Lecture Notes

Machine Learning

Winter Semester 24/25

Emma Bach

Lecture by
Dr. Daniele Cattaneo

March 1, 2025

Table of contents

1 Introduction **2**
1.1 Basic Prediction Models ... 2
1.1.1 Decision Trees ... 3
1.1.2 Neural Networks ... 3
1.2 Common Loss Functions ... 3
1.3 Overfitting and Underfitting ... 4
1.3.1 Regularization ... 4
1.4 ML Design Cycle ... 4
1.5 Non-Parametric Models ... 4
1.5.1 Nearest Neighbors ... 4
1.5.2 Naive Bayes ... 4

2 Linear Methods **6**
2.1 Linear Regresion ... 6
2.2 Linear Classification ... 7
2.2.1 Gradient Descent ... 7

3 Principles of Regularization **8**
3.1 Bias-Variance Tradeoff ... 8
3.2 Regularization ... 9

Chapter 1

Introduction

Given a set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, i.e. a set of features (inputs) $x = \{x_1, \ldots, x_n\}$ and a target (desired output) $y = \{y_1, \ldots, y_n\}$, Machine Learning is the process of algorithmically finding the best possible function $f(x)$ such that

$$\forall x_i [f(x_i) \approx y_i].$$

y is also known as the **ground truth**. In **unsupervised learning**, y is not explicitly given to the algorithm (and may not even exist). The focus of the course, and thus these notes, was **supervised learning**, where y is explicitly given.

In order to find an optimal $f(x)$, we first rephrase the function in terms of **learnable parameters** θ to get a function $f(x, \theta)$. We often write $f(x_i, \theta) = \hat{y}_i$, where the hat $\hat{}$ is supposed to show that $\hat{y}_i$ is an approximation of y_i. The full image $\hat{y}$ is often called the **prediction model**.

We now need to define a seperate function that we can use to judge how good our values of θ are. Such a function is known as a **loss function**, which we write as $\mathcal{L}(y, \hat{y})$. The problem of finding the best possible f then comes down to a minimization problem of the form

$$\underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \mathcal{L}(y_n, \hat{y}_n)$$

This function that we want to minimize is called the **empirical risk**.

1.1 Basic Prediction Models

Common basic prediction models include:

- Linear Models:

$$\hat{y_n} = \theta_0 + \sum_{m=1}^{M} \theta_m x_{n,m}$$

- Polynomial Regression Models:

$$\begin{aligned} \hat{y_n} = \theta_0 &+ \sum_{m=1}^{M} \theta_m x_{n,m} \\ &+ \sum_{m=1}^{M} \sum_{m'=1}^{M} \theta_{m,m'} x_{n,m} x_{n,m'} \\ &+ \sum_{m=1}^{M} \sum_{m'=1}^{M} \sum_{m''=1}^{M} \theta_{m,m',m''} x_{n,m} x_{n,m'} x_{n,m''} \\ &+ \ldots \end{aligned}$$

- Decision Trees
- Neural Networks

1.1.1 Decision Trees

In a Decision Tree, the model gives its answer by starting at the root node of a binary tree and then moving from each node to one of its children by answering a yes/no question at each leaf. A decision tree can be thought of as approximating an arbitrary function through a piecewise combination of constant functions.

1.1.2 Neural Networks

In a Neural Network, a given neuron labeled i consists of a non-linear function $f_i(x, \theta_i)$, such that if the output of a neuron i is connected to a neuron j, then the output $\hat{y_{n,j}}$ of the neuron j is

$$\hat{y_{n,j}} = f_j(f_i(x, \theta_i), \theta_j)$$

1.2 Common Loss Functions

A **Regression Problem** is a problem where the output is an arbitrary real number. Common loss functions for regression include:

- Least Squares:

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

  Least Square Loss is an approximation of the maximum likelihood estimator of a linear model with normally distributed error (with a mean of 0).

- L1 Loss:

$$\mathcal{L}(y_n, \hat{y}_n) = |y_n - \hat{y}_n|$$

For **Binary Classification Problems**, where y_n can only take on two possible values, more specialized loss functions are used:

- Logistic Loss, $y_n \in \{0, 1\}$:

$$\mathcal{L}(y_n, \hat{y}_n) = -y_n \log(\hat{y}_n) - (1 - y) \log(1 - \hat{y}_n)$$

- Hinge Loss, $y_n \in \{-1, 1\}$:

$$\mathcal{L}(y_n, \hat{y}_n) = \max(0, 1 - y_n \hat{y}_n)$$

A **Softmax** is a type of function used to express **Non-Binary Classification Problems**, where targets $y_n \in 1, \ldots, C$, as a set of binary classification targets:

$$y_{n,k} = \begin{cases} 1 & y_n = k \\ 0 & y_n \neq k \end{cases}$$

The final result is then obtained by re-expressing the results as probabilities among classes as follows:

$$\hat{y}_{n,k} = \frac{e^{f(x_n, \theta_k)}}{\sum_{i=1}^{C} e^{f(x_n, \theta_i)}}$$

