Expectation Maximization (EM)

Karl Stratos

June 27, 2018

The Latent Variable Paradigm

- Observed instances $x \in \mathcal{X}$ (your data)

The Latent Variable Paradigm

- Observed instances $x \in \mathcal{X}$ (your data)
- Latent variables $z \in \mathcal{Z}$

The Latent Variable Paradigm

- Observed instances $x \in \mathcal{X}$ (your data)
- Latent variables $z \in \mathcal{Z}$
- Probabilistic generative model: Φ defining

$$P_\Phi(x, z) = P_\Phi(z)P_\Phi(x|z)$$

The Latent Variable Paradigm

- Observed instances $x \in \mathcal{X}$ (your data)
- Latent variables $z \in \mathcal{Z}$
- Probabilistic generative model: Φ defining

$$P_\Phi(x, z) = P_\Phi(z) P_\Phi(x|z)$$

- Idea: we *believe* that x has been generated from some unobserved variables z, so we should model (x, z) jointly rather than just x (even though we don't see z in data).

Fitting Data Better with Latent Variables

- Data: two instances

$$x^{(1)} = (a, a)$$
$$x^{(2)} = (b, b)$$

- A generative model $P_\Theta(x)$ over $x \in \{a, b\}$ without latent variables: for each $i = 1, 2$,
  - Draw $x_1^{(i)} \sim P_\Theta(x)$.
  - Draw $x_2^{(i)} \sim P_\Theta(x)$.

  What's the highest probability that Θ can assign to this data?

Fitting Data Better with Latent Variables

- Data: two instances

$$x^{(1)} = (a, a)$$
$$x^{(2)} = (b, b)$$

- A generative model $P_\Theta(x)$ over $x \in \{a, b\}$ without latent variables: for each $i = 1, 2$,
  - Draw $x_1^{(i)} \sim P_\Theta(x)$.
  - Draw $x_2^{(i)} \sim P_\Theta(x)$.

  What's the highest probability that Θ can assign to this data?

- A latent-variable model $P_\Phi(x, z) = P_\Phi(z) P_\Phi(x|z)$ over $x \in \{a, b\}$ and $z \in \{1, 2\}$: for each $i = 1, 2$,
  - Draw $z^{(i)} \sim P_\Phi(z)$
  - Draw $x_1^{(i)} \sim P_\Theta(x|z^{(i)})$.
  - Draw $x_2^{(i)} \sim P_\Theta(x|z^{(i)})$.

  What's the highest probability that Φ can assign to this data?

What Are Latent-Variable Models Useful For?

1. **More expressive model**: which leads to improved performance

2. **Interpretability**: discover latent structure z to understand data/problem better

3. **Controlled generation**: once we learn the model, we can control our generation through z

$$z \sim P_\Phi(\cdot)$$
$$x \sim P_\Phi(\cdot|z)$$

Overview

The Learning Problem

- How can we learn model Φ that defines $P_\Phi(x, z)$ when we only observe x?

The Learning Problem

- How can we learn model Φ that defines $P_\Phi(x, z)$ when we only observe x?

- Thought experiment: had we observed z as well in our data, we could've just done maximum-likelihood estimate (MLE):

$$\Phi^* = \arg\max_{\Phi} \; \log P_\Phi(x, z)$$

The Learning Problem

- How can we learn model Φ that defines $P_\Phi(x, z)$ when we only observe x?

- Thought experiment: had we observed z as well in our data, we could've just done maximum-likelihood estimate (MLE):

$$\Phi^* = \arg\max_{\Phi} \; \log P_\Phi(x, z)$$

- If we don't observe z, we can still do MLE on what we *do* observe:

$$\Phi^* = \arg\max_{\Phi} \; \log P_\Phi(x)$$

where

$$P_\Phi(x) = \sum_{z \in \mathcal{Z}} P_\Phi(x, z)$$

Learning = Density Estimation

- Wikipedia:

  "**density estimation** is the construction of an estimate, based on observed data, of an unobservable underlying probability density function"

- From here on, we will focus on MLE: the problem of maximizing

$$\log \sum_{z \in \mathcal{Z}} P_\Phi(x, z)$$

over Φ when we only observe x

Overview

- Learning Latent-Variable Models by Density Estimation
  - Quick Review of Information Theory
- ELBO: Lower Bound on Log Likelihood
- The Expectation Maximization (EM) Algorithm
- Example: Naive Bayes

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.
- The number of bits to encode the amount of surprise upon seeing z is $\log(1/P(z))$.

