Deep Learning

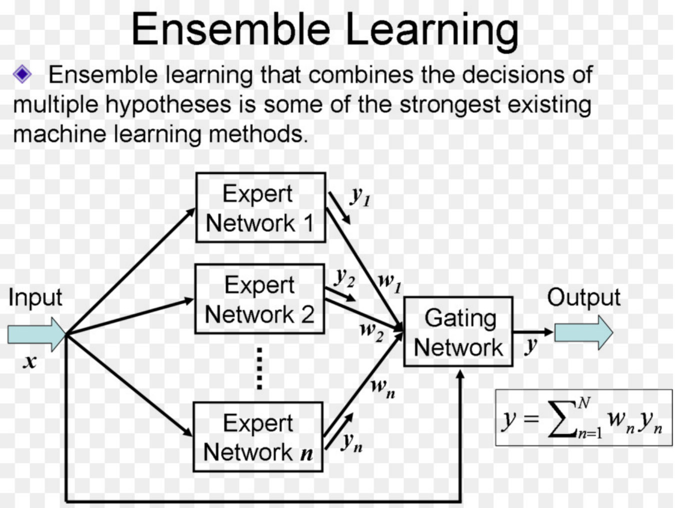
Vazgen Mikayelyan

December 26, 2020

FAST DISCOVERING THE FUTURE

Outline

Ensemble of Neural Networks



Ensemble of Neural Networks

Note that

$$(y - \hat{y})^2 = \left(\sum_{i=1}^{n} w_i y_i - \hat{y} \right)^2$$

Ensemble of Neural Networks

Note that

$$(y - \hat{y})^2 = \left(\sum_{i=1}^{n} w_i y_i - \hat{y}\right)^2 = \left(\sum_{i=1}^{n} w_i \left(y_i - \hat{y}\right)\right)^2$$

Ensemble of Neural Networks

Note that

$$(y-\hat{y})^2 = \left(\sum_{i=1}^{n} w_i y_i - \hat{y}\right)^2 = \left(\sum_{i=1}^{n} w_i \left(y_i - \hat{y}\right)\right)^2 \le \sum_{i=1}^{n} w_i^2 \sum_{i=1}^{n} (y_i - \hat{y})^2 \,.$$

Ensemble of Neural Networks

Note that

$$(y-\hat{y})^2 = \left(\sum_{i=1}^{n} w_i y_i - \hat{y}\right)^2 = \left(\sum_{i=1}^{n} w_i \left(y_i - \hat{y}\right)\right)^2 \leq \sum_{i=1}^{n} w_i^2 \sum_{i=1}^{n} \left(y_i - \hat{y}\right)^2 .$$

Let $w_1 = \ldots = w_n = \frac{1}{n}$, then

$$(y-\hat{y})^2 \leq \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}\right)^2$$

Ensemble of Neural Networks

Note that

$$(y-\hat{y})^2=\left(\sum_{i=1}^{n} w_i y_i-\hat{y}\right)^2=\left(\sum_{i=1}^{n} w_i\left(y_i-\hat{y}\right)\right)^2 \leq \sum_{i=1}^{n} w_i^2 \sum_{i=1}^{n}\left(y_i-\hat{y}\right)^2 .$$

