Exponential family

Properties of exponential family distributions

We will need

- Formulation
- Natural parameter
- Whether they keep product and sum
- Definition of conjugate

EM algorithm

Motivation

Suppose that we have sample X, that comes from a distribution with density $p(\cdot)$, parametrized by (unknown) parameters θ that we'd like to estimate from sample:

$$X = (x_1, x_2, \dots, x_n) \sim p(x|\theta)$$

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \to \max_\theta$$

What should we do if:

- $p(x|\theta)$ is $\mathcal{N}(\mu, \sigma)$
- $p(x|\theta)$ is from exponential family $p(x|\theta) = \frac{f(x)}{g(\theta)}\exp\big(\theta^\top u(x)\big)$
- $p(x|\theta)$ is not from exponential family

Motivation

If $p(x|\theta)$ is not from exponential family, we can insert **latent variables** z into our distribution, so that $p(x|z, \theta)$ is from exponential family.

Example: mixture models

$$p(x|\theta) = \sum_{k=1}^{K} \alpha_k p_k(x, \theta_k)$$

You can verify that $p(x|\theta)$ does not belong to exponential family.

Let's insert variables z, such that

- $z_k \in \{0, 1\}$
- $\sum_k z_k = 1$
- $q(z_k = 1) = \alpha_k$

Then,

$$p(x, z|\theta) = \prod_{k=1}^{K} (p_k(x, \theta_k))^{z_k}$$

You can verify that $p(x|z, \theta)$ belongs to exponential family with natural parameter $\sum_k z_k \theta_k$.

Derivation

Suppose that we have sample X, that follow the distribution with density $p(\cdot)$, parametrized by (unknown) parameters θ that we'd like to estimate from sample:

$$X = (x_1, x_2, \dots, x_n) \sim p(x|\theta)$$

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) \to \max_\theta$$

Note that $p(x_i|\theta)$ is not from exponential family. Therefore, we'll be using latent variables Z that follow the distribution $q(\cdot)$, such that $p(x_i, z_i|\theta)$ is from exponential family:

$$Z = (z_1, z_2, \dots, z_n) \sim q(z)$$

Derivation

$$L = \log p(x|\theta) = \log p(x|\theta) \cdot \int q(z)\mathrm{dz} = \int \mathrm{q(z)\log p(x|\theta)dz}$$

Now use full probability formula $p(x, z|\theta) = p(z|x, \theta)p(x|\theta)$:

$$L = \int q(z)\log p(x|\theta)\mathrm{dz} = \int \mathrm{q(z)} \log \frac{\mathrm{p(x,z|\theta)}}{\mathrm{p(z|x,\theta)}}\mathrm{dz} = \int \mathrm{q(z)} \log \frac{\mathrm{p(x,z|\theta)q(z)}}{\mathrm{p(z|x,\theta)q(z)}}\mathrm{dz}$$

Now let's use some properties of $\log$:

$$L = \int q(z)\log\frac{p(x,z|\theta)q(z)}{p(z|x,\theta)q(z)}\mathrm{dz} = \int \mathrm{q(z)}\left(\log\frac{\mathrm{p(x,z|\theta)}}{\mathrm{q(z)}} + \log\frac{\mathrm{q(z)}}{\mathrm{p(z|x,\theta)}}\right)\mathrm{dz}$$

Finally use the linearity of the integral:

$$L = \int q(z)\log\frac{p(x,z|\theta)}{q(z)}\mathrm{dz} + \underbrace{\int \mathrm{q(z)}\log\frac{\mathrm{q(z)}}{\mathrm{p(z|x,\theta)}}\mathrm{dz}}_{?}$$

KL divergence

$$D_{\mathrm{KL}}(p||q) \equiv KL(p||q) = \int p(x)\log\frac{p(x)}{q(x)}\mathrm{dx}$$

Properties:

- $KL(p||q) \neq KL(q||p)$
- $KL(p||q) \geqslant 0$ (prove)

