Deep Weight Prior

Kotova Darya

May, 2019

Contents

1. Variational inference
2. Reparametrisation trick
3. Variational autoencoder
4. Results
5. Conclusion

Variational Inference: Problem

Goal: transform prior knowledge $p(w)$ to the posterior distribution $p(w|D)$ with Bayes rule:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

The problem is $p(D)$ is unknown $\rightarrow$ $p(w|D)$ is intractable.
Then let's approximate $p(w|D)$ with $q_\theta(w)$.

Variational lower bound

To make it we minimize KL-divergence:

$$D_{KL}(q_\theta(w)||p(w|D)) = \mathbb{E}_{q_\theta} log \frac{q_\theta(w)}{p(w|D)} \to \min_\theta$$

It is equivalent to **variational lower bound maximization**

$$VLB := \mathbb{E}_{q_\theta} \log p(D|w) - D_{KL}(q_\theta(w)||p(w)) \to \max_\theta$$

Reparametrisation trick

$$VLB = \mathbb{E}_{q_\theta} \log p(D|w) - D_{KL}(q_\theta(w)||p(w)) \to \max_\theta$$

$$\nabla_\theta \mathbb{E}_{q_\theta} \log p(D|w) = \int \nabla_\theta q_\theta(w) \log p(D|w) \neq \mathbb{E}_{q_\theta} \nabla_\theta \log p(D|w)$$

So we can not obtain unbiased gradients and perform mini-batching training.

Reparametrisation trick

Idea:
Let's represent q_θ as deterministic differentiable function $f(\theta, \epsilon)$.
For example:

$$\epsilon \sim \mathcal{N}(0,1) \quad \xi \sim \mathcal{N}(\mu, \sigma^2)$$
$$\epsilon = \frac{\xi - \mu}{\sigma} \quad \xi = \mu + \sigma \cdot \epsilon$$

Now we can compute gradients of VLB, where instead of q_θ now is $f(\theta, \epsilon)$.

Variational lower bound

To make it we minimize KL-divergence:

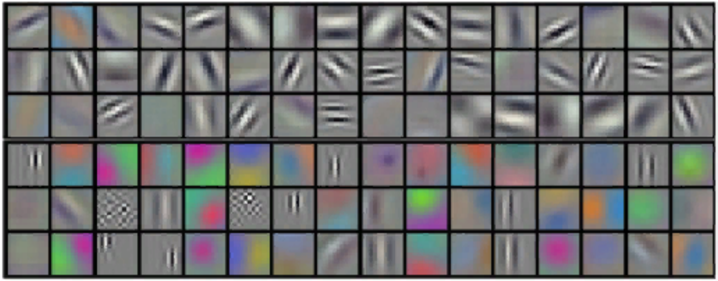
$$D_{KL}(q_\theta(w)||p(w|D)) = \mathbb{E}_{q_\theta} log \frac{q_\theta(w)}{p(w|D)} \to \min_\theta$$

It is equivalent to **variational lower bound maximization**

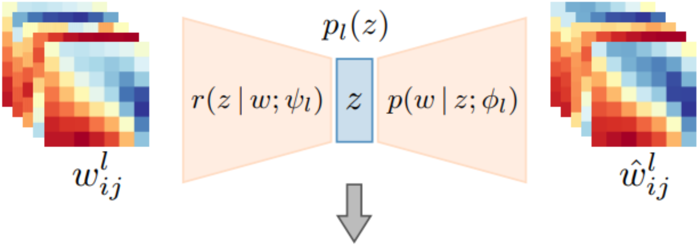
$$VLB = \mathbb{E}_{q_\theta} \log p(D|w) - D_{KL}(q_\theta(w)||p(w)) \to \max_\theta$$

Prior distribution

We suggest that networks learned on the similar datasets will have similar kernel's structures in convolutional layers.



Variational autoencoder



$$\hat{p}_l(w) = \int p(w \mid z; \phi_l) p_l(z)\, dz$$

It helps us to approximate $p(w)$, having weights from already learned networks.

