메모리, 계산 효율적 딥러닝
Beta-Bernoulli Dropout

이하연
MLAI Lab.
19. 07. 24

들어가기 전에..

Beta-Bernoulli Dropout (BBDrop)은
수학적 증명에 기반하여 네트워크의 sparsity를 유도했습니다.

→ 코드를 보기 전 짚고 넘어가야 할 수학적 증명이 있습니다.

→ 내용이 어렵게 느껴질 수 있습니다.

→ 다행히 첫번째 실습과 접근 방법이 비슷합니다

→ 실습도 쉽게 구성하였습니다. 차근차근 같이 해봅시다!

Beta-Bernoulli Dropout

Beta-Bernoulli 분포로부터 추출된 확률에 따라 네트워크의 뉴런을 드랍 시키자!

↓

성능은 유지하면서 **네트워크가 Sparse**해진다!

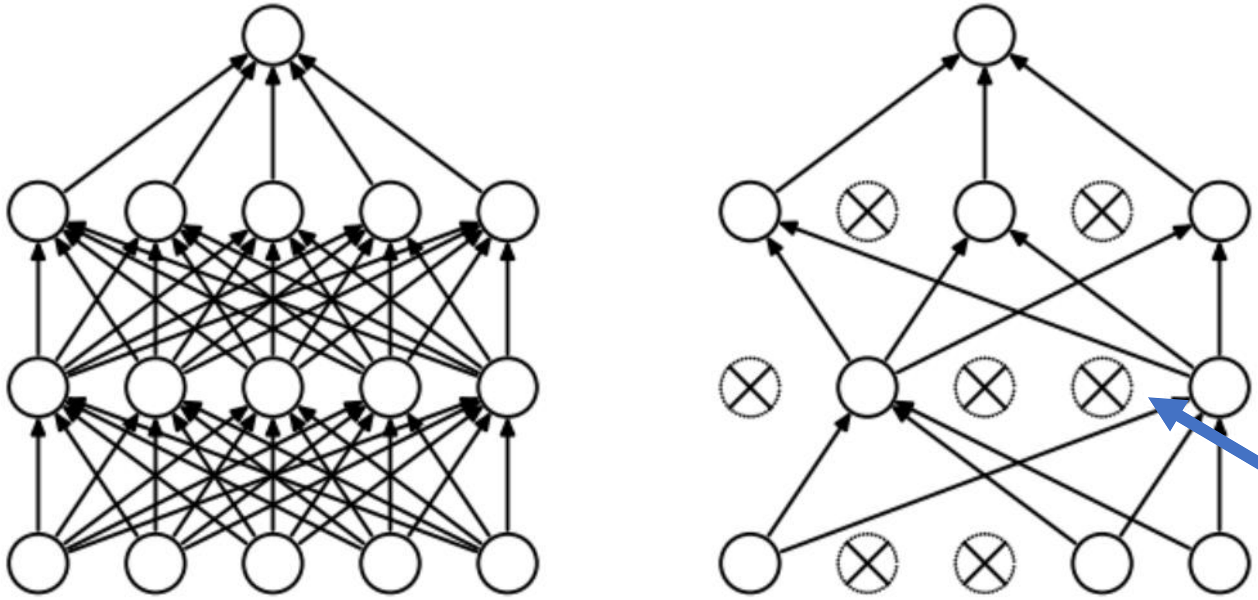
Dropout은 알지알지

Beta-Bernoulli 분포가 뭔데!

이 분포를 이용하면 어떻게
네트워크가 sparse해지는데!!

Dropout

랜덤하게 네트워크의 노드를 드랍시켜 오버피팅을 방지하고 학습을 일반화 시킴.



(a) Standard Neural Net   (b) After applying dropout.

$$z_k \sim Ber(p)$$

$$\begin{cases} z_k = 1, & p \\ z_k = 0, & 1-p \end{cases}$$

$$\boxed{w_k} * \boxed{z_k}$$

node   dropout mask

Beta-Bernoulli Distribution (BB 분포)

Beta 분포의 확률 밀도 함수 (pdf)

- 두 개의 파라메터, $\alpha > 0, \beta > 0$를 갖는 분포

$$beta(\alpha, \beta) = \begin{cases} \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}, & p \in [0,1] \\ 0, & otherwise \end{cases}$$

