Deep Generative Models

Chapter 2: Data Generation Problem

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Summer 2025

General Learning Problem

Using the notion of data distribution, we could say

Generic Learning Problem *including Data Generation*

In a learning problem, we aim to learn the data distribution

+ *It does not sound like what we used to do in other courses!*
  - ↳ *In classification, we find a classifier*
  - ↳ *In regression, we learn a regression model*
- Well, if we look with statistical glasses: *it does!*

Looking Back: *Supervised Learning*

In supervised learning, we have labeled data: *we have a dataset*

$$\mathbb{D} = \{(x_i, y_i) \in \mathbb{X} : i = 1, \dots, n\}$$

computational *We want to learn a model* $f_{\mathbf{w}}(\cdot)$

$$y \approx f_{\mathbf{w}}(x)$$

which matches the dataset $\mathbb{D}$

---

statistical *We have a dataset sampled from data distribution* $P(x, y)$

$$\mathbb{D} = \{(x_i, y_i) \in \mathbb{X} : i = 1, \dots, n\}$$

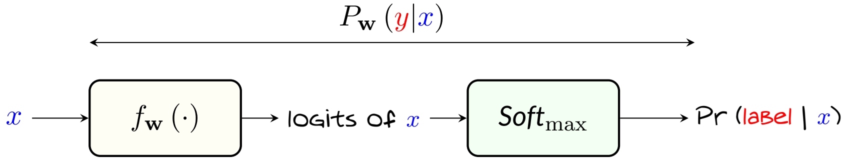
We want to learn $P_{\mathbf{w}}(y|x)$ *which well-approximates* data distribution, *i.e.,*

$$P_{\mathbf{w}}(y|x) \approx P(y|x) = \frac{P(x, y)}{P(x)}$$

Supervised Learning: *Classification*

+ *But, I cannot remember of learning $P_{\mathbf{w}}(y|x)$!*
– Sure, we did!

In classification, we literally learn $P_{\mathbf{w}}(y|x)$: we learn a classifier $f_{\mathbf{w}}(x)$ with a Soft$_{\max}$ at the output



Classification Model

A classification model describes the conditional data distribution $P_{\mathbf{w}}(y|x)$

Supervised Learning: *Regression*

+ *What about regression then?!*

In regression, we learn consider a hypothesis on the model and write

$$\hat{y} = f_{\mathbf{w}}(x)$$

which is a good approximation of true label

$$\hat{y} \approx y$$

We find $\mathbf{w}$ *by minimizing empirical risk* $\hat{R}(\mathbf{w})$*: say we use classical squared error*

$$\hat{R}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (f_{\mathbf{w}}(x_i) - y_i)^2$$

Supervised Learning: *Regression*

The hypothesis in fact assumes *distribution $P_{\mathbf{w}}(y|x)$ for data, where*

$$P_{\mathbf{w}}(y|x) \equiv \mathcal{N}\left(f_{\mathbf{w}}(x), \sigma^2\right)$$

for some constant variance σ^2

+ *How does it come?*
– Well, let's learn $P_{\mathbf{w}}(y|x)$ via maximum-likelihood

The likelihood of the model on the dataset $\mathbb{D}$ is

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{n} P_{\mathbf{w}}(y_i|x_i) = \frac{1}{(2\pi\sigma^2)^{n/2}} \prod_{i=1}^{n} \exp\left\{-\frac{(f_{\mathbf{w}}(x_i) - y_i)^2}{2\sigma^2}\right\}$$

$$= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(f_{\mathbf{w}}(x_i) - y_i)^2\right\}$$

Supervised Learning: *Regression*

We can work with log-likelihood instead, which is

$$\log \mathcal{L}(\mathbf{w}) = -\frac{n}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(f_{\mathbf{w}}(x_i) - y_i\right)^2$$

Comparing it to empirical risk, we can say

$$\log \mathcal{L}(\mathbf{w}) = -\frac{n}{2}\log 2\pi\sigma^2 - \frac{n}{2\sigma^2}\hat{R}(\mathbf{w})$$

This means that

maximizing likelihood with $P_{\mathbf{w}}(y|x) \equiv$ *minimizing empirical risk with* $f_{\mathbf{w}}(x_i)$

Moral of Story

In regression, we learn the density $P_{\mathbf{w}}(y|x)$ *from the dataset*

Discriminative Learning

Summary

In supervised learning, we learn data distribution from the dataset

- ↳ *The hypothesized model describes a parameterized distribution*
- ↳ *We use maximum-likelihood to fit the model to the dataset*

