# Machine Learning and Statistical Learning Lecture 1

Daniel Yu

September 9, 2024

## Contents

# 1 Review of Machine Learning Basics

**Statistical Learning**

- Input (aka feature/predictor): $\vec{x} = (\vec{x_1}, \dots, \vec{x_d}) = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \in X = \mathbb{R}^d$
- Output (label): $y \in C \subseteq \mathbb{R}$

Note: here y is an outcome we wish to predict from $\vec{x}$.

**Remark.** Measuring a pair $(\vec{x}, y)$ is a sample of pair of random variables $R.V.$ $(\vec{X}, Y)$ with underlying distributions.

We want to learn relation between $\vec{x}$ and $y$ from training set $D = \{(\vec{x}^i, y^i), i = 1, \dots, n\}$

Statistical Learning $\approx$ Machine Learning only that Statistical learning is less focused on algorithms than ML.

**Definition 1.** A learning algorithm is a function that takes training set $D = \{(\vec{x}^i, y^i), i = 1, \dots, n\}$ as input and has as output a prediction $y = \hat{f}(\vec{x})$ that for every $\vec{x}$ predicts y value.

**Definition 2.** A probability model is a joint probability distribution $P$ of $\vec{x} \in X$ and $y \in C$ on the pair $(\vec{X}, Y) \in X \times C = \mathbb{R}^d \times C$ of a random vector $\vec{X} \subseteq \mathbb{R}^d$ and random variable $Y \in C$.

**Definition 3.** A function model is a single function $f : X \to C$ or a class $F$ of such functions where one function is assumed to give a good prediction $y \in C$ from $\vec{x} \in X$.

1. Supervised Learning (regression and classification)
2. Unsupervised Learning

**Remark.** One example of Supervised Learning is in generative models: $(y, \vec{x}) \to P(y|\vec{x})P(\vec{x})$ *note that this is just the chain rule in probability where P(x,y) = P(y|x) P(x)* where $P(y|\vec{x})$ is the conditional probability that a set of inputs $\vec{x}$ produces $y$ and $P(\vec{x})$ represents the marginal probability of the input features $\vec{x}$ are in a dataset (i.e. how likely different feature combinations are as each $x_i$ is a feature).

However, in discriminant models, the above does not apply as they focus solely on learning $P(y|\vec{x})$ the conditional probability.

One example of Unsupervised Learning is probablistic modeling: $(\vec{x}) \to P(\vec{(x)})$, so we are simply estimating the probability distribution of underlying data, using each datapoint $\vec{x}$ to improve our estimation of the probability distribution $P(\vec{x})$.

## 1.1 Supervised Learning

1. regression - $y \in R$ is continuous, quantitative
2. classification - $y \in 1, \ldots, K$ is discrete, qualitative

In a statistical learning (theoretical standpoint) they are the same but in a ML algorithmic standpoint they are different!

**Remark.** A probability model has fixed joint prob distribution: $(\vec{X}, Y) \sim P(\vec{x}, y)$. For some measurable set $A \subset \mathbb{R} \times C$, the probability that $(\vec{X}, Y) \in A$ is:

$$P(A) = P((\vec{X}, Y) \in A).$$

TODO: Missing stuff – parametric vs non-parametric

## 1.2 Regression Model Assumptions

$$y^{(i)} = h(\vec{x}^{(i)}) + \varepsilon_i.$$

Here $h(\vec{x})$ depends on some finite or infinite collection of parameters $\vec{\theta}$

*We assume that errors are random variables of the form $\varepsilon_i \sim$ Normal $(0, \sigma^2)$.*

**Remark.** This is a strong assumption!

*Proof.*

$$\begin{align} E(Y|\vec{X} = \vec{x}) &= E(h(\vec{X}) + \varepsilon|\vec{X} = \vec{x}) \text{ since } h(\vec{X}) \text{ and } \varepsilon \text{ are independent} \tag{1}\\ &= E(h(\vec{X})|\vec{X} = \vec{x}) + E(\varepsilon|\vec{X} = \vec{x}) \tag{2}\\ &= h(\vec{x}) + 0 \tag{3}\\ &= h(\vec{x}) \tag{4} \end{align}$$

□

**Remark.** Proof above is only true for the assumption that $\varepsilon$ is random white noise.

## 1.3 Expected Prediction Error

Assume we have an algorithm for estimating $h(\vec{x}^0)$ for test point $\vec{x}^0$. The estimator is:

$$\vec{y}^0 = \hat{h}(\vec{x}^0) \approx h(\vec{x}^0).$$

**Definition 4.** Our loss function is then defined as:

$$L(Y, h(\vec{X})).$$

**Loss Function Types**

- $L(Y, h(\vec{X})) = (Y - h(\vec{X}))^2$ square loss error
- $L(Y, h(\vec{X})) = |Y - h(\vec{X})|$ absolute loss error