$B(\alpha, \beta)$: 베타함수

- pdf의 적분값이 1이 나오도록 하는 역할

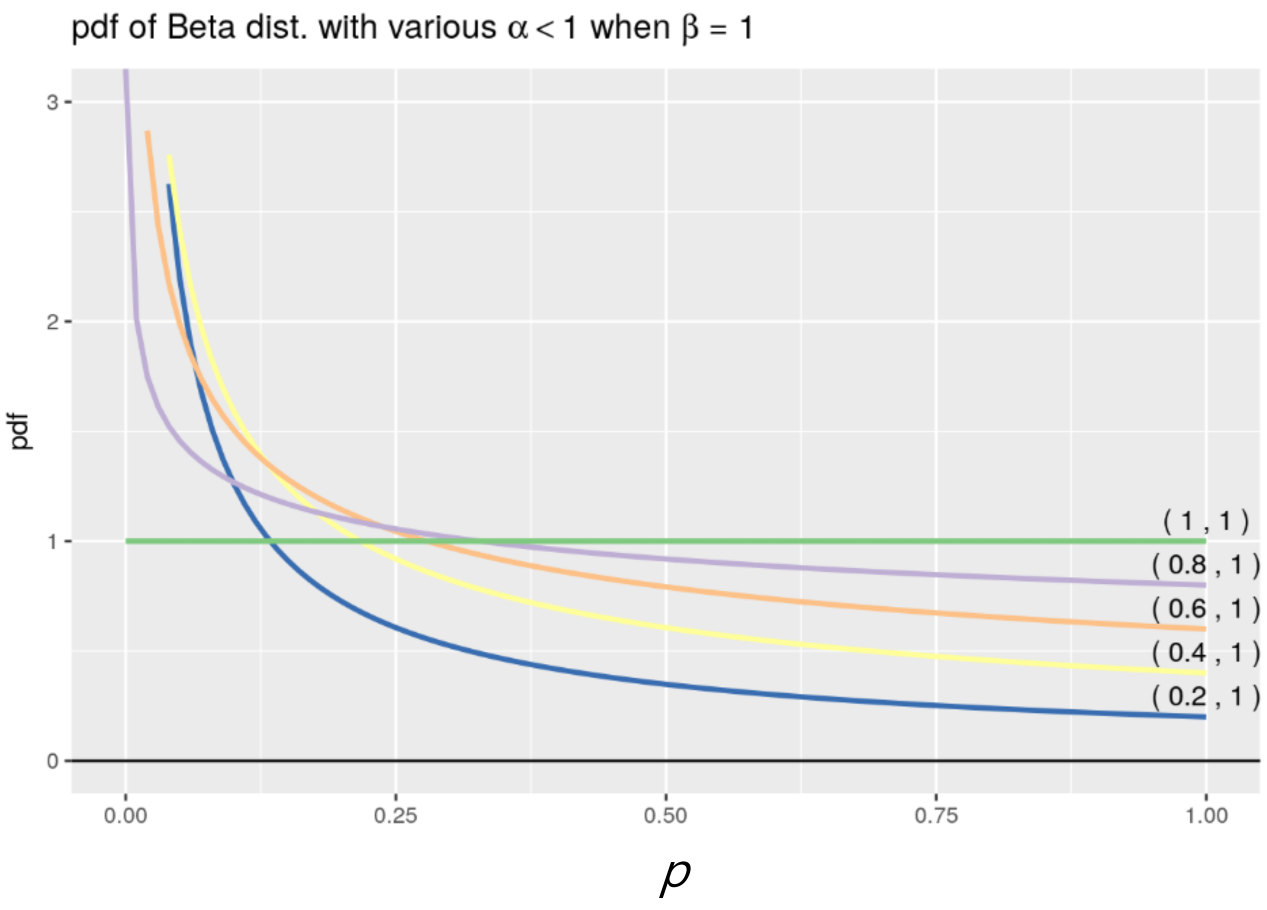
$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

Beta-Bernoulli Distribution (BB 분포)

다양한 α,β에 따른 Bata 분포

$$beta(\alpha,\beta) = \begin{cases} \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}, & p \in [0,1] \\ 0, & otherwise \end{cases}$$

$\alpha < 1,\ \beta = 1$
그래프가 왼쪽으로 치우침
(sparsity-inducing prior)



Beta-**Bernoulli** Distribution (BB 분포)

$$z \sim Ber(p)$$

Beta 분포로 부터 추출

$$\begin{cases} z = 1, & p \\ z = 0, & q = 1 - p \end{cases}$$

$$Ber(p) = p^z (1-p)^{1-z}$$

BB 분포와 Sparsification의 상관관계

Indian Buffet Processes (IBP)[1]

Latent Feature Model

$$\mathbf{d}_n = f(\mathbf{W}\mathbf{z}_n) = f\left(\sum_{k=1}^{K} z_{n,k}\mathbf{w}_k\right)$$

binary mask 1 또는 0 　 Latent feature

어떤 데이터 $\boldsymbol{d}$도 latent features $\boldsymbol{w}_{\boldsymbol{k}}$의 조합으로 생성될 수 있다.

1. [Griffiths & Ghahramani] Infinite latent feature models and the Indian buffet process. In NIPS, 2005.

BB 분포와 Sparsification의 상관관계

Indian Buffet Processes (IBP)[1]

Latent Feature Model

$$\mathbf{d}_n = f(\mathbf{W}\mathbf{z}_n) = f\left(\sum_{k=1}^{K} z_{n,k}\mathbf{w}_k\right)$$

IBP는 binary matrix $\boldsymbol{Z} \in \{\mathbf{0},\mathbf{1}\}^{N\times K}$를 생성

$$\begin{array}{cc} & K \text{ features} \\ N \text{ data points} & \begin{bmatrix} z_{1,1} & \cdots & z_{1,k} \\ \vdots & \ddots & \vdots \\ z_{N,1} & \cdots & z_{n,k} \end{bmatrix} \end{array}$$

주어진 데이터 셋에 따라 **적응적으로 행렬의 열의 개수를 조정** 할 수 있다.

1. [Griffiths & Ghahramani] Infinite latent feature models and the Indian buffet process. In NIPS, 2005.