Variational inference with implicit prior distribution

However, we still do not have probability density function of p(w).
VAE also helps to make auxilary bound for VLB:

$$VLB = \mathbb{E}_{q_\theta} \log p(D|w) - D_{KL}(q_\theta(w)||p(w)) =$$

$$\mathbb{E}_{q(w)}\mathbb{E}_{r(z|w)}(log\frac{p(w,z)}{r(z|w)}p(x|w) - log(q(w)) + \mathbb{E}_{q(w)}D_{KL}(r(z|w)||p(z|w)) =$$

$$= L^{aux} + \mathbb{E}_{q(w)}D_{KL}(r(z|w)||p(z|w)) \geq L^{aux}$$

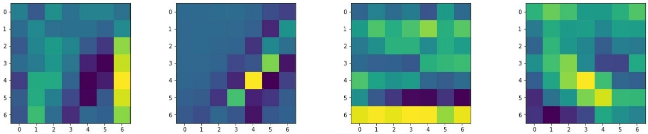
Results

Code and different examples are available on
`https://github.com/DahaKot/Deep-Weight-Prior`

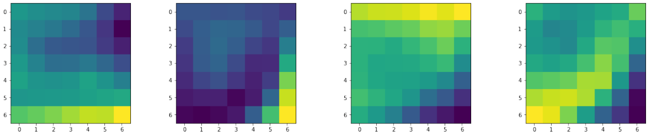
- trained 100 CNNs on notMNIST
- trained vae on source kernels
- used samples from vae for CNN learned on MNIST
- compare performance of CNNs with different initialization

Kernel examples

There are samples of the source kernels:

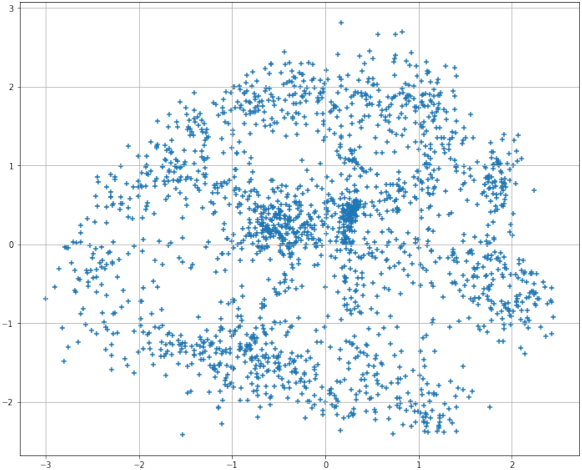


And kernels got from vae:



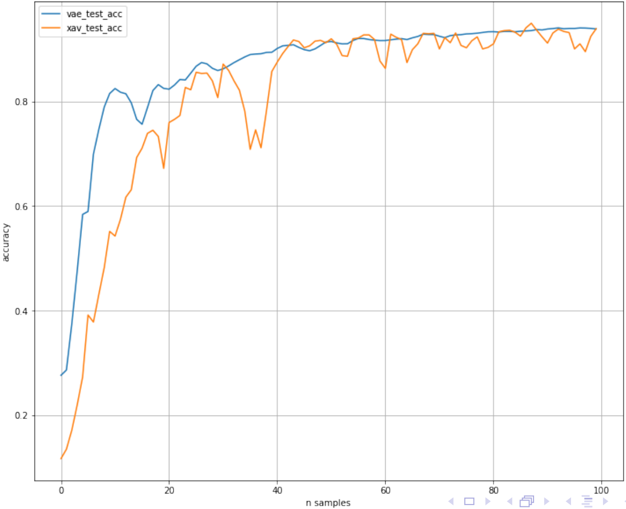
Latent space

Visualization of latent representations of convolutional filters for ConvNet on notMNIST:



Performance comparation

The performance of convolutional network with two different priors: deep weight prior (dwp) and standard normal:

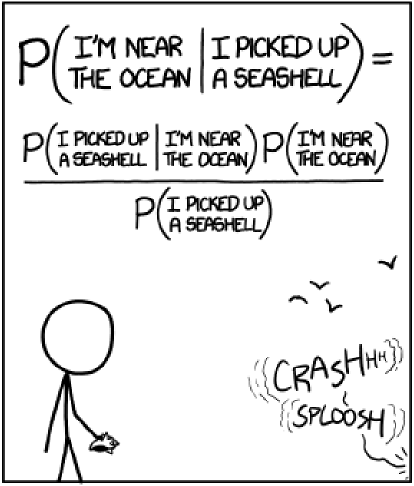


Conclusion

We considered a new way to initilize kernels of convolutional neural networks - deep weight prior:

- Performs better than 'default' random initialization
- Does not need as much memory and computations as initialization with already learned filters

Thank you for attention



STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND *DON'T* HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.

(not sure which meme to choose)