Derivation

Overall,

$$L = \log p(x|\theta) = \int q(z)\log\frac{p(x,z|\theta)}{q(z)}\mathrm{dz} + \mathrm{KL(q(z)||p(z|x,\theta))} \geqslant \int \mathrm{q(z)}\log\frac{\mathrm{p(x,z|\theta)}}{\mathrm{q(z)}}\mathrm{dz}$$

This quantity is called **variational lower bound**

$$\mathcal{L}(q, \theta) = \int q(z)\log\frac{p(x,z|\theta)}{q(z)}\mathrm{dz}$$

We will transform our problem into $\mathcal{L}(q, \theta) \to \max_{q,\theta}$. We will be solving this problem using **coordinate descent**, i.e. successively maximize along the two directions:

1. $q^* = \arg\max_q \mathcal{L}(q, \theta^*)$ (**E-step**)
2. $\theta^* = \arg\max_\theta \mathcal{L}(q^*, \theta)$ (**M-step**)

Tricks

E-step

Let's recall that $q(\cdot)$ was not present in the original likelihood, therefore $\partial L/\partial q \equiv 0$.

Also recall that at some point in derivation, we had the following equality: $L = \mathcal{L}(q, \theta) + KL(q(z)||p(z|x, \theta))$.

Therefore, maximizing $\mathcal{L}(q, \theta)$ w.r.t. q is equivalent to minimizing $KL(q(z)||p(z|x, \theta))$ w.r.t. q!

Think, where does KL-divergence achieve its minimum?

$$KL(p||q) = \int p(x)\log\frac{p(x)}{q(x)}\mathrm{dx}$$

$$\arg\min_p KL(p||q) = q$$

Therefore we have an exact solution for E-step (one limitation is obvious, does anyone notice?):

$$q^* = \arg\max_q \mathcal{L}(q, \theta^*) = p(z|x, \theta)$$

Tricks

M-step

$$\begin{aligned}\arg\max_\theta \mathcal{L}(q^*, \theta) &= \arg\max_\theta \int q^*(z)\log\frac{p(x,z|\theta)}{q^*(z)}\mathrm{dz} = \\ &= \arg\max_\theta\left(\int q^*(z)\log p(x,z|\theta)\mathrm{dz} - \int \mathrm{q^*(z)\log q^*(z)dz}\right) = \\ &= \arg\max_\theta \int q^*(z)\log p(x,z|\theta)\mathrm{dz}\end{aligned}$$

$$\mathcal{L}(q^*, \theta) = \int q^*(z)\log p(x,z|\theta)\mathrm{dz} = \int \mathrm{p(z|x,\theta)\log p(x,z|\theta)dz} = \mathbb{E}_{\mathrm{p(z|x,\theta)}}\log \mathrm{p(x,z|\theta)}$$

Tricks

M-step

For the full sample we'll have

$$\mathcal{L}(q^*, \theta) = \sum_{i=1}^{N}\mathbb{E}_{p(z|x,\theta)}\log p(x, z|\theta)$$

Which is impractical for large datasets.

Solution:

- Use Monte-Carlo estimation of mean and
- Stochastic gradient

$$\theta_{t+1} = \theta_t + \eta_t \cdot n \cdot \nabla_\theta \log p(x_i, z_i|\theta)$$

Tricks

Final algorithm

Iterate until convergence:

1. $q(z_i) = p(z_i|x_i, \theta)$
2. $\theta_{t+1} = \theta_t + \eta_t \cdot n \cdot \nabla_\theta \mathbb{E}_{p(z_i|x_i,\theta)}\log p(x_i, z_i|\theta)$

Code

Problem

Consider two coins, A and B, with different probabilities of success θ_A and θ_B. The experiment is as follows: we randomly choose a coin, then flip it n times and record the number of successes.

If we recorded which coin we used for each sample, we have complete information and can estimate θ_A and θ_B in closed form.