BB 분포와 Sparsification의 상관관계

Indian Buffet Processes (IBP)

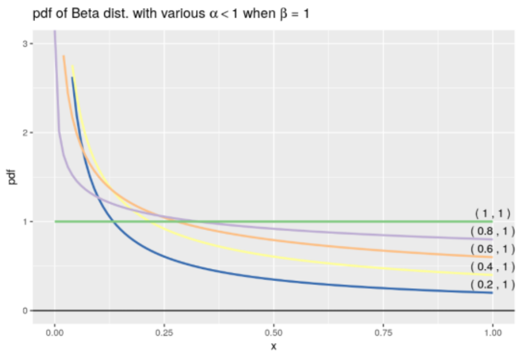
- IBP의 행렬은 BB 분포를 통해 얻어질 수 있다.

$$\begin{bmatrix} z_{1,1} & \cdots & z_{1,k} \\ \vdots & \ddots & \vdots \\ z_{N,1} & \cdots & z_{n,k} \end{bmatrix}$$

$beta(\alpha,\beta)$

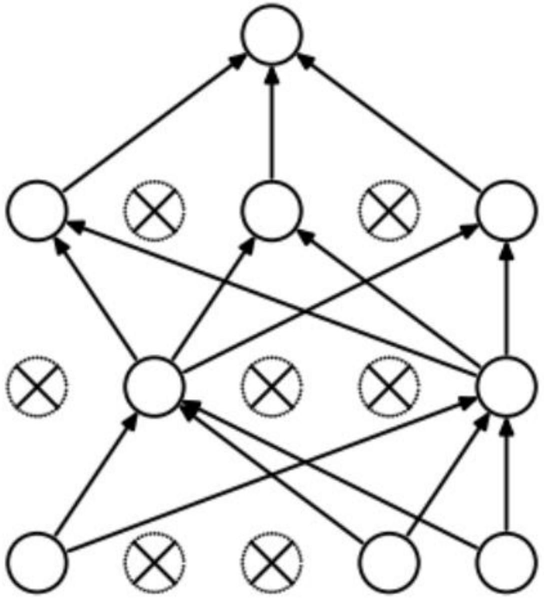
$$\boxed{\pi_k \sim \text{beta}(\alpha/K, 1), \quad z_{n,k} \sim \text{Ber}(\pi_k), \quad K \to \infty.}$$

- **BB 분포는 행렬 z의 sparsity를 발생**시킨다.
- K $\to \infty$, $z = 1$인 원소의 개수는 $N\alpha$로 수렴한다.
- N은 데이터 개수, α는 sparsity level을 조정하는 하이퍼파라메터



BB 분포와 Sparsification의 상관관계

Indian Buffet Processes (IBP)[1]



(b) After applying dropout.

Latent Feature Model

$$\mathbf{d}_n = f(\mathbf{W}\mathbf{z}_n) = f\left(\sum_{k=1}^{K} z_{n,k}\mathbf{w}_k\right)$$

dropout mask

node
(inputs or
Intermediate
Feature units)

BB 분포를 따르는 IBP를 dropout이 있는 네트워크에 적용시키면, 네트워크를 sparse하게 만들 수 있음.

1. [Griffiths & Ghahramani] Infinite latent feature models and the Indian buffet process. In NIPS, 2005.

Variational Inference

사후 확률 (posterior) ∝ 사전 확률 (prior) × 우도 (likelihood)

$$p(\mathbf{W}|\mathcal{D}) \propto p(\mathbf{W}) \times p(\mathcal{D}|\mathbf{W})$$

$$p(\mathcal{D}|\mathbf{W}) \downarrow p(\mathbf{y}|\mathbf{f}(\mathbf{x};\mathbf{W}))$$

데이터를 고려한 후에
자신이 이전에 가지고 있던 사전 확률 분포를 수정하여
사후 확률 분포를 얻음.

Variational Inference

$$\underbrace{p(\mathbf{y}_*|\mathbf{x}_*, \mathcal{D})}_{\text{예측 기대 값}} = \int \underbrace{p(\mathbf{y}_*|\mathbf{f}(\mathbf{x}_*; \mathbf{W}))}_{\substack{\text{예측 값} \\ \text{(prediction)}}} \underbrace{p(\mathbf{W}|\mathcal{D})}_{\substack{\text{사후 확률 분포} \\ \text{(posterior)}}} \mathrm{d}\mathbf{W}$$

$p(\mathcal{D})$: intractable → approximate $p(\mathbf{W}|\mathcal{D})$ as $q(\mathbf{W}; \phi)$

Variational Inference

minimize $\mathrm{D_{KL}}[q(\mathbf{W};\phi)\|p(\mathbf{W}|\mathcal{D})]$

≈ maximize ELBO

$$\mathcal{L}(\phi) = \sum_{n=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{W}))] \quad \leftarrow \text{expected log-likelihood}$$

$$- \mathrm{D_{KL}}[q(\mathbf{W};\phi)\|p(\mathbf{W})], \quad \leftarrow \text{regularization}$$

Maximum Likelihood와 cross-entropy

$$\sum_{n=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{W}))]$$ ← expected log-likelihood

$$\mathcal{L}(\phi) = \sum_{n=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n;\mathbf{W}))] - \mathrm{D}_{\mathrm{KL}}[q(\mathbf{W};\phi)\|p(\mathbf{W})],$$

Maximize log-likelihood = Minimize cross entropy + softmax

Prediction 확률을 최대화

$$-\sum_{x} P(x) log Q(x)$$

label Softmax(logits)

In Cell *Define the functions and utils* [Cell link]

```
def cross_entropy(logits, labels):
    return tf.losses.softmax_cross_entropy(logits=logits, onehot_labels=labels)
```