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.
- The number of bits to encode the amount of surprise upon seeing z is $\log(1/P(z))$.
- The **entropy** of P is the expected number of bits to encode the amount of surprise when z is drawn from P itself:

$$H(P) := \mathbf{E}_{z \sim P(\cdot)} \left[\log \frac{1}{P(z)} \right] = -\sum_{z} P(z) \log P(z)$$

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.
- The number of bits to encode the amount of surprise upon seeing z is $\log(1/P(z))$.
- The **entropy** of P is the expected number of bits to encode the amount of surprise when z is drawn from P itself:

$$H(P) := \mathbf{E}_{z \sim P(\cdot)} \left[\log \frac{1}{P(z)} \right] = - \sum_{z} P(z) \log P(z)$$

- Convention: if $P(z) = 0$ for some z, we ignore it in the sum.

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.
- The number of bits to encode the amount of surprise upon seeing z is $\log(1/P(z))$.
- The **entropy** of P is the expected number of bits to encode the amount of surprise when z is drawn from P itself:

$$H(P) := \mathbf{E}_{z \sim P(\cdot)} \left[\log \frac{1}{P(z)} \right] = - \sum_{z} P(z) \log P(z)$$

- Convention: if $P(z) = 0$ for some z, we ignore it in the sum.
- Thus if $P(z) = 1$ is deterministic, then the entropy is 0.

Entropy

Given a distribution P over z,

- The "amount of surprise" upon seeing z is quantified by $1/P(z)$.
- The number of bits to encode the amount of surprise upon seeing z is $\log(1/P(z))$.
- The **entropy** of P is the expected number of bits to encode the amount of surprise when z is drawn from P itself:

$$H(P) := \mathbf{E}_{z \sim P(\cdot)} \left[\log \frac{1}{P(z)} \right] = - \sum_{z} P(z) \log P(z)$$

- Convention: if $P(z) = 0$ for some z, we ignore it in the sum.
- Thus if $P(z) = 1$ is deterministic, then the entropy is 0.
- The entropy is always nonnegative and maximized when P is uniform over z.

Cross Entropy and KL Divergence

Given a distribution P and Q over z,

- The **cross entropy** between P and Q is the the expected number of bits to encode the amount of surprise of Q when z is drawn from P:

$$H(P, Q) := \mathbf{E}_{z \sim P(\cdot)} \left[\log \frac{1}{Q(z)} \right] = - \sum_{z} P(z) \log Q(z)$$

Cross Entropy and KL Divergence

Given a distribution P and Q over z,

- The **cross entropy** between P and Q is the the expected number of bits to encode the amount of surprise of Q when z is drawn from P:

$$H(P,Q) := \mathbf{E}_{z \sim P(\cdot)}\left[\log \frac{1}{Q(z)}\right] = -\sum_{z} P(z) \log Q(z)$$

- The cross entropy is always nonnegative and minimized when $P = Q$.

Cross Entropy and KL Divergence

Given a distribution P and Q over z,

- The **cross entropy** between P and Q is the the expected number of bits to encode the amount of surprise of Q when z is drawn from P:

$$H(P,Q) := \mathbf{E}_{z\sim P(\cdot)}\left[\log\frac{1}{Q(z)}\right] = -\sum_z P(z)\log Q(z)$$

- The cross entropy is always nonnegative and minimized when $P = Q$.

- The **KL divergence** from Q to P is the *additional* number of bits to encode the amount of surprise of Q compared to the amount of surprise of P, when z is drawn from P:

$$D_{\mathrm{KL}}(P||Q) := H(P,Q) - H(P)$$

Cross Entropy and KL Divergence

Given a distribution P and Q over z,

- The **cross entropy** between P and Q is the the expected number of bits to encode the amount of surprise of Q when z is drawn from P:

$$H(P,Q) := \mathbf{E}_{z\sim P(\cdot)}\left[\log \frac{1}{Q(z)}\right] = -\sum_{z} P(z)\log Q(z)$$

- The cross entropy is always nonnegative and minimized when $P = Q$.

- The **KL divergence** from Q to P is the *additional* number of bits to encode the amount of surprise of Q compared to the amount of surprise of P, when z is drawn from P:

$$D_{\text{KL}}\left(P||Q\right) := H(P,Q) - H(P)$$

- The KL divergence is zero iff $P = Q$.

Overview

The Idea of Introducing an Auxiliary Posterior

- Maximizing $\log P_\Phi(x)$ is hard, whereas maximizing $\log P_\Phi(x, z)$ when z is observed is easier.

