Machine learning - Flow of machine learning

Yonghoon Dong

March 8, 2024

What is machine learning?

A popular definition of **machine learning** or **ML**, due to Tom Mitchell, is as follows
"A computer program is said to learn from experience E with respect to some class of task T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Types of machine learning

1. Supervised learning : learn the relationship between inputs and outputs to predict outputs for new inputs.
2. Unsupervised learning : the model learns from input data alone to discover hidden structures or patterns
3. Reinforcement learning : emphasizes learning to achieve a clear objective through feedback in dynamic environments

Supervised learning

1. The most common form of ML
2. the task T is to learn a mapping f from input x to outputs y
3. x is called **features**, or **input**
4. y is called **label, target**, or **output**
5. The experience E is given in the form of a set of N input-output pairs $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$, known as the **training set**

Performance evaluation: Empirical risk

How can we evaluate the performance of our model?

1. Define the **loss function**
$$\ell(y_n, f(x_n; \theta))$$
2. Define **empirical risk** to be the average loss of the predictor on the training set
$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta))$$

A loss function should be defined so that a smaller value indicates a better model.

Model fitting / Training

One way to define the problem of **model fitting** or **training** is to find parameters that minimizes the empirical risk on the training set.

$$\begin{aligned}\hat{\theta} &= \arg\min_{\theta} \mathcal{L}(\theta) \\ &= \arg\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n; \theta))\end{aligned}$$

This is called **empirical risk minimization**.

Performance evaluation in supervised learning

Q: Then how can we define a loss function $\ell(y_n, f(x_n; \theta))$?

A: It depends on **the type of output** we are predicting

Note that we will see a lot of loss functions but this is not the only way to define them.

Performance evaluation in supervised learning

Q: Then how can we define a loss function $\ell(y_n, f(x_n; \theta))$?
A: It depends on **the type of output** we are predicting

Note that we will see a lot of loss functions but this is not the only way to define them.

Performance evaluation in supervised learning

Q: Then how can we define a loss function $\ell(y_n, f(x_n; \theta))$?
A: It depends on **the type of output** we are predicting

Note that we will see a lot of loss functions but this is not the only way to define them.

Types of supervised learning

- Classification : the goal is to predict which **category** or **class** a new observation belongs to, based on its features or attributes. The output variable in classification is discrete and categorical.
- Regression : the goal is to predict a **continuous numerical value** or a **quantity**. The output variable in regression is continuous, and the model learns to map input features to a target variable that can take any value within a given range.

Classification

Definition (Classifiction)

the output space is a set of C unordered and mutually exclusive labels known as **classes**, $\mathcal{Y} = \{1, 2, \ldots, C\}$. The problem of predicting the class label given an input is also called **pattern recognition**. (If there are just two classes, it is called **binary classification**.)

Classification

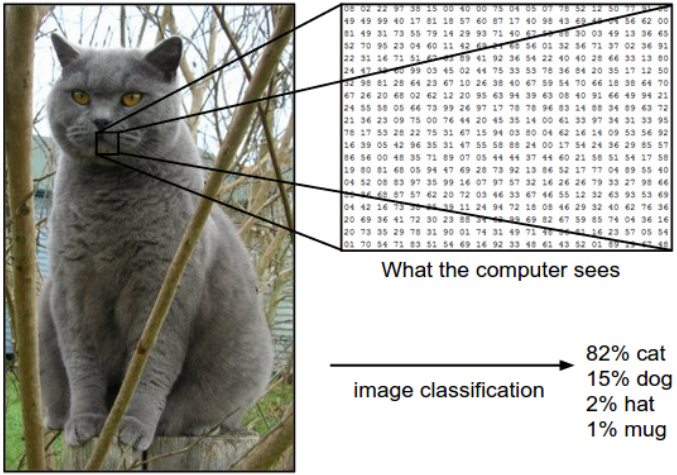

Figure: Illustration of the image classification problem.

Uncertainty in classification problem

In many cases, we will not be able to perfectly predict the exact output given the input due to

1. lack of knowledge of the input-output mapping: model uncertainty
2. intrinsic stochasticity in the mapping: data uncertainty

We can capture our uncertainty using the conditional probability distribution

$$p(y = c|x, \theta) = f_c(x; \theta)$$

where $f : \mathcal{X} \to [0, 1]^C$ maps inputs to a probability distribution over the C possible output labels.

Uncertainty in classification problem

However, f may not satisfy the properties of the probability. To avoid this restriction, we can use the **softmax function**, which is defined as follows

$$\text{softmax}(a) = \left[\frac{e^{a_1}}{\sum_{c'=1}^{C} e^{a_{c'}}} \quad \cdots \frac{e^{a_C}}{\sum_{c'=1}^{C} e^{a_{c'}}} \right]$$

The input of the softmax $a = f(x; \theta)$ are called **logits**. We thus define the overall model as follows

$$p(y = c|x, \theta) = \text{softmax}(f_c(x; \theta))$$

If the problem is binary classification, we can use sigmoid instead of the softmax function. We will discuss the difference between sigmoid and softmax funciton deeply in the next chapter.