**Definition 5.** Expected Prediction Error (Expected Value of Error)

$$EPE(h) = E_{Y,\vec{X}}[L(Y, h(\vec{X}))] = E_{\vec{X}} E_{Y|\vec{X}}[L(Y, h(\vec{X})|\vec{X}].$$

We want to minimize pointwise:

$$\hat{h}(\vec{x}) = argmin_c E_{Y|X}[L(Y, c)|\vec{X} \to \vec{x}].$$

where $c = h(\vec{X})$

**Remark.** In Theory we want prediction error for all future values but in practice we only have to the test error.

### 1.3.1 Squared Error Loss

Let $L(Y, h(\vec{X})) = (Y - h(\vec{X}))^2$ square loss error

$$\hat{h}(\vec{x}) = argmin_c E_{Y|\vec{X}}[(Y - c)^2|\vec{X} = \vec{x}].$$

$$= argmin_c E_{Y|\vec{X}}[Y^2 - 2cY + c^2|\vec{X} = \vec{x}].$$

Note:

$$E[Y^2 - 2cY + c^2|\vec{X} = \vec{x}] = E[Y]^2 - 2cE[Y] + c^2.$$

$$\frac{d}{dc}[E[Y]^2 - 2cE[Y] + c^2] = -2E[Y] + 2c = 0.$$

$$c = E[Y] = E[Y|\vec{X} = \vec{x}].$$

Logically this makes sense because the minimizer of the squared loss function when $h(\hat{x}) = E[Y|\vec{X} = \vec{x}]$ is the mean (expected value) of Y given X = x which captures the squared loss function's central tendency as it squares distance.

### 1.3.2 Absolute Error

Let $L(Y, h(\vec{X})) = |Y - h(\vec{X})|$ absolute loss error:

$$\hat{h}(\vec{x}) = argmin_c E_{Y|\vec{X}}[|Y - c||\vec{X} = \vec{x}].$$

Expand the $|Y - c|$ piecewise:

$$E[|Y - c|] = \int_{-\infty}^{c} (c - y)p(y)dy + \int_{c}^{\infty} (y - c)p(y)dy.$$

Take the derivative to minimize the expected value:

$$\frac{d}{dc}(E[|Y - c|]) = \frac{d}{dc}[\int_{-\infty}^{c} (c - y)p(y)dy + \int_{c}^{\infty} (y - c)p(y)dy] = 0.$$

$$= \int_{-\infty}^{c} p(y)dy - \int_{c}^{\infty} p(y)dy = 0.$$

$$\int_{-\infty}^{c} p(y)dy = \int_{c}^{\infty} p(y)dy.$$

**This implies that to minimize the Expected Value, c is the median of the distribution of Y, because the cumulative probability mass to the left of c must be equal to the cumulative probability mass to the right of c.**

$$\hat{h}(\vec{x}) = median(Y|\vec{X} = \vec{x}).$$

**Remark.** Absolute Value is not differentiable (although it is continuous) so greedy descent methods can't be used.

### 1.4 Classification

Consider the binary case where $Y = 0, 1$ for class 1 and class 2 respectively:

$$\begin{aligned}\hat{h}(\vec{x}) &= E[Y|\vec{X} = \vec{x}] \\ &= P(\text{class } 1|\vec{X} = \vec{x}) \cdot 1 + P(\text{class2}|\vec{X} = \vec{x}) \cdot 0 \\ &= P(Y = 1|\vec{X} = \vec{x}) \cdot 1 + P(Y = 0|\vec{X} = \vec{x}) \cdot 0 \\ &= P(Y = 1|\vec{X} = \vec{x}) \in \mathbb{R}.\end{aligned}$$

In the multiclass case: $Y = 1, \dots, K$:

$$\begin{aligned}\hat{\vec{x}} &= \arg\min_{f(x)} E_{Y|\vec{X}}[L(Y, h(X))|\vec{X} = \vec{x}] \\ &= \arg\min_{k} E_{Y|\vec{X}}[L(Y, k)|\vec{X} = \vec{x}] \\ &= \arg\min_{k} \sum_{y=1}^{K} L(y, k)P(Y = y|\vec{X} = \vec{x}).\end{aligned}$$

TODO classifiers stuff

## 2 Bias-Variance Tradeoff

Assuming, iid $\varepsilon_i$ as noise:

**Definition 6.** The Expected Prediction (in practice test) Error (EPE) for a fixed test point $\vec{x}^0$ is the average over both $y^0$ and the entire training set $D$.

$$EPE(\vec{x}^0) = E_{y^0,D}[L(y^0, \hat{h}(\vec{x}^0))].$$

We want in theory **full expected prediction error**, meaning we average over all training sets: $D = \{(\vec{x}^i, \vec{y}^i, i = 1, \dots, n\}$ where each $D_i$ is one possible split of train/test data and over all possible values $Y = y^0$ at $\vec{x}^0$.

