilon$ for all π then

$$U(\hat{\pi}, D) \geq U(\pi^*, P) - 2\epsilon.$$

Error due to restricted classes

- We may use a constrained $\hat{\Pi} \subset \Pi$.
- Then $\max_{\hat{\pi} \in \hat{\Pi}} U(\pi, P) \leq \max_{\pi \in \Pi} U(\pi, P)$.

Classification

The classifier as a decision rule

A decision rule $\pi(a|x)$ generates a decision $a \in [m]$. It is the conditional probability of a given x.
Even though normally conditional probabilities are defined as $P(A|B) = P(A \cap B)/P(B)$, the probability of the decision a is undefined without a given x. So it's better to

The accuracy of a single decision

$$U(a_t, y_t) = \mathbb{I}\{a_t = y_t\} = \begin{cases} 1, & \text{if } a_t = y_t \\ 0, & \text{otherwise} \end{cases}$$

$$U(\pi, D) \triangleq \frac{1}{T} \sum_{t=1}^{T} \sum_{a=1}^{m} \pi(y_t|x_t)$$

The accuracy on the training set

$$U(\pi, D) \triangleq \frac{1}{T} \sum_{t=1}^{T} \sum_{a=1}^{m} \pi(y_t|x_t)$$

Regression

The regressor as a decision rule

A decision rule $\pi(a|x)$ generates a decision $a \in \mathbb{R}^m$. It is the conditional density of a given x.

Accuracy

If $(x, y) \sim P$, the accuracy U of a decision rule π under the distribution P is:

$$U(\pi, P) \triangleq \int_X \int_Y dP(x, y)\pi(y|x).$$

Mean-Squared Error

If $(x, y) \sim P$, the mean-square error of a deterministic decision rule $\pi : X \to \mathbb{R}$ under the distribution $P(x, y) = P(x|y)P(y)$ is:

$$\int_X \sum_{y=1}^m dP(x|y)P(y) \sum_{a=1}^m \pi(a|x)$$

The perceptron algorithm

Input

- Feature space $X \subset \mathbb{R}^n$.
- Label space $Y = \{-1, 1\}$.
- Data (x_t, y_t), $t \in [T]$, with $x_t \in X, y_t in Y$.

Algorithm

- $w_1 = w_0$.
- For $t = 1, \ldots, T$.

– $a_t = \mathrm{sgn}(w_t^\top x_t)$. – If $a_t \neq y_t$ — $w_{t+1} = w_t + y_t x_t$ – Else — \$w_{t+1} = w_t

- Return w_{T+1}

Theorem

The number of mistakes made by the perceptron algorithm is bounded by $(r/\rho)^2$, where $\|x_t\| \leq r$, $\rho \leq y_t(v^\top x_t)/\|v\|$ for some margin ρ and hyperplane v.

Perceptron examples

Example 1: One-dimensional data

- Done on the board
- Shows how the algorithm works.
- Demonstrates the idea of a margin

Example 2: Two-dimensional data

- See in-class programming exercise

Python concepts

Numpy

- np.random.multivariate$_{\text{normal}}$(): generate samples from an n-D normal distribution
- np.random.choice(): generate samples from a discrete distribution
- np.zeros(): generate an array of zeros
- np.array(): create an array from a list
- np.block(): make an array from nested lists
- np.dot(): calculate the dot (aka inner) product

matplotlib.pyplot

- plt.plot(): Plot lines and points
- plt.axis(): manipulate axes
- plt.grid(): show a grid
- plt.show(): display the plot

Gradient methods example

Estimate the expected value

$x_t \sim P$ with $\mathbb{E}_P[x_t] = \mu$.

Objective

$$\min_\theta \mathbb{E}_P[(x_t - \theta)^2].$$

Derivative

Idea: at the minimum the derivative should be zero.

$$d/d\theta\, \mathbb{E}_P[(x_t - \theta)^2] = \mathbb{E}_P[d/d\theta (x_t - \theta)^2] = \mathbb{E}_P[-(x_t - \theta)] = \mathbb{E}_P[x_t] - \theta.$$

Setting the derivative to 0, we have $\theta = \mathbb{E}_P[x_t]$. This is a simple solution.

Real-world setting

- The objective function does not result in a simple solution
- The distribution P is not known.
- We can sample $x \sim P$.

Stochastic gradient for mean estimation

$$\begin{aligned}\frac{d}{d\theta}\,\mathbb{E}_P[(x-\theta)^2] &= \int_{-\infty}^{\infty} dP(x)\frac{d}{d\theta}(x-\theta)^2 \\ &= \frac{d}{d\theta}\int_{-\infty}^{\infty} dP(x)(x-\theta)^2\end{aligned}$$

Simple linear regression

Input and output

- Data pairs (x_t, y_t), $t = 1, \ldots, T$.
- Input $x_t \in \mathbb{R}^n$
- Output $y_t \in \mathbb{R}$.