Uncertainty in classification problem

When we consider the uncertainty, it is common to use the negative log probability as our loss function:

$$\ell(y_n, f(x_n, \theta)) = -\log p(y_n|x_n, \theta)$$

This concept is related to the Maximum likelihood estimation which will discuss later.

Softmax function vs Sigmoid function

These functions can be used to generate probabilities, but they differ in their applications and mathematical formulations.

1. Softmax function : commonly used in the output layer of neural networks for multi-class classification problems. Similarly, this function is used when we define the loss function of categorical classification problems.
2. Sigmoid (logistic) function : commonly used as a activation function for non-linearity. However, it can be used in the output layer of neural networks for binary classification problems. Similarly, this function is used when we define the loss function of binary classification problems.

Predicting new data

Then how can we predict the new data after training? (i.e. In this case θ is fixed)

$$c^* = \arg\max_c p(y = c|x, \theta)$$

where θ is fixed

Examples of loss function in classification problem

We can define a lot of loss function for a classification problem.

1. Cross-Entropy loss (Check the following website : Link)
   - binary case: binary cross-entropy loss

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log\Big(p(y_i=1|x_i,\theta)\Big) + (1-y_i)\log\Big(1-p(y_i=1|x_i,\theta))\Big)$$

   - Multi-class case: categorical cross-entropy loss

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log\Big(p(y_i=j|x_i;\theta)\Big)$$

where $y_{ij}=1$ when $y_i=j$, $y_{ij}=0$ otherwise. (We can interpret this as a cross entropy with empirical distribution)

Examples of loss function in classification problem

2. Hinge Loss (a.k.a. SVM loss)
   - binary case loss

$$\mathcal{L}(w, b) = \frac{1}{N} \sum_{i=1}^{N} \max\{0, \Delta - y_i(wx_i + b)\}$$

   - Multi-class case

$$\mathcal{L}(w, b) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max\{0, w_j^T x_i - w_{y_i}^T x_i + \Delta)\}$$

It is set up so that the SVM "wants" the correct class for each image to a have a score higher than the incorrect classes by some fixed margin Δ Check the following website : ▸ Link

Regression

Definition (Regression)

We want to predict a **real-valued** quantity $y \in \mathbb{R}$ instead of a class label. This is known as **regression**.

Regression

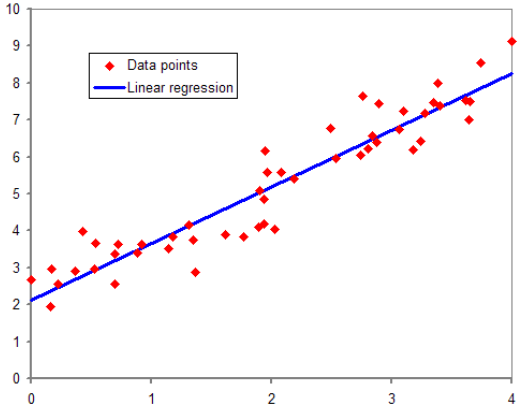

Figure: Illustration of the regression problem.

Performance evaluation: mean squared error (a.k.a. MSE)

We can define a loss function (Note that this is **not** the only way to define a loss function) as

$$\ell(y_n, f(x_n; \theta)) = (y_n - f(x_n; \theta))^2$$

We can interpret $f(x_n; \theta)$ as a prediction of y_n. In many context, it is written as $\hat{y}_n$. Therefore, the empirical risk is defined as

$$\mathcal{L}(\theta) = \text{MSE}(\theta) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n; \theta))^2$$

Actually, this is related to the assumption that y is generated by the Gaussian $\mathcal{N}(f(x, \theta), \sigma^2)$. Therefore, when we use MSE loss function, **we already consider the uncertainty** in our prediction. This can be proved by using the Maximum likelihood estimation. We will prove this at chapter 2.

Overfitting/Underfitting problem in Supervised learning

We can rewrite the empirical risk in the following equivalent way

$$\mathcal{L}(\theta; \mathcal{D}_{\text{train}}) = \frac{1}{|D_{\text{train}}|} \sum_{(x,y) \in D_{\text{train}}} \ell(y, f(x, \theta))$$

1. It is possible that our model is just memorizing the correct output for each input. A model that perfectly fits the training data, but which is too complex, is said to suffer from **overfitting**.
2. If a model is too simple so that our model can't predict the training data, it is said to suffer from **underfitting**.

Overfitting/Underfitting problem in Supervised learning

To detect wheter our model is overfitting or not, let us assume that we have access to the true distribution $p^*(x, y)$ used to generated the training set. Then we can compute the theoretical expected loss or **population risk**

$$\mathcal{L}(\theta; p^*) = \mathbb{E}_{p^*}[\ell(y, f(x; \theta))]$$

The difference $\mathcal{L}(\theta; p^*) - \mathcal{L}(\theta; \mathcal{D}_{\text{train}})$ is called the **generalization gap**. If a model has a large generalization gap, it is a sign that it is overfitting.