**Remark.** D is a random variable whose distribution is all possible train/test splits. $y^0$ is a random variable representing the true value of the outcome at $\vec{x}^0$ but inherents the random "noise" in the data generation process, leading to different $\hat{y}^0$ being observed.

## 2.1 Mathematical Decomposition of Expected Prediction Error at fixed $\vec{x}^{(0)}$

**Note.** This is for the case where the loss function is the **Mean Squared Error** loss function.

$$\begin{aligned} EPE(\vec{x}^0) &= E_{y^0,D}[(y^0 - \hat{y}^0)^2] \\ &= E_{y^0} E_D[(y^0 - \hat{y}^0)^2] \end{aligned}$$

$$\begin{aligned} &= E_{y^0} E_D\{[(y^0 - E_{y^0}[y^0]) + (E_{y^0}[y^0] - E_D[\hat{y}^0]) + (E_D[\hat{y}^0] - \hat{y}^0)]^2\}. \\ &= E_{y^0}[y^0 - E_{y^0}[y^0]]^2 + [E_{y^0}[y^0] - E_D[\hat{y}^0]]^2 + E_D[E_D[\hat{y}^0] - \hat{y}^0]^2. \end{aligned}$$

where $y_0 = h(\vec{x}^0) + \varepsilon$ and $\hat{y}^0 = \hat{h}\,(\vec{x})$

**Note.** Note all the cross terms dissappear from the expression about which is why we can eliminate them:

$E_{y^0} E_D[(Y^0 - E_{y^0}[y^0])\left(E_D[\hat{y}^0] - \hat{y}^0\right)]$ the first expression does not depend on D

$= E_{y^0}[(y^0 - E_{y^0}[y^0])\, E_D\left(E_D[\hat{y}^0] - \hat{y}^0\right)]$

$= 0.$

The last step is possible because $y^0, \hat{y}^0$ are independent since $\varepsilon_i$ are independent and $\hat{y}^0$ depends on D, and by probability $E[X \cdot Y] = E[X] \cdot E[Y]$ when $X, Y$ are independent.

1. $(y^0 - E_{y^0}[y^0])$ – can be thought of as the **noise** of the model that distorts the true value of y from its expected value (coming from the $\varepsilon$ assumed to be random gaussian noise)
2. $E_{y^0}[y^0] - E_D[\hat{y}^0]$ – can be thought of as the **bias**, difference between true expected output $E_{y^0}[y^0]$ and the model's prediction $E_D[\hat{y}^0]$.
3. $(E_D[\hat{y}^0] - \hat{y}^0)$ – can be thought of as **variance**, variability of model's prediction $\hat{y}^0$ from the actual expected value $E_D[\hat{y}^0]$ (recall that D is the set of all possible training sets from different train/test splits where our model takes only train set).

We can rewrite the formula since $var = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$:

$$\begin{aligned} &= var_{y^0}(y^0) + \left[E_{y^0}[y^0] - E_D[\hat{y}^0]\right]^2 + var_D(\hat{y}^0) \\ &= \sigma^2 + \left[h(x^0) - E_D[\hat{y}^0]\right]^2 + var_D(\hat{y}^0) \quad \text{from } N(\mu = h(x^0), \sigma) \\ &= \sigma^2 + [bias]^2\,(\hat{y}^0) + var_D(\hat{y}^0) \\ &= \text{unavoidable error } + \text{bias}^2 + var_D(\hat{y}^0) \end{aligned}$$

where bias is difference between true and predictec

TODO knn and LINEAR model notes

## 3 KNN

## 4 Linear Regression - OLS

## 5 Ridge Regression

Motivation: no longer working with unbiased estimators, trade off some biass to large decrease in variance.

**Definition 7.** Assume data $D = (X, \vec{y})$ is centered, the mean $E(X) = \vec{0}$ and $E(\vec{y}) = 0$, so

$$h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_1 x_1 + \ldots + \theta_d x_d.$$

**Ridge Regression** minimize cost function:

$$J(\vec{\theta}) = \sum_{i=1}^{n} (y^i - h_\theta(\vec{x}^i))^2 + \lambda \sum_{j=1}^{d} \theta_j^2 \quad = (X\vec{\theta} - \vec{y})^T (X\vec{\theta} - \vec{y}) + \lambda \vec{\theta}^T \vec{\theta}.$$

Taking the first derivative:

$$\begin{aligned} \frac{d}{d\vec{\theta}} J &= \ldots \\ &= \ldots. \end{aligned}$$

$$\vec{\theta} = (X^T X + \lambda I)^{-1} X^T \vec{y}.$$

TODO More examples:   TODO High Dimension Data and Reduction

## 6 Readings

1. https://mlu-explain.github.io/bias-variance/