One key observation is that *we learn* $P(y|x)$ *in this case*

we do discriminative learning $\leftrightsquigarrow$ $P_{\mathbf{w}}(y|x)$ *is a discriminative model*

Discriminative Model

Discriminative models describe the conditional label distribution $P(y|x)$

Looking Back: *Unsupervised Learning*

+ *Was it also the case in unsupervised learning that we learn distribution?*
– Yes! And indeed, *we learn generation in this case!*

In unsupervised learning: *we have unlabeled data*

$$\mathbb{D} = \{x_i : i = 1, \ldots, n\}$$

computational *We want to learn a model $f_{\mathbf{w}}(\cdot)$, e.g.,*

$$\textit{Cluster}(x) \leftarrow f_{\mathbf{w}}(x)$$

which captures the pattern in dataset $\mathbb{D}$

---

statistical *We have a dataset sampled from data distribution $P(x)$ and want to learn $P_{\mathbf{w}}(x)$ which well-approximates data distribution, i.e.,*

$$P_{\mathbf{w}}(x) \approx P(x)$$

Unsupervised Learning: *Clustering*

A famous example is clustering: *k-means clustering finds cluster of a data sample x by comparing it against k learnable centroids $\mu_1, \dots, \mu_k$*

↳ *The cluster of x is the one whose centroid is closest*

$$\text{Cluster}\left(x\right) = \operatorname*{argmin}_{i=1,\dots,k} \left(x - \mu_i\right)^2$$

We learn $\mu_1, \dots, \mu_k$ by minimizing the sum-distance over the dataset

---

statistical *k-means clustering assumes that samples are drawn from one of the k normal distributions whose means are learnable*, i.e.,

$$P_{\mu_i}\left(x\right) \equiv \mathcal{N}\left(\mu_i, \sigma^2\right)$$

for fixed variance $\sigma^2 \rightsquigarrow$ the cluster specifies the distribution x comes from

Unsupervised Learning: *Maximum-Likelihood Clustering*

+ *How does it come?*
- Well! Again think about maximum likelihood estimation

A sample x, belongs to distribution which returns maximum likelihood

$$\begin{aligned}\textit{Cluster}\,(x) &= \underset{i=1,\ldots,k}{\operatorname{argmax}}\; P_{\mu_i}(x) \\ &= \underset{i=1,\ldots,k}{\operatorname{argmax}}\; \log P_{\mu_i}(x) \\ &= \underset{i=1,\ldots,k}{\operatorname{argmax}}\; -\frac{1}{2}\log 2\pi\sigma^2 - \frac{(x-\mu_i)^2}{2\sigma^2} \\ &= \underset{i=1,\ldots,k}{\operatorname{argmin}}\; (x-\mu_i)^2\end{aligned}$$

Bingo! k-means clustering does maximum likelihood!

Generative Learning

Summary

In unsupervised learning, we again learn data distribution from the dataset

↳ *We hypothesize a parameterized model for distribution*

↳ *We use maximum-likelihood to fit the model*

And interestingly, we learn $P(x)$

we do generative learning $\leftrightsquigarrow$ $P_{\mathbf{w}}(x)$ *is a generative model*

Generative Model

Generative models describe the complete data distribution

↳ *If we have no label, it's simply* $P(x)$

↳ *If we have label, it's a joint distribution* $P(x, y)$

Generative Learning: *More Generic Framework*

Generative learning describes a more generic framework

+ *How can we use it if we want to learn label of a data sample?*
- Well, we can use Bayes rule!

Say we have a supervised problem with dataset

$$\mathbb{D} = \{(x_i, y_i) : i = 1, \dots, n\}$$

We can specify a generative model as

$$P_{\mathbf{w}}(x, y)$$

and train the model on $\mathbb{D}$ *to find* $\mathbf{w}$ *such that* $P_{\mathbf{w}}(x, y) \approx P(x, y)$

+ *Can we use* $P_{\mathbf{w}}(x, y)$ *to estimate label of a new sample* $x_\star$*?*

Generative Learning: *More Generic Framework*

We can extract a discriminative model directly from the trained model as

$$P_{\mathbf{w}}(y|x) = \frac{P_{\mathbf{w}}(x,y)}{\boxed{P_{\mathbf{w}}(x)}} \longrightarrow \textbf{?}$$

