02-Linear Regression (Formulation)

Formulating a supervised learning task

- Variables related to the task can be separated into two parts
- $\boldsymbol{x}$ is the input, always known (during training and inference)
  - From now on, we assume $\boldsymbol{x}$ is a feature vector

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d, \quad d \text{ is the dimension}$$

  - If a feature is real-valued, then it is just the value
  - If a feature is categorical, then we may use one-hot representation. For example, if we would like to indicate if an animal is a cat, a dog, or an elephant, then we can use three binary indicators (although only one is one) to represent such information

whether a cat? → x_1
a dog? → x_2
an elephant? → x_3

$x_i \in \{0, 1\}$

  - Different features can be concatenated as a longer vector

- t is the target/label/output, which is of particular interest
  Note: knowing $\boldsymbol{x}$ predicting $t \neq$ knowing t predicting $\boldsymbol{x}$
  - In classification, t is a categorical variable
  - In regression, t is a real value, i.e.,

$$t \in \mathbb{R}$$

- For training, the machine learning model is given a set of data examples/samples in the form of $(\boldsymbol{x}, t)$.

  We name ith data sample $(\boldsymbol{x}^{(i)}, t^{(i)})$.

  Suppose we observe M samples in total for training, the training set is

$$\mathcal{D}_{train} = \{(\boldsymbol{x}^{(i)}, t^{(i)})\}_{i=1}^{M}$$

- During inference, the model is asked to predict the label $\hat{t}_*$ of an unseen input $\boldsymbol{x}_*$
  We may be given a set of unseen samples, and they are called a test set.

Formulating a supervised machine learning system

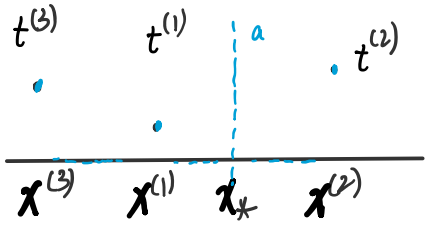
- Humans define a set of functions mapping from input to output, from which a machine learning system can pick up a particular function. This is known as a hypothesis class $\mathcal{H}$.

Usually, $\mathcal{H} \subsetneq \{f : X \to Y\}$

Note: If we allow to learn from, or have no preference on, all functions, then learning is infeasible.

Suppose $\boldsymbol{x}_* \notin \mathcal{D}_X$, consider a set of functions $\{f_a(\boldsymbol{x}) : a \in \mathbb{R}\}$, st.

$$f_a(\boldsymbol{x}) = \begin{cases} y^{(i)}, & \text{if } \boldsymbol{x} = \boldsymbol{x}^{(i)} \\ a, & \text{if } \boldsymbol{x} = \boldsymbol{x}_* \\ 0, & \text{otherwise} \end{cases}$$



Since we only observe $\{(\boldsymbol{x}^{(i)}, t^{(i)})\}_{i=1}^{M}$
and f_a performs equally well on the training set,
We have no preference on different values of a
The inference could be arbitrary.

Thus, we have to express our preference (hard and/or soft) on all possible functions, based on previous experience or simply for aesthetic purposes. Such preference comes with the machine learning system before training, and is called inductive bias.

It is also noted that the goal of machine learning systems is not to predict well on seen examples $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots$, but rather on unseen examples $\boldsymbol{x}_*$

- In the training/learning phase, the learning machine picks a particular function according to a certain criterion J, i.e., $h \in \mathcal{H}$

$$h = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \; J(h, \mathcal{D}_{\text{train}})$$

J is sometimes known as a loss, a cost, or a training objective.

"Statisticians seem to be pessimistic creatures who think in terms of losses. Decision theorists in economics and business talk instead in terms of gains (utility)."
--- James O. Berger (1985). *Statistical Decision Theory and Bayesian Analysis*.

