# ECEN 740:
# Machine Learning Engineering

## Lecture 3: Generative Approach to Machine Learning

Tie Liu

Texas A&M University

# Generative approach

- The *generative* approach to machine learning is an *indirect* approach that aims to first learn the *data-generating* distribution and then uses the learned distribution to perform on a given task
- It can be viewed as a *universal* approach in that once the data-generating distribution is learned, one can use the learned distribution to perform on *any* given task
- On the other hand, the data-generating distribution generally contains *more* information than what is needed to perform on a specific task. With only a limited amount of training examples, the generative approach may perform substantially worse than a more direct approach that aims to learn *just* what is needed to perform on the specific task
- Furthermore, the learning of the data-generating distribution is mostly driven by a *generative* goal rather than the *discriminative* goal prescribed by the specific task. This potentially also makes the generative approach perform worse than the more direct discriminative approach

# Generative probabilistic models

- The main challenge for learning the data-generating distribution is the *curse of dimension*
- More specifically, if our data is $d$-dimensional and the alphabet size of each dimension is $a$, the data-generating distribution is parameterized by a total of $a^d$ probabilities
- The fact that the number of parameters of a general $d$-dimensional distribution grows *exponentially* with $d$ makes it very difficult to learn a *general* high-dimensional distribution from a limited number of training data examples
- The curse of dimension thus forces us to learn a high-dimensional distribution by *restricting* it to within a family of distributions indexed by a parameter $\theta$
- We call the family of distributions a *parametric probabilistic model* if $\theta$ is *finite-dimensional* and a *non-parametric probabilistic model* if $\theta$ is *infinite-dimension*. In the latter case, we usually think of $\theta$ as a *function*

# Point estimation vs. Bayesian learning

- Once a family of distributions is specified, learning can be done through either *point estimation* or *Bayesian learning*
- For the point-estimation approach, the goal of learning is to perform a *point estimate* of the model parameter $\theta$ based on the training data examples - a topic that we have discussed extensively in our previous lectures
- In this approach, we assume that all information from the training data examples is *completely* captured by the learned model parameter:

$$p(z_{new}|\theta^*, z_1, \ldots, z_m) = p(z_{new}|\theta^*)$$

  where $\theta^*$ is the point estimate of $\theta$ based on the training examples $(z_1, \ldots, z_m)$
- An important advantage of this approach is that once learning is done, all future generative or discriminative tasks will be performed based on the learned parameter $\theta^*$; the training data examples $(z_1, \ldots, z_m)$ can be completely disregarded

- Instead of producing a point estimate of $\theta$, Bayesian learning assigns a *prior* to $\theta$ (as a Bayesian approach always does) and then calculates the *posterior* of $\theta$ given the training data examples $(z_1, \ldots, z_m)$
- The *predictive density* given the training data examples $(z_1, \ldots, z_m)$ is thus given by

$$p(z_{new}|z_1, \ldots, z_m) = \int p(z_{new}|\theta)p(\theta|z_1, \ldots, z_m)d\theta$$

  which is a *Bayesian mixture* of the distributions from the specified family
- The downstream generative or discriminative task will then be performed based on the predictive density $p(z_{new}|z_1, \ldots, z_m)$
- Compared with the point-estimation approach, Bayesian learning is more *robust* but at the expense of more computations. In addition, unlike the point-estimation approach, training data examples *cannot* be disregarded and must be carried on to the downstream task

- While a parametric model can be trained either through point estimation or Bayesian learning, a non-parametric model is *exclusively* trained through Bayesian learning

- We will discuss a specific non-parametric (discriminative) probabilistic model known as *Gaussian process* in a later lecture

- For the rest of this lecture, we will focus on *parametric* generative probabilistic models trained via *point estimation*

- We emphasize here that point estimation itself can be done through either a frequentist approach or a Bayesian approach. The difference between the point-estimation approach and Bayesian learning is whether the predictive density is given by a *single* distribution or a *Bayesian mixture* of the distributions from the family

# Modeling data generation

- A powerful approach for building a *learnable* probabilistic model is to think about how data might be *generated* and make *simplifying* assumptions along the data-generation process
- Here we mention two important techniques for modeling data generation: *dependency* modeling and the use of *latent* variables
- For dependency modeling, we will discuss a general mathematical framework known as *Bayesian network*
- For the use of latent variables, we will discuss a specific generative probabilistic model for text corpora known as *latent Dirichlet allocation (LDA)*