Predicting the conditional mean $\mathbb{E}[y_t | x_t]$

- Parameters $\theta \in \mathbb{R}^n$
- Function $f_\theta : \mathbb{R}^n \to \mathbb{R}$, defined as

$$f_\theta(x_t) = \theta^\top x_t = \sum_{i=1}^n \theta_i x_{t,i}$$

Optimisation goal: Miniminise mean-squared error.

$$\min_\theta \sum_{t=1}^T [y_t - \pi_\theta(x_t)]^2$$

How can we solve this problem?

Gradient descent algorithm

Minimising a function

$$\min_{\theta} f(\theta) \geq f(\theta')\forall\theta', \qquad \theta^* = \arg\min_{\theta} f(\theta) \Rightarrow f(\theta^*) = \min_{p} aramf(\theta)$$

Gradient descent for minimisation

- Input θ_0
- For $n = 0, \ldots, N$:
- $\theta_{n+1} = \theta_n - \eta_n \nabla_\theta f(\theta_n)$

Step-size η_n

- η_n fixed: for online learning
- $\eta_n = c/[c + n]$ for asymptotic convergence
- $\eta_n = \arg\min_\eta f(\theta_n + \eta\nabla_\theta)$: Line search.

Gradient desecnt for squared error

Cost function

$$\ell(\theta) = \sum_{t=1}^{T} [y_t - \pi_\theta(x_t)]^2$$

Cost gradient

Using the chain rule of differentiation:

$$\begin{aligned}
\nabla_\theta \ell(\theta) &= \nabla \sum_{t=1}^{T} [y_t - \pi_\theta(x_t)]^2 \\
&= \sum_{t=1}^{T} \nabla [y_t - \pi_\theta(x_t)]^2 \\
&= \sum_{t=1}^{T} 2[y_t - \pi_\theta(x_t)][-\nabla \pi_\theta(x_t)]^2
\end{aligned}$$

Parameter gradient

Analytical Least-Squares Solution

Stochastic gradient descent algorithm

When f is an expectation

$$f(\theta) = \int_X dP(x) g(x, \theta).$$

Replacing the expectation with a sample:

$$\begin{aligned} \nabla f(\theta) &= \int_X dP(x) \nabla g(x, \theta) \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \nabla g(x^{(k)}, \theta), \qquad\qquad x^{(k)} \sim P. \end{aligned}$$

Back-propagation

The chain rule

$$f : X \to Z, \qquad g : Z \to Y, \qquad \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}$$

Parametrised functions

$$f : \mathcal{W} \times X \to Z, \qquad g : \Omega \times Z \to Y, \qquad \pi = fg$$

(network mappings)

$$\ell(D, \pi) = \sum_{(x,y)\in D} [y - \pi(x)]^2 \tag{1}$$

Gradient descent with *back-propagation*

Apply the chain rule

$$\nabla_{w,\omega}\pi = \nabla_{\omega}$$

Neural architectures

Layers

- Input to layer $x \in R^n$
- Output from layer $z \in R^m$.

Linear layer

Transform the output of previous layers or features into either:

- A higher-dimensional space.
- A lower-dimensional space.
- They have adaptive parameters.
- Parameters can be dependent on each other for invariance (cf. convolution)

Non-linear layers

- Simple transformations of previous output
- Examples: Sigmoid, Softmax

Liner layer

Definition

This is a linear combination of inputs $x \in \mathbb{R}^n$ and parameter matrix

$$\boldsymbol{W} \in \mathbb{R}^{m \times n} \text{ where } \boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_i \\ \vdots \\ \boldsymbol{w}_m \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,j} & \cdots & w_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \cdots \\ w_{i,1} & \cdots & w_{i,j} & \cdots & w_{i,m} \\ \vdots & \ddots & \ddots & \ddots & \cdots \\ w_{n,1} & \cdots & w_{i,j} & \cdots & w_{n,m} \end{bmatrix}$$

$$f(\boldsymbol{W}, \boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} \qquad f_i(\boldsymbol{W}, \boldsymbol{x}) = \boldsymbol{w}_i \cdot \boldsymbol{x} = \sum_{j=1}^{n} w_{i,j} x_i,$$

Gradient

Each partial derivative is simple:

$$\frac{\partial}{\partial w_{i,j}} f_k(\boldsymbol{W}, x) = x_i \, \mathbb{I}\{j = k\}$$

Sigmoid layer

Definition

This layer transforms each input non-linearly

$$f_j(\boldsymbol{x})1/[1+\exp(-x_j)] =$$

without looking at the other inputs.

Derivative

So let us ignore the other inputs for simplicity:

$$\frac{d}{dx}f(x) = \exp(-x)/[1+\exp(-x)]^2$$

Softmax

Probabilistic modelling

The problem

- Model family $\{P_\theta : \theta \in \Theta\}$
- Each model assigns a probability $P_\theta(x)$ to the data x.
- How can we estimate θ from x?

Maximum Likelihood (ML) Estimation

$\hat{\theta}(x) = \arg\max_\theta P_\theta(x)$.

Maximum A Posteriori (MAP) Estimation

Here we also need a prior distribution, but still estimate a single parameter:

- Prior $\beta(\theta)$, a distribution on Θ.
- $\hat{\theta}(x) = \arg\max_\theta P_\theta(x)\beta(\theta)$.