We can use the marginalization rule to find $P_{\mathbf{w}}(x)$ *from the trained model*

$$P_{\mathbf{w}}(x) = \sum_y P_{\mathbf{w}}(x,y)$$

So, we have the following discriminative model

$$P_{\mathbf{w}}(y_\star|x_\star) = \frac{P_{\mathbf{w}}(x_\star,y_\star)}{\sum_y P_{\mathbf{w}}(x_\star,y)}$$

This is going to give us the probability of each possible label for sample $x_\star$ ✔

Components of Generative Learning

For any data distribution, we can apply *Bayes rule*

$$P(y_\star|x_\star) = \frac{P(x_\star, y_\star)}{\boxed{P(x_\star)}} = \frac{P(x_\star, y_\star)}{\sum_y P(x_\star, y)} = \frac{P(x_\star|y_\star)P(y_\star)}{\sum_y P(x_\star|y)P(y)}$$

Sample Marginal

Each component in this expression has a name

- $P(x_\star)$ *is the marginal sample distribution*
  - ↳ *It can be computed from the data distribution via marginalization*

$$P(x_\star) = \sum_y P(x_\star|y)P(y)$$

  - ↳ *This is to be learned if we don't care about the label of generated sample*

Components of Generative Learning

For any data distribution, we can apply *Bayes rule*

Label Generative

$$P(y_\star|x_\star) = \frac{P(x_\star, y_\star)}{P(x_\star)} = \frac{P(x_\star, y_\star)}{\sum_y P(x_\star, y)} = \frac{\boxed{P(x_\star|y_\star)}P(y_\star)}{\sum_y P(x_\star|y)P(y)}$$

Each component in this expression has a name

- $P(x_\star|y)$ *is the label generative model*
  - ↳ *It can be computed from the data distribution via conditioning*

$$P(x_\star|y) = \frac{P(x_\star, y)}{P(y)}$$

  - ↳ *This is to be learned if we want to generate samples of a specific label*
  - ↳ *Sometimes it is easier to hypothesize the parameterized model* $P_{\mathbf{w}}(x_\star|y)$

Components of Generative Learning

For any data distribution, we can apply *Bayes rule*

$$P(y_\star | x_\star) = \frac{P(x_\star, y_\star)}{P(x_\star)} = \frac{P(x_\star, y_\star)}{\sum_y P(x_\star, y)} = \frac{P(x_\star | y_\star) P(y_\star)}{\sum_y P(x_\star | y) \boxed{P(y)}}$$

Label Prior

Each component in this expression has a name

- $P(y)$ *is the label prior distribution*
  - ↳ *We typically postulate based on our prior belief*
    - ↳ *The prior belief could be build by assumption or numerical approximations*
    - ↳ *It's not hard to approximate it* ⇝ *CIFAR-100 has only 100 labels* ☺

Components of Generative Learning

For any data distribution, we can apply *Bayes rule*

Label Posterior

$$\boxed{P(y_\star|x_\star)} = \frac{P(x_\star, y_\star)}{P(x_\star)} = \frac{P(x_\star, y_\star)}{\sum_y P(x_\star, y)} = \frac{P(x_\star|y_\star)P(y_\star)}{\sum_y P(x_\star|y)P(y)}$$

Each component in this expression has a name

- $P(y_\star|x_\star)$ *is the label posterior distribution*
  - ↳ *We can compute it from data distribution via Bayes rule*
  - ↳ *We need it if we intend to infer label*
  - ↳ *In discriminative learning we hypothesize this distribution*
    - ↳ *Just assume it to be a parameterized model* $P_{\mathbf{w}}(y_\star|x_\star)$
  - ↳ *If we have trained a generative model: we can compute it by Bayes rule!*
    - ↳ *Note that the other way around is not easy* $\rightsquigarrow$ *we need* $P(x)$*!*

Generative or Discriminative Learning

+ *Do we always learn generative model when we want to generate and discriminative model when we are inferring?*
– Though classic, it is not necessarily the case
   ↳ *It's better to think through the universal framework*
+ *Are there examples that use generative learning for discriminative tasks?*
– Sure! Most famous one is the *naive Bayes approach*

Next Step

We take a look at naive Bayes: the most basic generative learning approach

↳ *This helps us understand the difference between*

generative and discriminative modeling

↳ *We are then ready to dive into benchmark deep generative models*