- We will introduce an auxiliary model Ψ that specifies (its own) posterior distribution $P_\Psi(z|x)$ and use it to "help" Φ.

The Idea of Introducing an Auxiliary Posterior

- Maximizing $\log P_\Phi(x)$ is hard, whereas maximizing $\log P_\Phi(x, z)$ when z is observed is easier.

- We will introduce an auxiliary model Ψ that specifies (its own) posterior distribution $P_\Psi(z|x)$ and use it to "help" Φ.

- **Clarification**: Φ is a model that defines a joint distribution

$$P_\Phi(x, z)$$

which defines marginal $P_\Phi(x) = \sum_z P_\Phi(x, z)$ and posterior $P_\Phi(z|x) = P_\Phi(x, z)/P_\Phi(x)$ probabilities.

The Idea of Introducing an Auxiliary Posterior

- Maximizing $\log P_\Phi(x)$ is hard, whereas maximizing $\log P_\Phi(x, z)$ when z is observed is easier.

- We will introduce an auxiliary model Ψ that specifies (its own) posterior distribution $P_\Psi(z|x)$ and use it to "help" Φ.

- **Clarification**: Φ is a model that defines a joint distribution

$$P_\Phi(x, z)$$

  which defines marginal $P_\Phi(x) = \sum_z P_\Phi(x, z)$ and posterior $P_\Phi(z|x) = P_\Phi(x, z)/P_\Phi(x)$ probabilities.

  In constrast, Ψ is some other model that defines its own posterior

$$P_\Psi(z|x)$$

  Ψ does not have to define a joint distribution over x and z.

ELBO: Evidence Lower Bound

$$\mathsf{ELBO}(\Phi, \Psi) := \log P_\Phi(x) - \underbrace{D_{\mathsf{KL}}\left(P_\Psi(z|x) || P_\Phi(z|x)\right)}_{\geq 0}$$

ELBO: Evidence Lower Bound

$$\text{ELBO}(\Phi, \Psi) := \log P_\Phi(x) - \underbrace{D_{\text{KL}}\left(P_\Psi(z|x) || P_\Phi(z|x)\right)}_{\geq 0}$$

For any choice of Ψ, $\text{ELBO}(\Phi, \Psi)$ is a **lower bound** on the log likelihood of observed data

$$\log P_\Phi(x) := \log \sum_{z} P_\Phi(x, z)$$

Claim 1: ELBO and Expected Likelihood

$$\text{ELBO}(\Phi, \Psi)$$

$$= \mathbf{E}_{z \sim P_\Psi(\cdot|x)} \left[\underbrace{\log P_\Phi(x, z)}_{\text{“fully observed”}} \right] + H(P_\Psi(z|x))$$

Claim 2: ELBO and Autoencoder

$$\begin{aligned}&\text{ELBO}(\Phi, \Psi)\\&= \mathbf{E}_{z\sim P_\Psi(\cdot|x)}\left[\log P_\Phi(x|z)\right] - D_{\text{KL}}\left(P_\Psi(z|x)||P_\Phi(z)\right)\end{aligned}$$

Ψ "encodes" x into z, Φ "decodes" x from z.

Overview

EM: Coordinate Ascent on ELBO

Input: data x, definition of $P_\Phi(x,z)$ and $P_\Psi(z|x)$, integer T
Output: estimation of Φ that locally maximizes $\log P_\Phi(x)$

1. Initialize $\Phi^{(0)}$ and $\Psi^{(0)}$.
2. For $t = 1 \dots T$,

$$\Psi^{(t)} \leftarrow \arg\max_{\Psi} \ \text{ELBO}(\Phi^{(t-1)}, \Psi)$$
$$\Phi^{(t)} \leftarrow \arg\max_{\Phi} \ \text{ELBO}(\Phi, \Psi^{(t)})$$

3. Return $\Phi^{(T)}$.

EM: ELBO Definition Expanded

Input: data x, definition of $P_\Phi(x, z)$ and $P_\Psi(z|x)$, integer T
Output: estimation of Φ that locally maximizes $\log P_\Phi(x)$

1. Initialize $\Phi^{(0)}$ and $\Psi^{(0)}$.
2. For $t = 1 \dots T$,

$$\Psi^{(t)} \in \{\Psi : \ P_\Psi(z|x) = P_{\Phi^{(t-1)}}(z|x)\}$$

$$\Phi^{(t)} \leftarrow \arg\max_\Phi \ \mathbf{E}_{z \sim P_{\Psi^{(t)}}(\cdot|x)} [\log P_\Phi(x, z)]$$