The exponential function is used to avoid negative probabilities. A common loss function for Softmax problems is **Logloss**:

$$\mathcal{L}(y_n, \hat{y}_n) = -\sum_{i=i}^{C} y_{n,c} \log(\hat{y}_n, c)$$

1.3 Overfitting and Underfitting

Two common issues every model needs to deal with are **Underfitting**, where the bias of a model is too high, leading to a model that is unable to capture the data's complexity, and **Overfitting**, where the variance of a model is too high, meaning that the model has captured noise in the training data and is unable to generalize to pieces of data not included in the training data. Very simple models like linear models tend to be prone to underfitting, while most modern models that are more complex tend to be more prone to overfitting. Decision Trees in particular are very prone to overfitting.

1.3.1 Regularization

Regularization is the process of fighting overfitting by reducing the the size of the parameters of a model. This is expressed by adding a **Regularization Function** as a penalty term to the underlying optimization problem of minimizing the empirical risk:

$$\underset{\theta}{\text{argmin}} \sum_{n=1}^{N} \mathcal{L}(y_n, \hat{y}_n) + \alpha\Omega(\theta)$$

Just as there are many different possible loss functions, there are also many different possible regularization functions. Using a regularization always comes with the tradeoff of increasing model bias (but is done in the vast majority of models, since with complex modern models, overfitting tends to be a more common issue than underfitting).

1.4 ML Design Cycle

1. Pre-processing
2. Feature extraction / Feature encoding
3. Feature selection
4. Machine learning
5. Evaluation / Model Selection
6. Post-processing

1.5 Non-Parametric Models

1.5.1 Nearest Neighbors

Predict target y_i by averaging over the targets of the k nearest points $x_1, \ldots, x_k$ to the point x_i. For a continuus target the aggregation is simply the mean:

$$f(x_i) = \frac{1}{k} \sum_{y_j \in Y_{near}}^{k} y_j \tag{1.1}$$

For classification, we get our probability of the target being a certain class by taking the number of points belonging to each class and dividing by the total number of points we sampled:

$$f_c(x) = \frac{|y_j y_j \in Y_{near} \wedge y_j = c|}{k} \tag{1.2}$$

1.5.2 Naive Bayes

Bayes Rule:

$$P(y|x) = P(y)\frac{P(x|y)}{P(x)} \tag{1.3}$$

Using the law of total probability this can be extended to:

$$P(y|x) = P(y)\frac{P(x|y)}{\sum_{y' \in Y} P(x|y')P(y')} \tag{1.4}$$

Note that all of this only formally works if all features are conditionally independent.

Chapter 2

Linear Methods

2.1 Linear Regresion

The regression problem:

- Dataset of N instances (x_i, y_i) (x_i is generally a vector)
- Find a model $\hat{y}$ that minimizes a loss function $\mathcal{L}$:

$$\underset{w \in W}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(y_i f(x_i; w))$$

- Very simple and basic model: **Linear Regression**

$$f(x_i; w) = w_0 + x_i w^T$$

  w_0 is a fixed offset and thus often called the **bias**, while w is the **weight vector** or **parameter vector**.

Linear Regressions are more powerful than they seem - for example, $f(x) = (x+1)^2$ can be written as

$$f(a, b) = a + 2b + 1$$

(such that $b = x$ and $a = x^2$). Formally, we can take any set of arbitrary **basis functions** $\phi_j(x_i)$ and arrive at a linear model

$$f(x_i; w) = w_0 + \sum_{j=1}^{M} w_j \phi_j(x_i)$$

Linear models are therefore linear in the weights w_i, but they don't have to be linear in the inputs x_i.

The **residual error** of a linear model is given by:

$$y_i - f(x_i; w)$$

For the loss function of a linear model, we just some the losses of the individual predictions. Typical loss choices are L2 loss $(y - \hat{y})^2$ and L1 loss $|y - \hat{y}|$

Let X be the input matrix, i.e.

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

A closed-form solution for linear regression using L2 loss exists under the following assumptions:

- The expected value of the residuals is 0: $\forall i : E(\epsilon_i) = 0$

- Residuals are uncorrelated and share the same variance: $\forall i : Var(\epsilon_i) = \sigma_2$
- Residuals follow a normal distribution: $\forall i : \epsilon_i \sim \mathcal{N}(0, \sigma^2)$

If these hold, then the closed-form solution is:

$$w = (X^T X)^{-1} X^T y \tag{2.1}$$

2.2 Linear Classification

Logistic regression is a way of using linear regression to solve classification problems, by bounding the output of a linear model to $0 \leq h_w(x) \leq 1$ by taking the sigmoid of our output. The output is then interpreted as the probability that $y = 1$ (formally, $h_w(x) = P(y = 1|x; w)$). It's kind of fucked up that it's called "logistic regression" instead of "logistic classification".

$$h_w(x) = \sigma(w^T x) \tag{2.2}$$

Alternative idea: predict $y = 1$ if $wx^T \geq 0$ and $y = 0$ otherwise (see support vector machines in a later chapter.)