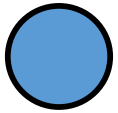
In Cell *Create models* [Cell link]

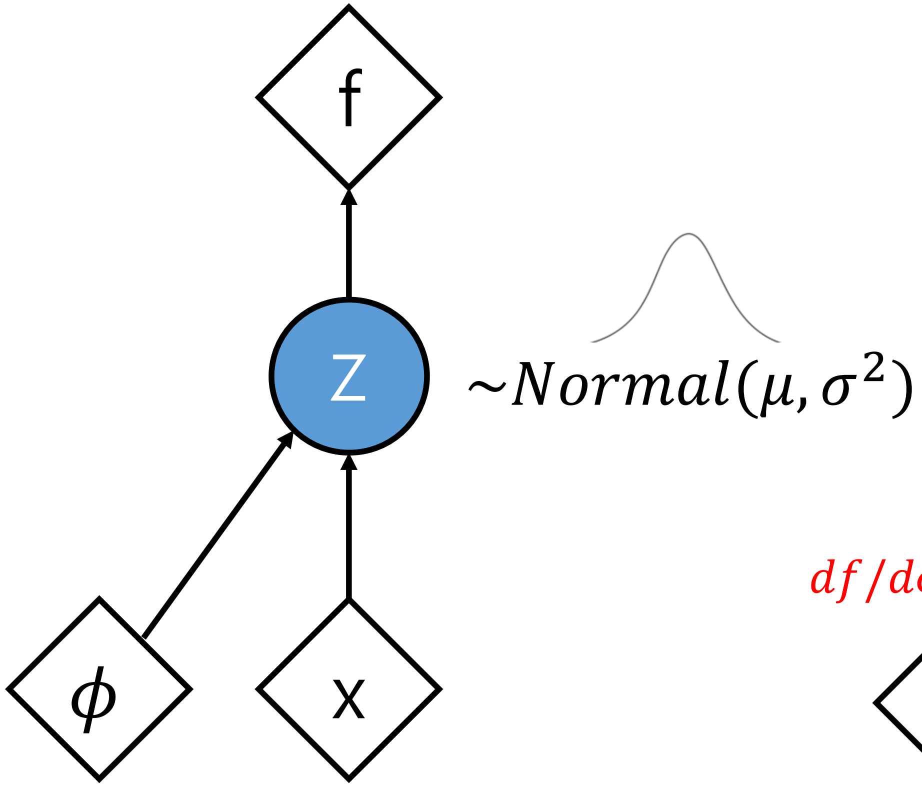
```
net['cent'] = cross_entropy(x, y)
```

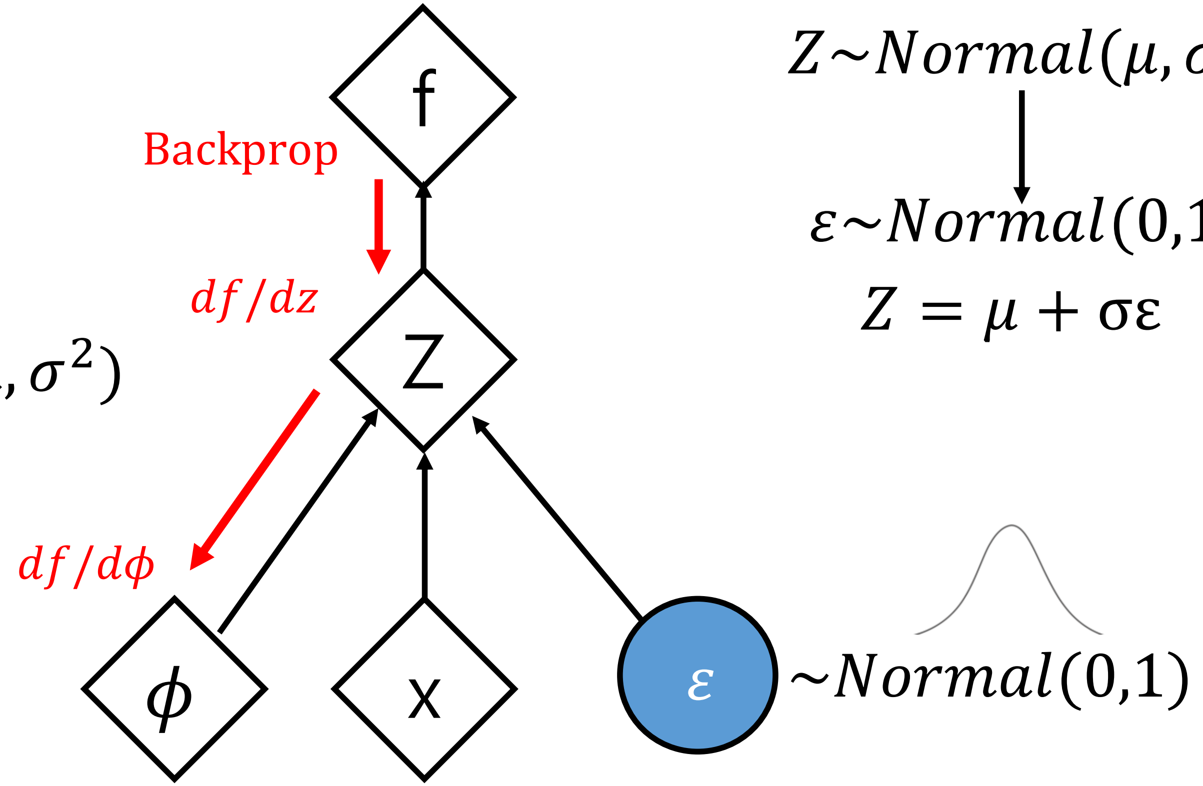
In Cell *Lets' run the code!* [Cell link]

```
loss = net['cent'] + tf.add_n(net['kl'])/float(N) + net['wd']
```