Overfitting/Underfitting problem in Supervised learning

The problem is we don't know p^*. Instead, we partition the data into two subsets, known as the training set and the **test set**. Then we can approximate the population risk using the **test risk**

$$\mathcal{L}(\theta; \mathcal{D}_{\text{test}}) = \frac{1}{|D_{\text{test}}|} \sum_{(x,y) \in D_{\text{train}}} \ell(y, f(x, \theta))$$

Overfitting/Underfitting problem in Supervised learning

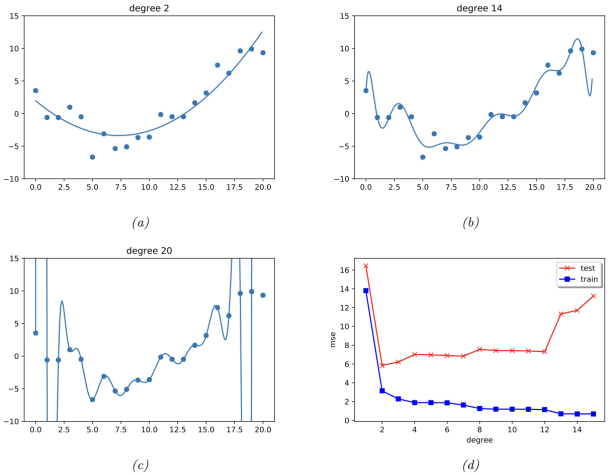

Figure: Illustration of overfitting and underfitting problem

Unsupervised learning

1. The goal is to extract meaningful patterns, structures, or representations from the data.
2. Unlike the supervised learning, there is **no** corresponding outputs y_n
3. The objective **may** be to learn certain aspects or properties of the data distribution that are relevant to the task at hand, such as clustering, dimensionality reduction, or density estimation. (Not always)
4. Avoids the need to learn how to partition the world into often arbitrary categories.
5. Forces the model to **explain** the high-dimensional inputs, rather than just the low-dimensional outputs

Types of unsupervised learning

- Clustering : partition the input into regions that contain similar points
- Discovering latent "factors of variation" : reduce the dimensionality by projecting it to a lower dimensional subspace which captures the essence of the data
- Self-supervised learning : create proxy supervised tasks from unlabeled data

Performance evaluation in unsupervised learning

It is very hard to evaluate the quality of the output of an unsupervised learning method, because there is no ground truth. A common method for evaluating unsupervised models is to measure the probability assigned by the model to unseen test examples.

$$\mathcal{L}(\theta; \mathcal{D}_{\text{unseen}}) = -\frac{1}{|\mathcal{D}_{\text{unseen}}|} \sum_{x \in \mathcal{D}_{\text{unseen}}} \log p(x|\theta)$$

This treats the problem of unsupervised learning as one of **density estimation**.

Performance evaluation in unsupervised learning

Unfortunately, density estimation is difficult, especially in high dimensions.

1. Instead of directly evaluating the unsupervised model's performance on a specific task, the learned unsupervised representation **can be used as features or input to a downstream supervised learning method**.
2. If the downstream task performs well with the unsupervised features, it suggests that the unsupervised model has successfully captured useful information.

By doing so, we can increase the **sample efficiency** of learning (i.e. reduce the number of labeled examples needed to get good performance)

Reinforcement learning

Definition (Reinforcement learning)

Basic reinforcement learning is modeled as a Markov decision process

1. a set of environment and agent states, $\mathcal{S}$
2. a set of actions, $\mathcal{A}$, of the agent
3. $p_a(s, s') = Pr(S_{t+1} = s' | S_t = s, A_t = a)$, the probability of transition at time t from state s to state s' under action a
4. $r_a(s, s')$, the immediate reward after transition from s to s' with action a

The purpose of reinforcement learning is for the agent to learn an optimal, or nearly-optimal, policy that maximizes the "reward function"

Reinforcement learning

1. Reinforcement learning has grown in popularity recently, due to its broad applicability
2. It can be harder to make reinforcement learning work than it is for supervised or unsupervised learning because it may be unclear to define states and rewards
3. To compensate it, it is common to use other information sources like expert demonstrations.

No free lunch theorem

- Given the large variety of models in the literature, it is natural to wonder which one is best.
- Unfortunately, there is **no** single best model that works optimally for all kinds of problems; this is sometimes called the **no free lunch theorem**.

The best way to pick a suitable model is based on **domain knowledge**, and/or **trial and error** or **Bayesian methods**.

Bibliography

1. https://hyunw.kim/blog/2017/10/14/Entropy.html
2. https://m.blog.naver.com/wooy0ng/222666100291
3. Probabilistic Machine Learning: An introduction, Kevin P. Murphy