# Dependency modeling

- Assume that each of our data examples $z = (z_1, \ldots, z_d)$ is $d$-dimensional. By the chain rule of probability, the joint probability

$$p(z) = \prod_{i=1}^{d} p(z_i | z_1, \ldots, z_{i-1})$$

- That is, a $d$-dimensional data example $\mathsf{z}$ can be generated in $d$ *sequential* steps: First draw $\mathsf{z}_1 \sim p(z_1)$. Second, draw $\mathsf{z}_2 \sim p(z_2|z_1)$. Next, draw $\mathsf{z}_3 \sim p(z_3|z_1, z_2)$. The process goes on till finally draw $\mathsf{z}_d \sim p(z_d|z_1, \ldots, z_{d-1})$

- Of course, the above generation process can follow *any* order on $[d]$, not necessarily the natural order

- The chain rule of probability also suggests that to estimate the joint probability $p(z)$, we can *separately* estimate the conditional probabilities $p(z_i|z_1, \ldots, z_{i-1})$ and multiply the estimates together to obtain an estimate of the joint probability $p(z)$

- In general, this strategy does *not* help because the conditional probabilities towards the *bottom* of the chain remain to be *high-dimensional*

- On the other hand, the above separate estimation strategy will work if the generation of each $\mathsf{z}_i$ only depends on a *small* subset $\mathsf{z}_{p(i)}$ of $\{\mathsf{z}_1, \ldots, \mathsf{z}_{i-1}\}$:

$$p(z_i|z_1, \ldots, z_{i-1}) = p(z_i|z_{p(i)})$$

  i.e., $\mathsf{z}_i$ is conditionally independent of $\mathsf{z}_{[i-1]\setminus p(i)}$ given $\mathsf{z}_{p(i)}$

- If $p(i)$ is small for all $i$, all conditional probabilities are *low-dimensional* and hence can be estimated from a limited amount of training data examples

# Bayesian network

- We can represent the aforementioned conditional dependencies using a *directed graph*, where
  - node $i$ represents the random variable $\mathsf{z}_i$; and
  - there is an arc from node $j$ to node $i$ if and only if $j \in p(i)$
- Clearly, in the above graph, $p(i)$ represents the set of *parents* of $i$. Furthermore, if $p(i)$ is a subset of $[i-1]$ for all $i$, the directed graph must be *acyclic*
- Conversely, for any *directed acyclic graph (DAG)* for which each node represents a random variable, we can relabel these random variables according to a *topological order* of the graph. Under this labeling, the set of *parents* $p(i)$ must be a *subset* of $[i-1]$
- Thus, the aforementioned conditional dependencies can be *completely* characterized by a *DAG*. A set of random variables whose conditional dependencies are represented by a DAG is known as a *Bayesian network*

# Parameter estimation and structure learning

- Once the underlying DAG is specified, a Bayesian network model is parameterized by the conditional probabilities
- When the conditional probabilities are *low-dimensional*, they can be estimated from a limited amount of training data examples
- Note that the entire joint probability can be wiped off by a single zero conditional probability. Therefore, it is important to apply *smoothing techniques* such as Laplace's rule of succession when it comes to estimating the conditional probabilities
- On the other hand, learning a DAG from the training data examples is known as *structure learning*, which is much more difficult than parameter estimation

# Naive Bayes

- For the problem of *learn-to-predict*, each data example $\mathsf{z} = (\mathsf{x}, \mathsf{y})$ consists of an observed variable $\mathsf{x}$ and a target variable $\mathsf{y}$
- The observed variable $\mathsf{x} = (\mathsf{x}_1, \ldots, \mathsf{x}_d)$ is usually high-dimensional and we assume that it has $d$ *attributes*
- A very simple probabilistic generative model in this case is to assume that $\mathsf{z}$ is a *Bayesian network* for which

$$p(z) = p(y) \prod_{i=1}^{d} p(x_i|y)$$

  i.e., the attributes are *conditionally independent* given the target. In the literature, this model is known as *naive Bayes*
- Note that in naive Bayes, all attributes have a *single* parent, so all conditional probabilities can be estimated from a limited amount of training examples
- Despite being a rather *coarse* generative model, many empirical studies showed that in many cases it leads to predictors that are surprisingly accurate