3. Return $\Phi^{(T)}$.

EM: Lazy Version

Input: data x, definition of $P_\Phi(x, z)$, integer T
Output: estimation of Φ that locally maximizes $\log P_\Phi(x)$

1. Initialize $\Phi^{(0)}$.
2. For $t = 1 \ldots T$,

$$\Phi^{(t+1)} \leftarrow \arg\max_{\Phi} \quad \mathbf{E}_{z \sim P_{\Phi^{(t)}}(\cdot|x)} \left[\log P_\Phi(x, z)\right]$$

3. Return $\Phi^{(T)}$.

Overview

Learning Latent-Variable Models by Density Estimation

Quick Review of Information Theory

ELBO: Lower Bound on Log Likelihood

The Expectation Maximization (EM) Algorithm

Example: Naive Bayes

Naive Bayes (NB) Review

- A generative model for classification

  Input. List of d discrete (here, binary) features $\boldsymbol{x} \in \{0,1\}^d$
  Output. One of m discrete labels $y \in \{1 \dots m\}$

- $m + 2dm$ parameters

  $q(y)$ for each $y = 1 \dots m$
  $q(0|y,j)$ and $q(1|y,j)$ for each $j = 1 \dots d$ and $y = 1 \dots m$

- Conditional independence assumption!

$$p(\boldsymbol{x}, y) = q(y) \prod_{j=1}^{d} q(x_j | y, j)$$

- Inference: given $\boldsymbol{x} \in \{0,1\}^d$, calculate

$$y^* = \arg\max_{y \in \{1 \dots m\}} p(y|\boldsymbol{x}) = \arg\max_{y \in \{1 \dots m\}} p(\boldsymbol{x}, y)$$

Naive Bayes Review: Supervised Learning

- **Lemma.** Given any $c_1 \ldots c_l \geq 0$ (not all zero),

$$q_1^* \ldots q_l^* = \underset{q_1 \ldots q_l \geq 0:\ \sum_{i=1}^{l} q_i = 1}{\arg\max} \sum_{i=1}^{l} c_i \log q_i$$

  are given by $q_i^* = c_i / \sum_{j=1}^{l} c_j$.

- Given labeled training data $(\boldsymbol{x}^{(1)}, y^{(1)}) \ldots (\boldsymbol{x}^{(n)}, y^{(n)})$, log likelihood under NB is

$$\sum_{i=1}^{n} \log q(y^{(i)}) + \sum_{j=1}^{d} \log q(x_j^{(i)} | y, j)$$

$$= \sum_{y=1}^{m} \textbf{count}(y) \log q(y^{(i)})$$

$$+ \sum_{y=1}^{m} \sum_{j=1}^{m} \sum_{x \in \{0,1\}} \textbf{count}(y, j, x) \log q(x | y, j)$$

Naive Bayes Review: Supervised Learning (Cont.)

- Thus MLE solution is given by counts:

$$q(y) = \frac{\textbf{count}(y)}{n} \qquad \forall y \in \{1 \dots m\}$$

and

$$q(x|y,j) = \frac{\textbf{count}(y,j,x)}{\textbf{count}(y,j,0) + \textbf{count}(y,j,1)} \quad \begin{array}{l} \forall y \in \{1 \dots m\} \\ j \in \{1 \dots d\} \\ x \in \{0,1\} \end{array}$$

Naive Bayes: Unsupervised Learning

Now I remove the labels $y^{(1)} \ldots y^{(n)}$. Your data consists of n feature vectors

$$\boldsymbol{x}^{(1)} \ldots \boldsymbol{x}^{(n)} \in \{0,1\}^d$$

We can use EM to learn NB parameters $q(y)$ and $q(x|y,j)$ that optimize $\log p(\boldsymbol{x}^{(1)} \ldots \boldsymbol{x}^{(n)})$. Apply the EM algorithm below:

Input: data $\boldsymbol{x}^{(1)} \ldots \boldsymbol{x}^{(n)} \in \{0,1\}^d$, integer T

1. Initialize NB parameters $\Phi^{(0)}$.
2. For $t = 1 \ldots T$,

$$\Phi^{(t+1)} \leftarrow \arg\max_{\Phi} \sum_{i=1}^{n} \sum_{y=1}^{m} P_{\Phi^{(t)}}(y|\boldsymbol{x}^{(i)}) \times \log P_{\Phi}(\boldsymbol{x}^{(i)}, y)$$

3. Return $\Phi^{(T)}$.