- What is the probabilistic model of this experiment?

$$X \sim Be\left(\frac{1}{2}\right)Bi(\theta_A, n) + Be\left(\frac{1}{2}\right)Bi(\theta_B, n)$$

- What are the MLE estimators for θ_A and θ_B?

$$\theta_A^{\mathrm{MLE}} = \frac{\text{number of successes for A}}{\text{number of trails for A}}$$

In [10]:
```
import numpy as np
import scipy.stats as sts
```

In [17]:
```
n = 1000

theta_A = 0.8
theta_B = 0.35

theta_true = [theta_A, theta_B]

coin_A = sts.bernoulli(theta_A)
coin_B = sts.bernoulli(theta_B)

coins = [coin_A, coin_B]
```

In [18]:
```
zs = np.array([0, 0, 1, 0, 1])
zs_bool = zs.astype(bool)
xs = np.array([coins[coin].rvs(n).sum() for coin in zs])
```

In [19]:
```
ml_A = xs[~zs_bool].sum() / (3 * n)
ml_B = xs[zs_bool].sum() / (2 * n)
ml_A, ml_B
```

Out[19]:
```
(0.794, 0.3445)
```

Problem

Consider two coins, A and B, with different probabilities of success θ_A and θ_B. The experiment is as follows: we randomly choose a coin, then flip it n times and record the number of successes and failures.

But if we don't record the coin we used, we have missing data and the problem of estimating θ is harder to solve. One way to solve it is to use EM algorithm.

We add latent variable w representing the probability of a sample being generated from coin A. Then we will look at the numbers of samples by coin A as:

$$\#A = w\sum_i x_i$$

Denote $X = \sum_i x_i$. Likelihood of the model is:

$$p(X|w, \theta) = \prod_{i=0}^{n} p_0^{wX} p_1^{(1-w)X}$$

Prior distribution is:

$$q(w|\theta) = Be(w)$$

The posterior distribution of w is:

$$p(w|X, \theta) = \frac{p(X|w,\theta)q(w|\theta)}{\sum_w p(X|w,\theta)q(w|\theta)} = \frac{p(X|w,\theta)}{\sum_w p(X|w,\theta)}$$

So, E-step is to set $w = q(w|\theta) = p(w|X, \theta)$. The M-step is to set θ as MLE under fixed w, so p_0 is the average of the samples with w and p_1 is the average of the samples $(1 - w)$.

In [20]:
```
def em(xs, thetas, max_iter=100, tol=1e-6):
    """Expectation-maximization for coin sample problem."""

    ll_old = -np.infty
    for i in range(max_iter):
        ll = np.array([np.sum(xs * np.log(theta), axis=1) for theta in thetas])
        lik = np.exp(ll)
        # E-step
        ws = lik/lik.sum(0)
        # M-step
        vs = np.array([w[:, None] * xs for w in ws])
        thetas = np.array([v.sum(0)/v.sum() for v in vs])
        ll_new = np.sum([w*l for w, l in zip(ws, ll)])
        if np.abs(ll_new - ll_old) < tol:
            break
        ll_old = ll_new
    return i, thetas, ll_new
```

In [38]:
```
np.random.seed(1234)

n = 100
p0 = 0.8 # 0.51
p1 = 0.7 # 0.53
xs = np.concatenate([np.random.binomial(n, p0, int(n/2)), np.random.binomial(n, p1, int(n/2))])
xs = np.column_stack([xs, n-xs])
np.random.shuffle(xs)
```

In [34]:
```
st_point = np.random.random((2,1))
st_point = np.column_stack([st_point, 1-st_point])
```

In [35]:
```
st_point
```

Out[35]:
```
array([[0.3573748 , 0.6426252 ],
       [0.63721697, 0.36278303]])
```

In [36]:
```
results = [em(xs, st_point, max_iter=10000) for i in range(10)]
i, thetas, ll = sorted(results, key=lambda x: x[-1])[-1]
print(i)
for theta in thetas:
    print(theta)
print(ll)
```

```
22
[0.70051739 0.29948261]
[0.7934922 0.2065078]
-5585.5899811092095
```