Reparametrization Trick

미분이 안되는 샘플링 된 노드  를 미분 가능하게 바꿔 Back-propagation을 가능하게 한다.





$$Z \sim Normal(\mu, \sigma^2)$$

$$\downarrow$$

$$\varepsilon \sim Normal(0,1)$$

$$Z = \mu + \sigma\varepsilon$$

Variational Inference

Mini-batch 단위로 샘플링

$$\sum_{n=1}^{N} \nabla_{\phi} \mathbb{E}_q [\log p(\mathbf{y}_n | \mathbf{f}(\mathbf{x}_n; \mathbf{W}))]$$

$$\approx \frac{N}{|B|} \sum_{n \in B} \nabla_{\phi} \mathbb{E}_q [\log p(\mathbf{y}_n | \mathbf{f}(\mathbf{x}_n; \mathbf{W}))]$$

Variational Inference

+ Reparametrization trick
+ Mini-batch sampling

$$\nabla_{\phi}\mathcal{L}(\phi) \xrightarrow{\text{update}} \phi$$

$$q(\mathbf{W};\phi)$$

BBDrop을 VI을 이용하여 어떻게 최적화하는지 한편 살펴봅시다!

VI를 이용한 최적화

사전 정보(prior):
W는 Gaussian 분포를 따른다.

$$\mathbf{W} \sim \mathcal{N}(\mathbf{0}, \lambda\mathbf{I}) \qquad \boldsymbol{\pi} \sim \prod_{k=1}^{K} \mathrm{beta}(\pi_k; \alpha/K, 1),$$

$$\mathbf{z}_n|\boldsymbol{\pi} \sim \prod_{k=1}^{K} \mathrm{Ber}(z_{n,k}; \pi_k), \quad \widetilde{\mathbf{W}}_n = \mathbf{z}_n \otimes \mathbf{W}.$$

BB분포를 따라 z_n을 뽑으면, 0이 많이 뽑혀서 네트워크가 sparse해짐.

VI를 이용한 최적화

Goal

주어진 데이터 $\mathcal{D}$를 관찰 한 후의 사후 확률 분포

$$p(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi} | \mathcal{D})$$

를 잘 근사한

Variational 분포 구하기

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi} | \mathbf{X})$$

VI를 이용한 최적화

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi} | \mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \underbrace{\prod_{k=1}^{K} q(\pi_k)} \underbrace{\prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k} | \pi_k)}$$

$\pi_k \sim Beta(\alpha/\mathrm{K}, 1)$
근사

$z_{n,k} | \pi_k \sim Ber(\pi_k)$
근사

VI를 이용한 최적화

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

$q(\pi_k)$ 를 구하기 위해 **Kumaraswamy 분포**[2] 사용.

$$q(\pi_k; a_k, b_k) = a_k b_k \pi_k^{a_k - 1} (1 - \pi_k^{a_k})^{b_k - 1}$$

비교) $beta(\alpha, \beta) = \begin{cases} \frac{\pi^{\alpha-1}(1-\pi)^{\beta-1}}{B(\alpha,\beta)}, & \pi \in [0,1] \\ 0, & otherwise \end{cases}$

$Beta$ 분포와 닮았으면서 매개 변수를 구하기 쉽게 바꿀 수 있음 (reparametrizable).

2. [Kumaraswamy] A generalized probability density function for double-bounded random processes. Journal of Hydrology, 1980.

VI를 이용한 최적화

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} {\color{red} q(\pi_k)} \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

$\pi_k \sim q(\pi_k)$: Kumaraswamy 분포를 따르는 난수 생성기

↓

하지만, 텐서플로우는 Kumaraswamy 분포를 이용한 난수 생성기를 제공 X

↓

Inverse Transform Sampling[3]를 이용

↓

Kumaraswamy분포의 누적 분포 함수(CDF)의 역함수에 $u \sim unif([0,1])$를 인풋으로 제공

$$\pi_k(u; a_k, b_k) = \left(1 - u^{\frac{1}{b_k}}\right)^{\frac{1}{a_k}}, \quad u \sim \mathrm{unif}([0,1])$$

3. https://en.wikipedia.org/wiki/Inverse_transform_sampling

VI를 이용한 최적화

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

Kumaraswamy분포의 누적 분포 함수(CDF)의 역함수에 $u \sim unif([0,1])$를 인풋으로 제공

$$\pi_k(u; a_k, b_k) = \left(1 - u^{\frac{1}{b_k}}\right)^{\frac{1}{a_k}}, \quad u \sim \mathrm{unif}([0,1])$$

In Cell *Define the Beta-Bernoulli Dropout* [Cell link]

```
pi = (1 - tf.random_uniform([K])**(1/b))**(1/a)
```