# Latent Dirichlet allocation (LDA)

- *Latent Dirichlet allocation (LDA)* is a generative probabilistic model of a text *corpus*, for which the basic idea is that *documents* are represented as random mixtures over latent *topics*
- In this model, the corpus is characterized by a *distribution* over *documents*, each document is characterized by a *distribution* over *topics*, and each topic is characterized by a *distribution* over *words*
- More specifically, let $\theta_i$ be the topic distribution for document $i$, and let $\varphi_k$ be the word distribution for topic $k$. LDA assumes the following generative process for each document $\mathsf{w}_i$ in a corpus $\mathsf{D}$:

  1) Choose $\theta_i \sim Dir(\alpha)$
  2) For each word $\mathsf{w}_{i,j}$ in $\mathsf{w}_i$:
     a) Choose a topic $\mathsf{z}_{i,j} \sim Cat(\theta_i)$
     b) Choose a word $\mathsf{w}_{i,j} \sim Cat(\varphi_{z_{i,j}})$

  where $\alpha$ (which is a positive vector whose dimension equals the number of topics) and $\varphi = (\varphi_k)_k$ are the parameters of the model

- Given the parameters $(\alpha, \varphi)$, the joint distribution of the topic distribution $\theta_i$, the topics $\mathsf{z}_i = (\mathsf{z}_{i,j})_j$, and the words $\mathsf{w}_i = (\mathsf{w}_{i,j})_j$ is given by

$$p(\theta_i, z_i, w_i | \alpha, \varphi) = p(\theta_i | \alpha) \prod_j p(z_{i,j} | \theta_i) p(w_{i,j} | z_{i,j}, \varphi)$$

- Integrating over $\theta_i$ and summing over $z_i$, we obtain the marginal distribution of a document

$$p(w_i | \alpha, \varphi) = \int p(\theta_i | \alpha) \prod_j \left\{ \sum_{z_{i,j}} p(z_{i,j} | \theta_i) p(w_{i,j} | z_{i,j}, \varphi) \right\} d\theta_i$$

- Finally, taking the product of the marginal probabilities of single documents, we obtain the probability of a corpus

$$p(D | \alpha, \varphi) = \prod_i \left\{ \int p(\theta_i | \alpha) \prod_j \left\{ \sum_{z_{i,j}} p(z_{i,j} | \theta_i) p(w_{i,j} | z_{i,j}, \varphi) \right\} d\theta_i \right\}$$

# Parameter estimation

- To learn the model parameter $(\alpha, \varphi)$ from a given corpus $D$, one may consider the ML estimator:

$$\delta_{ML}(D) = \arg\max_{(\alpha,\varphi)} \log p(D|\alpha, \varphi)$$

- The log-likelihood, however, is *intractable*. Instead, one may consider maximizing the *ELBO* instead:

$$\begin{aligned}&\delta_{ME}(D)\\ &= \arg\max_{(\alpha,\varphi)} \left\{ \max_{q\in\mathcal{Q}} \{\log p(D|\alpha, \varphi) - D_{KL}(q(\theta, z)\|p(\theta, z|D, \alpha, \varphi)\} \right\}\end{aligned}$$

- Once a variational family $\mathcal{Q}$ is specified, the above double-maximization problem can be (approximately) solved using an *iterative* procedure that iterates between maximizing over the variational factor $q(\theta, z)$ for a fixed parameter $(\alpha, \varphi)$ and maximizing over the parameter $(\alpha, \varphi)$ for a fixed variational factor $q(\theta, z)$

- Note that the parameter $\varphi$ is a collection of word *distributions*, for which the ML estimates are known to be *non-smooth*

- To produce a smooth estimate of the word distributions, one may have to consider a *fuller* Bayesian approach. The details can be found in the immensely popular paper by Blei, Ng, and Jordan (2003)

# VI vs. generative probabilistic model

- We saw from our previous studies that both VI and the generative approach to machine learning requires a probabilistic *model*
- However, there is a key difference on how such a model needs to be chosen between these two settings
- For VI, the only factor for choosing a variational family is *computational* complexity
- For the generative approach, in addition to computational complexity, choosing a probabilistic model also needs to consider its *learnability* from a *finite* number of training data examples, i.e., the *sample complexity*
- This is the key difference between a computational problem such as VI and a learning problem (no matter what approach we will be using)