We use **Logistic Loss**, as already seen in the introduction:

$$\mathcal{L}(y_i, h_w(x_i)) = \begin{cases} -log(1 - h_w(x_i)) & y_i = 0 \\ -log(h_w(x_i)) & y_i = 1 \end{cases} \tag{2.3}$$

Note: I wrote $h_w(x_i)$ here because that's what the lecture does, but there's no real reason not to just write $\hat{y}(x_i)$ instead.

2.2.1 Gradient Descent

:

- For linear regression using L2 loss:

$$\mathcal{L}(w) = \sum_{i=1}^{N} (y_i - f(x_i; w))^2 \tag{2.4}$$

$$\implies \frac{\partial}{\partial w} J(w) = \sum_{i=1}^{N} -2(y_i - f(x_i; w))x_i \tag{2.5}$$

- For logistic regression using logistic loss:

$$\mathcal{L}(w) = \sum_{i=1}^{N} -y_i log(h_w(x_i)) - (1 - y_i) log(1 - h_w(x_i)) \tag{2.6}$$

$$\implies \frac{\partial}{\partial w} J(w) = \sum_{i=1}^{N} -(y_i - h_w(x_i))x_i \tag{2.7}$$

Chapter 3

Principles of Regularization

Reminder: for $x \in \mathbb{R}$, the polynomial prediction model of degree M is simply:

$$\hat{y} = f(x;\theta) = \sum_{j=0}^{M} \theta_j x^j \tag{3.1}$$

Optimal values $\theta 1^*$ are learned by minimizing the empirical risk (which is just the L2 loss? Why is a new term needed here?)

$$\theta^* := \underset{\theta}{\text{argmin}} \sum_{i=1}^{N} (f(x_i;\theta) - y_i)^2 \tag{3.2}$$

Small M leads to underfitting, large M leads to overfitting.

3.1 Bias-Variance Tradeoff

The expected target of y given x is written

$$\overline{y}(x) := E_{y|x}(x) = \int_y y p(y|x) \tag{3.3}$$

The expected prediction model $\overline{f}(x)$ over sample datasets is the model we expect to obtain given that we train randomly sampled data D that was sampled following a distribution $\mathcal{P}$:

$$\overline{f}(x) := E_{D \sim P^N}[\hat{f}(x;D)] \tag{3.4}$$

The expected value of the loss function is also called the **expected test error**:

$$E_{(x,y)\sim P}[(\hat{f}(x;D) - y)^2] \tag{3.5}$$

It turns out that the expected test error can be directly decomposed into a simple function of the variance, the bias squared, and the expected noise:

$$E_{x,y,D}[(\hat{f}(x;D) - y)^2] = E_{x,D}[(\hat{f}(x;D) - \overline{f}(x;D))^2] \tag{3.6}$$
$$+ E_{x,D}[(\overline{f}(x;D) - \overline{y}(x))^2] \tag{3.7}$$
$$+ E_{x,y}[(\overline{y}(x) - y)^2] \tag{3.8}$$

Recall the definitions of noise, bias and variance:

- The **variance** of a random variable X is defined as:

$$\sigma = E[(X - E[X])^2] = E[(f - E(f))^2] = E[f - \overline{f}] \tag{3.9}$$

- The **bias** of a predictor X that predicts y is defined as:

$$b = E[E[X] - y] = E[\overline{f} - \overline{y}] \tag{3.10}$$

  (Note: Why $\overline{y}$ instead of y? I assume both work because its wrapped up in an expected value anyways?)

- The **noise** is the difference in our data from the expected value of the data, i.e:

$$(\bar{(}y) - y) \tag{3.11}$$

3.2 Regularization

An increase in model complexity will always lead to an increase in variance, but a decrease in bias. In practice, this means that any method of decreasing a models variance will increase the bias and vice versa. Since sensible models are more likely to overfit, Regularization is very commonly used, since it decreases variance. Common regularization functions are:

$$L1 : \Omega(\theta) = \frac{1}{|\theta|} \sum_{k=1}^{|\theta|} |\theta_k| \tag{3.12}$$

$$L2 : \Omega(\theta) = \frac{1}{|\theta|} \sum_{k=1}^{|\theta|} \theta_k^2 \tag{3.13}$$