3. https://en.wikipedia.org/wiki/Inverse_transform_sampling

VI를 이용한 최적화

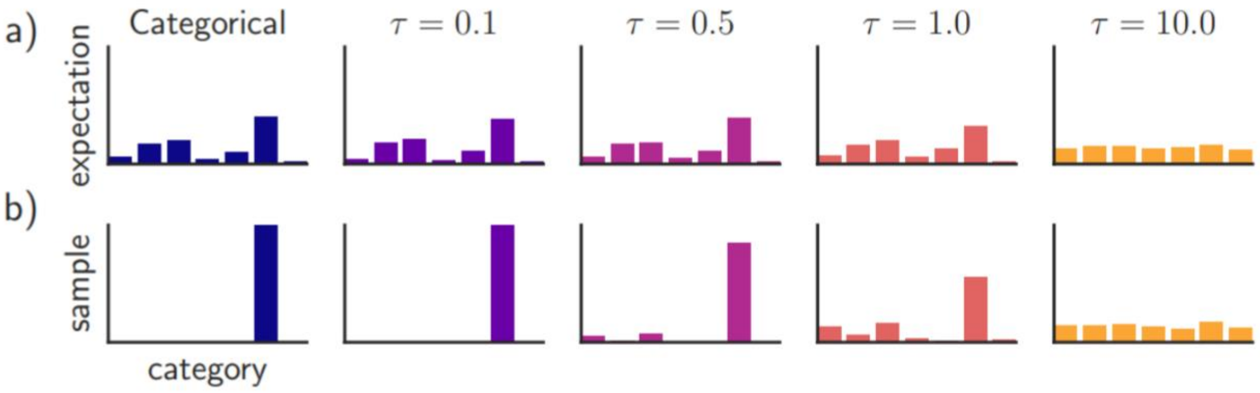
$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi} | \mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k} | \pi_k)$$

$q(z_{n,k}|\pi_k)$ 는 *continuous* **relaxation**[4]를 이용하여 $Bernoulli$ 분포 근사

$$z_k = \mathrm{sgm}\left(\frac{1}{\tau}\left(\log\frac{\pi_k}{1-\pi_k} + \log\frac{u}{1-u}\right)\right)$$

$$\mathrm{sgm}(x) = \frac{1}{1+e^{-x}}$$

$$u \sim \mathrm{unif}([0,1])$$



$q \to discrete,\ \tau \to 0$

$q \to constant\ distribution,\ \tau \to \infty$

4. [Maddison et al.] The concrete distribution: A continuous relaxation of discrete random variables, ICLR17.

증명 → 레퍼런스 BinConcrete 참고.

VI를 이용한 최적화

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\widehat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

$q(z_{n,k}|\pi_k)$ 는 *continuous* **relaxation**[4]를 이용하여 $Bernoulli$ 분포 근사

$$z_k = \mathrm{sgm}\left(\frac{1}{\tau}\left(\log\frac{\pi_k}{1-\pi_k} + \log\frac{u}{1-u}\right)\right) \qquad \mathrm{sgm}(x) = \frac{1}{1+e^{-x}}$$

$$u \sim \mathrm{unif}([0,1])$$

In Cell *Define the Beta-Bernoulli Dropout* [Cell link]

```
z = RelaxedBernoulli(tau, logits=logit(pi)).sample(N)
```

```
return x*z
```

4. [Maddison et al.] The concrete distribution: A continuous relaxation of discrete random variables, ICLR17.

증명 → 레퍼런스 BinConcrete 참고.

VI를 이용한 최적화

Prior와 variational distribution의 KL-divergence[5]

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\hat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

$$\mathrm{D_{KL}}[q(\mathbf{Z}, \boldsymbol{\pi}) \| p(\mathbf{Z}, \boldsymbol{\pi})] = \sum_{k=1}^{K} \left\{ \frac{a_k - \alpha/K}{a_k} \left(-\gamma - \Psi(b_k) - \frac{1}{b_k} \right) + \log \frac{a_k b_k}{\alpha/K} - \frac{b_k - 1}{b_k} \right\},$$

γ: Euler-Mascheroni constant

Ψ :digamma function

Kumaraswamy분포의 파라메터

$$\pi_k(u; a_k, b_k) = (1 - u^{\frac{1}{b_k}})^{\frac{1}{a_k}}, \quad u \sim \mathrm{unif}([0,1])$$

5. [Nalisnick and Smyth] Stick-breaking variational autoencoders, ICLR17.

VI를 이용한 최적화

Prior와 variational distribution의 KL-divergence[5]

$$q(\mathbf{W}, \mathbf{Z}, \boldsymbol{\pi}|\mathbf{X}) = \delta_{\hat{\mathbf{W}}}(\mathbf{W}) \prod_{k=1}^{K} q(\pi_k) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{n,k}|\pi_k)$$

$$\begin{aligned} &\mathrm{D_{KL}}[q(\mathbf{Z}, \boldsymbol{\pi}) \| p(\mathbf{Z}, \boldsymbol{\pi})] \\ &= \sum_{k=1}^{K} \left\{ \frac{a_k - \alpha/K}{a_k} \left(-\gamma - \Psi(b_k) - \frac{1}{b_k} \right) \right. \\ &\qquad \left. + \log \frac{a_k b_k}{\alpha/K} - \frac{b_k - 1}{b_k} \right\}, \end{aligned}$$

In Cell *Define the Beta-Bernoulli Dropout* [Cell link]

```
_digamma = digamma_approx
kl = (a-alpha)/a * (-Euler - _digamma(b) - 1/b) ₩
    + log(a*b) - log(alpha) - (b-1)/b
```

5. [Nalisnick and Smyth] Stick-breaking variational autoencoders, ICLR17.