Bayesian Estimation

Here we estimate the complete distribution over parameters

- $\beta(\theta|x) = P_\theta(x)\beta(\theta) / \sum_{\theta'} P_{\theta'}(x)\beta(\theta')$

The Bernoulli distribution: Modelling a coin

Definition

If $x_t \sim \mathrm{Bernoulli}(\theta)$ then $x_t = 1$ w.p. θ and $x_t = 0$ w.p. $1 - \theta$.

Maximum Likelihood Estimate

$\hat{\theta}_t = \frac{1}{t} \sum_{k=1}^{t} x_k$

Bayesian Estimate

- Prior $\theta \sim \mathrm{Beta}(\alpha_1, \alpha_0)$
- Posterior $\theta \sim \mathrm{Beta}(\alpha_1 + \sum_{k=1}^{t} x_k, \alpha_0 + \sum_{k=1}^{t} x_k)$.

The Gaussian distribution: Modelling gambling gains

Discriminative modelling: general idea

- Data (x, y)
- Easier to model $P(y|x)$
- No need to model $P(x)$.

Examples

- Linear regression
- Logistic regression
- Multi-layer perceptron

Linear regression

Model

- $z = \theta^\top x$
- $p_\theta(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}|z - y|^2)$

Two-class classification: logistic regression

Model

- $z = \theta^\top x$
- $P_\theta(y = 1|x) = \frac{1}{1-e^z}$

Generative modelling

general idea

- Data (x, y).
- Need to model $P(y|x)$.
- Model the complet data distribution: $P(x|y)$, $P(x)$, $P(y)$.
- Calculate $P(y|x) = \frac{P(x|y)P(x)}{P(y)}$.

Examples

- Naive Bayes classifier
- Density estimation
- Sequence models

Classification: Naive Bayes Classifier

- Data (x, y)
- $x \in X$
- $y \in Y \subset \mathbb{N}$, N_i: amount of data from class i

Separately model each class

- Assume each class data comes from a different normal distribution
- $x|y = i \sim \mathrm{Normal}(\mu_i, \sigma_i I)$
- For each class, calculate
  - Empirical mean $\hat{\mu}_i = \sum_{t:y_t=i} x_t / N_i$
  - Empirical variance $\hat{\sigma}_i$.

Decision rule

Use Bayes's theorem:

$$P(y|x) = P(x|y)P(y)/P(x),$$

choosing the y with largest posterior $P(y|x)$.

- $P(x|y = i) \propto \exp(-\|\hat{\mu}_i - x\|^2 / \hat{\sigma}_i^2$

Density estimation

Schedule

Material

The problems of Machine Learning (1 week)

Learning as Optimisation (4 weeks)

Learning as Probabilistic Inference (4 weeks)

- Probabilistic Models
- Discriminative modelling
- Generative modelling

Sequence modelling (2 weeks)

Reinforcement Learning (2 weeks)

The problem of sequence prediction

- Data $x_1, x_2, x_3, \ldots$
- At time t, make a prediction a_t for x_t.

Auto-regressive models

General idea

- Predict x_t from the last k inputs

$$x_t \approx g(x_{t-k}, \ldots, x_{t-1})$$

Optimisation view

We wish to minimise the difference between our predictions a_t and the next symbol

$$\sum_t (a_t - x_t)^2$$

Probabilistic view

We wish to model

$$P(x_t | x_{t-k}, \ldots, x_{t-1})$$

Linear auto-regression

Recursive models

General idea

- Maintain an *internal state* z_t, which summarises what has been seen.

$$z_t = f(z_{t-1}, x_{t-1}) \qquad \text{(change state)}$$

- Make predictions using the internal state

$$\hat{x}_t = g(z_t) \qquad \text{(predict)}$$

Examples

- Hidden Markov models
- Recurrent Neural Networks

Hidden Markov Models: General setting

Variables

- State z_t
- Observations x_t

Parameters

- Transition θ
- Observation ψ

Distributions

- Transition distribution $P_\theta(z_{t+1}|z_t)$
- Observation distribution $P_\psi(x_t|z_t)$.

HMMs: Discrete case

Variables

- State $z_t \in [n]$
- Observation $x_t \in [m]$

Transition distribution

Multinomial with

$$P_\theta(z_{t+1} = j | z_t = i) = \theta_{i,j}$$

Observation distribution

Multinomial with

$$P_\theta(x_t = j | z_t = i) = \psi_{i,j}$$

HMMs: Continuous case

Variables

- State $z_t \in [n]$
- Observation $x_t \in \mathbb{R}^m$

Transition distribution

Multinomial with

$$P_\theta(z_{t+1} = j | z_t = i) = \theta_{i,j}$$

Observation distribution

Gaussian with

$$P_\theta(x_t = x | z_t = i) \propto \exp\left(-\|x - \psi_i\|\right)$$

Density Estimation with EM

HMM Estimation with EM

Schedule

Material

The problems of Machine Learning (1 week)

Learning as Optimisation (4 weeks)

Learning as Probabilistic Inference (4 weeks)

Sequence modelling (2 weeks)

Reinforcement Learning (2 weeks)