- In the inference/prediction phase, the learning machine uses h to predict unseen samples $\boldsymbol{x}_*$, i.e.,

  $$\hat{t}_* = h(\boldsymbol{x}_*)$$

  In machine learning research, the model will be evaluated by a measure of success, which we call error

  $$E(\underbrace{h(x_*)}_{\hat{t}_*}, t_*) \in \mathbb{R}$$

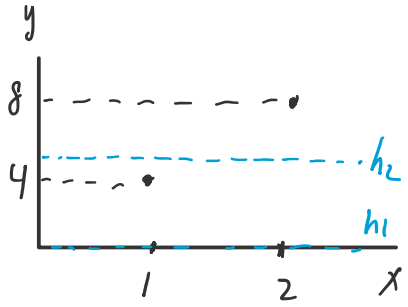
$$J(h) = \sum_{n=1}^{2} \left| t^{(i)} - h(x^{(i)}) \right|^2$$

First toy example (finite hypothesis class)

$x \in \mathbb{R}$, $t \in \mathbb{R}$ $\quad \mathcal{D}_{\text{train}}$: $\{(1,4), (2,8)\}$ $\quad (x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)})$

Machine Learning System 1:

- $\mathcal{H} = \{h_1, h_2\}$
  where $h_1(x) \equiv 0$, $h_2(x) \equiv 5$
- $J(h) = \sum_{m=1}^{2} \left| t^{(m)} - h(x^{(m)}) \right|^2$
- Optimization algorithm: by enumeration



Training:

$J(h_1) = 4^2 + 8^2 = 80$
$J(h_2) = 1^2 + 3^2 = 10$
$\Rightarrow$ pick $h = h_2$

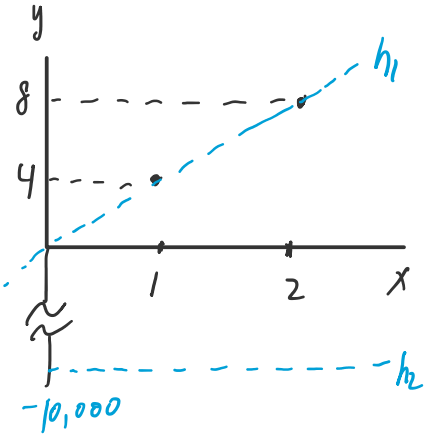
Inference: $\forall x_*$, the predicted $\hat{t}_* = h_2(x_*) \equiv 5$

Machine Learning System 2

- $\mathcal{H} = \{h_1, h_2\}$

  $h_1(x) = 4x$

  $h_2(x) = -10{,}000$



- $J(h) = \sum_{n=1}^{2} |h(x^{(n)}) + 10{,}000|^4$
- Optimization algorithm: by enumeration

Training: $J(h_1)$: very large, $J(h_2) = 0$ $\Rightarrow$ pick $h = h_2$

Inference:

$\forall x_*$, the predicted label $\hat{t}_* = h_2(x) \equiv -10{,}000$

Measure of Success:

Given test samples $x_*^{(1)}, t_*^{(1)} = (1.2, 4.5)$

$x_*^{(2)}, t_*^{(2)} = (1.7, 7)$

$x_*^{(3)}, t_*^{(3)} = (2.2, 8.5)$

The measure of success (error) is defined to be

$$E = \sum_{i=1}^{3} |h(x_*^{(i)}) - t_*^{(i)}|$$

For Machine learning system 1: $E = 5.5$

For Machine learning system 2: $E = \cdots$ (very large)

Conclusion: Machine learning system 1 wins

Lessons learned:

- A machine learning system involves many aspects, including the hypothesis class, the training objective function, and the optimization algorithm
- A machine learning model has no control on the test data and the measure of success
  - In particular, the measure of success could be different from the training objective

The Linear Regression Model

Hypothesis class

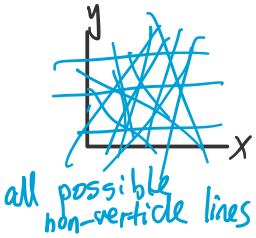
Let input $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d$, output $t \in \mathbb{R}$