VI를 이용한 최적화

인풋 x_*에 대한 prediction:

$$\begin{aligned} & p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}, \mathbf{W}) \\ & = \mathbb{E}_{p(\mathbf{z}_*, \boldsymbol{\pi}, \mathbf{W} | \mathcal{D})}[p(\mathbf{y}_* | \mathbf{f}(\mathbf{x}_*; \mathbf{z}_* \otimes \mathbf{W}))] \\ & \approx \mathbb{E}_{q(\mathbf{z}_*, \boldsymbol{\pi})}[p(\mathbf{y}_* | \mathbf{f}(\mathbf{x}_*; \mathbf{z}_* \otimes \widehat{\mathbf{W}}))], \end{aligned}$$

Naïve approach in practice:

$$p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}, \mathbf{W}) \approx p(\mathbf{y}_* | \mathbf{f}(\mathbf{x}_*; \mathbb{E}_q[\mathbf{z}_*] \otimes \widehat{\mathbf{W}})),$$

where

$$\mathbb{E}_q[z_{*,k}] = \mathbb{E}_{q(\pi_k)}[\pi_k],$$

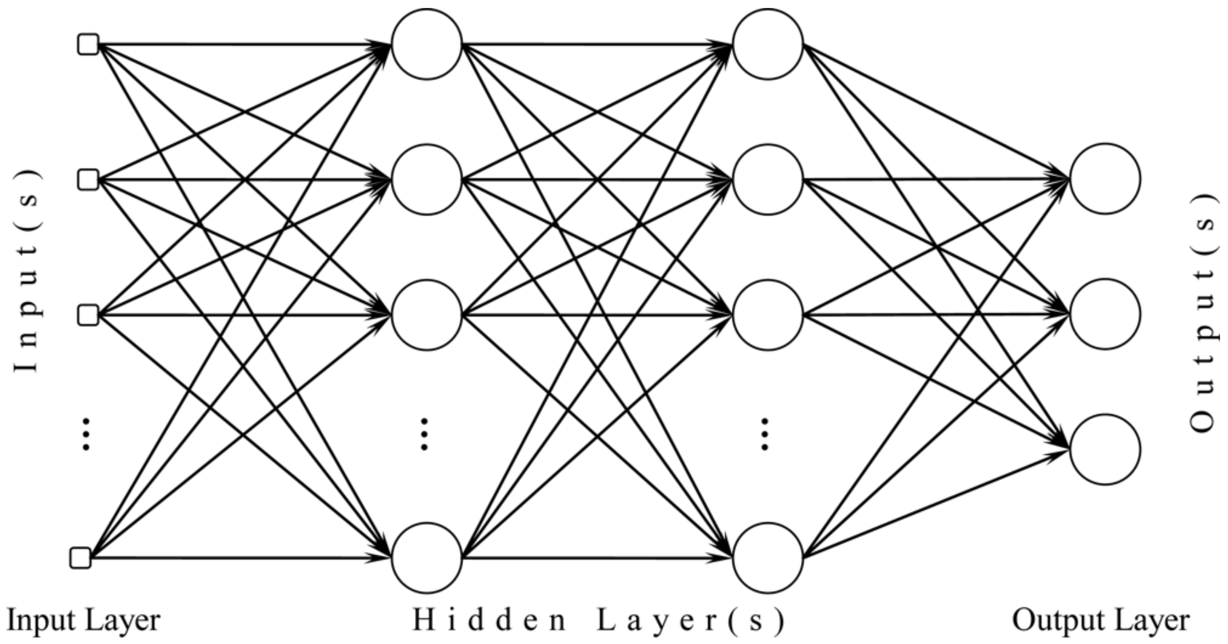
$$\mathbb{E}_{q(\pi_k)}[\pi_k] = \frac{b_k \Gamma(1 + a_k^{-1}) \Gamma(b_k)}{\Gamma(1 + a_k^{-1} + b_k)}.$$

In Cell *Define the Beta-Bernoulli Dropout* [Cell link]

```
pi = b*tf.exp(lgamma(1+1/a) + lgamma(b) - lgamma(1+1/a+b))
```

데이터셋, 네트워크

MNIST



784 **500** **300** **10**

훈련

https://github.com/HayeonLee/sparsification_samsung

```
[ ]  train()

Epoch 1 start, learning rate 0.010000
train: epoch 1, (5.304 secs), cent 0.413326, acc 0.877817
test: epoch 1, (5.596 secs), cent 0.110209, acc 0.967600
kl: [4585.8677, 3876.8252, 2495.3503]
n_active: [523, 469, 297]

Epoch 2 start, learning rate 0.010000
train: epoch 2, (4.134 secs), cent 0.164370, acc 0.949483
test: epoch 2, (4.369 secs), cent 0.083183, acc 0.973700
kl: [4452.902, 3971.1667, 2566.4827]
n_active: [511, 467, 297]

Epoch 3 start, learning rate 0.010000
train: epoch 3, (4.131 secs), cent 0.125859, acc 0.960067
test: epoch 3, (4.362 secs), cent 0.067286, acc 0.978200
kl: [4552.4395, 4077.387, 2617.6377]
n_active: [503, 465, 296]
```

https://github.com/HayeonLee/sparsification_samsung/blob/1823e44e1231cc56d3cec8c887742c1a5ec2a79c/bbdropout_samsung.ipynb

테스트

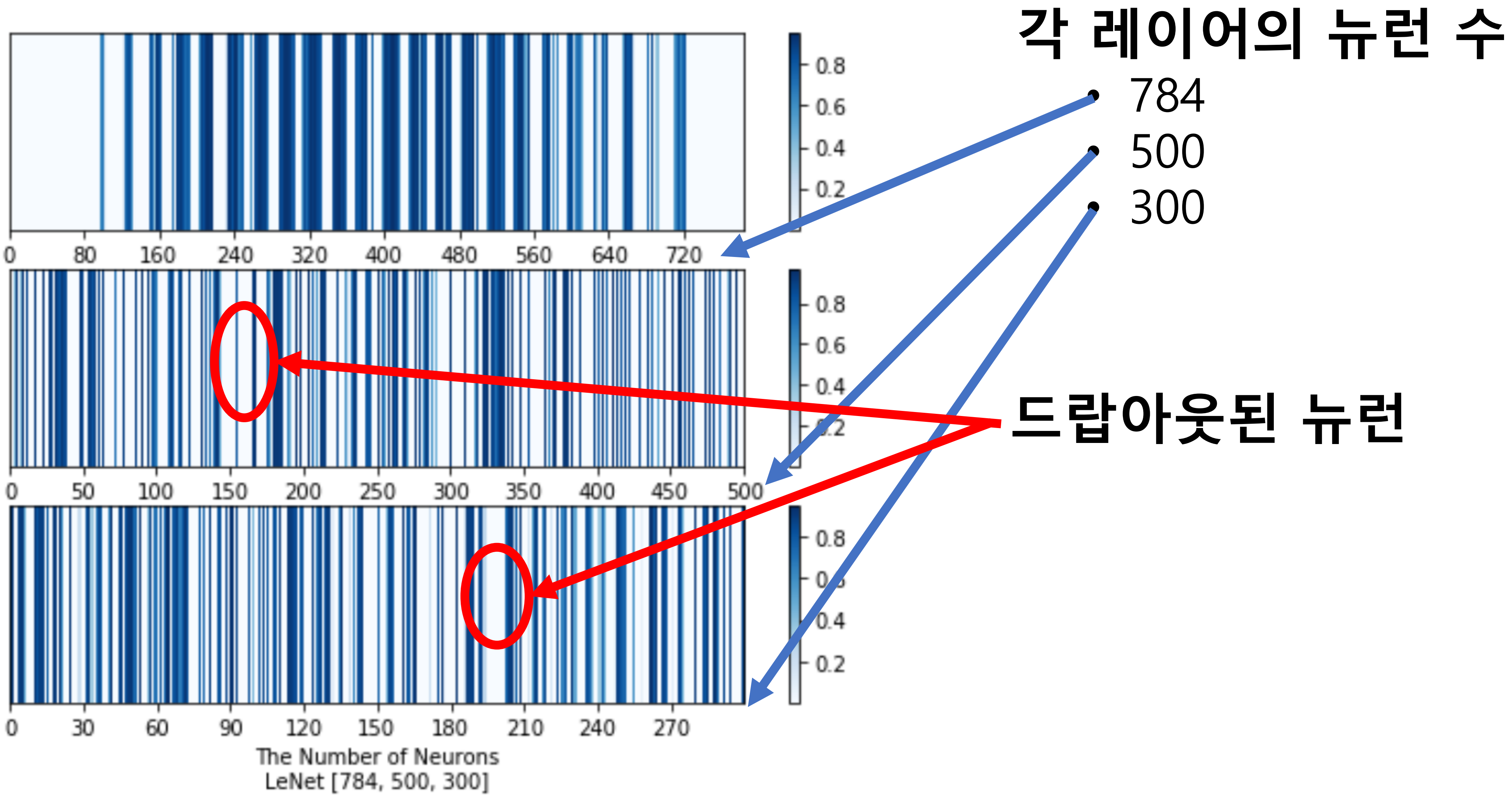
```
[21] def test():
         sess = tf.Session()
         saver = tf.train.Saver(tnet['weights']+tnet['qpi_vars'])
         saver.restore(sess, os.path.join(savedir, 'model'))
         logger = Accumulator('cent', 'acc')
         to_run = [tnet['cent'], tnet['acc']]
         for j in range(n_test_batches):
             bx, by = mnist.test.next_batch(batch_size)
             logger.accum(sess.run(to_run, {x:bx, y:by}))
         logger.print_(header='test')

         n_active = sess.run(tnet['n_active'])
         print("The percentage of activated neurons per layer:")
         for na, nl in zip(n_active, [784, 500, 300]):
           print('{}/{} = {:.2f}%'.format(na, nl, float(na)/nl * 100))

     test()
```

```
test: cent 0.041238, acc 0.987600
The percentage of activated neurons per layer
358/784 = 45.66%
209/500 = 41.80%
152/300 = 50.67%
```

시각화



0.8
0.6
0.4
0.2
0
80
160
240
320
400
480
560
640
720
0
50
100
150
200
250
300
350
400
450
500
0
30
60
90
120
150
180
210
240
270
The Number of Neurons
LeNet [784, 500, 300]
각 레이어의 뉴런 수
784
500
300
드랍아웃된 뉴런