The hypothesis class is the affine functions on $\boldsymbol{x}$ predicting t in the form of

$$h(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b = \sum_{i=1}^{d} w_d x_d + b$$

where $\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are model parameters

called weights called bias



By linear algebra definitions, this can be denoted as

$$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

The hypothesis class $\mathcal{H} = \{ h(\boldsymbol{x}) : h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b, \ \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R} \}$

Note: linear vs affine transformation

Linear transformation takes the form of $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} = w_1 x_1 + \cdots + w_d x_d$

Affine transformation allows the bias term $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$

Intuitively, the bias term in linear regression learns an offset/baseline of the task.

Why "linear" regression?

- The simplest model
- We have to make assumptions (inductive bias)
- Non-linear problems can be transformed into a linear problem (usually in a higher dimensional space
  - Human feature engineering
  - Kernel methods (implicit high dimensional space defined by inner-product)
  - Neural networks (composite non-linear functions)

Augmented feature

We define a trivial feature $x_0 = 1$ for all data points

We define the augmented feature representation

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} x_0 \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

Then, the linear regression model predicts

$$h(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$$

where $\tilde{\boldsymbol{w}} := \begin{pmatrix} b \\ \boldsymbol{w} \end{pmatrix} = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_d \end{pmatrix} \in \mathbb{R}^{d+1}$

We even drop "~" for simplicity.

Training objective of linear regression

- Suppose the parameters of a linear regression model is $\boldsymbol{w}$

  [feature augmentation is implicit]

- We often define the loss as the square error
  For a training sample $(\boldsymbol{x}^{(m)}, t^{(m)})$

  The per-sample loss is $J^{(m)} = \frac{1}{2}\left(h(\boldsymbol{x}^{(m)}) - t^{(m)}\right)^2$

  $$= \frac{1}{2}\left(\boldsymbol{w}^T \boldsymbol{x}^{(m)} - t^{(m)}\right)^2$$

  Then, the overall loss could be the average of per-sample loss

  $$J = \frac{1}{M}\sum_{m=1}^{M} J^{(m)} = \frac{1}{2M}\sum_{m=1}^{M}\left(\boldsymbol{w}^T \boldsymbol{x}^{(m)} - t^{(m)}\right)^2$$

  Remarks:
  - The square loss has profound theoretical justifications and we will explore some in future lectures. At this moment, we can intuitively think of it such that square error penalizes more on bad predictions
  - 1/2 is a constant having no effect in the optimization objective. It's included here mainly for aesthetic purposes (simplifying the derivative)
  - Likewise, 1/M has not effect either, but in numerical optimization, including 1/M makes the model more stable with different sizes of data
  - The model is also called ordinary least square (OLS).

Matrix representations

- The input can be represented as a matrix

$$\mathbf{X} = \begin{bmatrix} — (\boldsymbol{x}^{(1)})^T — \\ — (\boldsymbol{x}^{(2)})^T — \\ \vdots \\ — (\boldsymbol{x}^{(M)})^T — \end{bmatrix}_{M \times (d+1)}$$

- The predicted targets are $\underset{M\times 1}{\boldsymbol{y}} = \underset{M\times(d+1)}{\mathbf{X}}\ \underset{(d+1)\times 1}{\boldsymbol{w}}$

  (also denoted by $\hat{\boldsymbol{t}}$)

- The loss is

$$J = \frac{1}{2M} \| \boldsymbol{y} - \boldsymbol{t} \|_2^2$$

where $\| \cdot \|_2$ is the l_2-norm of a vector

For $\boldsymbol{v} \in \mathbb{R}^d$

$$\| \boldsymbol{v} \|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_d^2}$$

Or in general

$$\| \boldsymbol{v} \|_p = \left(\sum_{i=1}^{d} v_i^p \right)^{1/p}$$