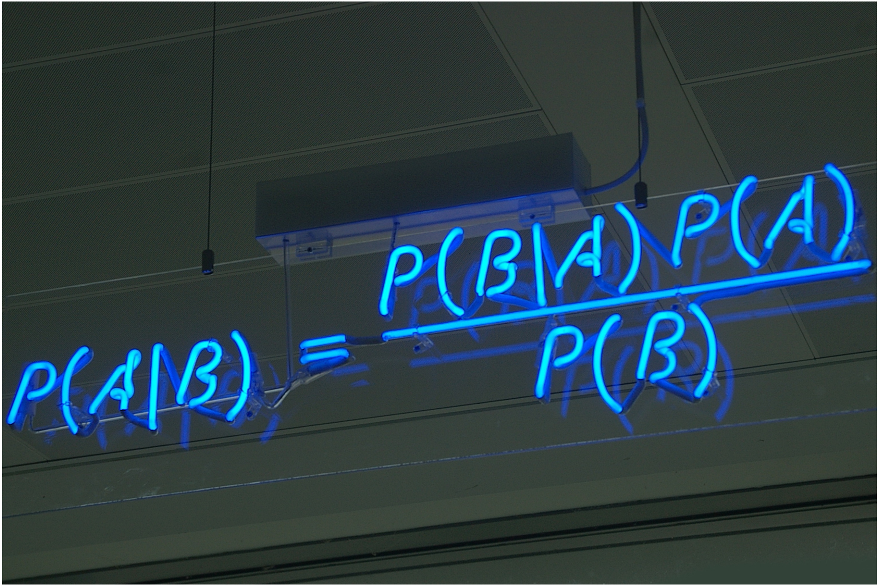
Let $w_1=\ldots=w_n=\dfrac{1}{n}$, then

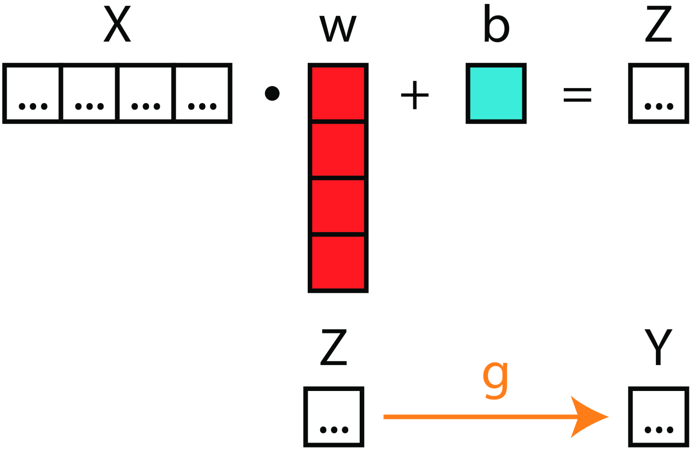
$$(y-\hat{y})^2 \leq \frac{1}{n} \sum_{i=1}^{n}\left(y_i-\hat{y}\right)^2$$

Can we do ensemble learning with infinite number of neural networks?

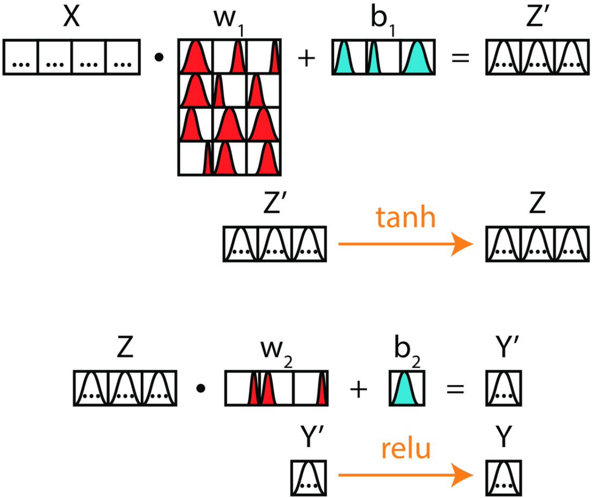
Outline

BNNs




X
w
b
Z
...
...
...
...
...
Z
g
Y
...
...

BNNs



BNNs

Let $\mathcal{D} = \{(x_i, y_i) : i = 1, \ldots, n\}$ be our training data.

BNNs

Let $\mathcal{D} = \{(x_i, y_i) : i = 1, \ldots, n\}$ be our training data. Recall MLE:

$$w^{MLE} = \underset{w}{\operatorname{argmax}}\ p\left(\mathcal{D}|w\right) = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} p\left(y_i|x_i, w\right)$$

$$= \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log p\left(y_i|x_i, w\right)$$

BNNs

Let $\mathcal{D} = \{(x_i, y_i) : i = 1, \ldots, n\}$ be our training data. Recall MLE:

$$w^{MLE} = \underset{w}{\operatorname{argmax}}\ p(\mathcal{D}|w) = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} p(y_i|x_i, w)$$

$$= \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y_i|x_i, w)$$

Here the weights of our model are "fixed", but the data is viewed as a random variable. If we instead view the data as being fixed and the model weights as being a random variable, we can train to maximize the posterior distribution $p(w|\mathcal{D})$:

BNNs

Let $\mathcal{D} = \{(x_i, y_i) : i = 1, \dots, n\}$ be our training data. Recall MLE:

$$w^{MLE} = \underset{w}{\operatorname{argmax}}\ p\left(\mathcal{D}|w\right) = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} p\left(y_i|x_i, w\right)$$

$$= \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log p\left(y_i|x_i, w\right)$$

Here the weights of our model are "fixed", but the data is viewed as a random variable. If we instead view the data as being fixed and the model weights as being a random variable, we can train to maximize the posterior distribution $p\left(w|\mathcal{D}\right)$:

$$w^{MAP} = \underset{w}{\operatorname{argmax}}\ p\left(w|\mathcal{D}\right) = \underset{w}{\operatorname{argmax}}\ \frac{p\left(\mathcal{D}|w\right) p\left(w\right)}{p\left(\mathcal{D}\right)}$$

$$= \underset{w}{\operatorname{argmax}} \left(\log p\left(\mathcal{D}|w\right) + \log p\left(w\right)\right).$$

BNNs

We will construct a new distribution for $q\left(w|\theta\right)$, for approximating $p\left(w|\mathcal{D}\right)$.

We will construct a new distribution for $q\left(w|\theta\right)$, for approximating $p\left(w|\mathcal{D}\right)$. So we need to do the following optimization:

$$\theta^{*} = \underset{\theta}{\operatorname{argmin}}\ KL\left(q\left(w|\theta\right)||p\left(w|\mathcal{D}\right)\right)$$

BNNs

We will construct a new distribution for $q(w|\theta)$, for approximating $p(w|\mathcal{D})$. So we need to do the following optimization:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\ KL\left(q\left(w|\theta\right)||p\left(w|\mathcal{D}\right)\right)$$

$$= \underset{\theta}{\operatorname{argmin}}\left(KL\left(q\left(w|\theta\right)||p\left(w\right)\right) - \mathbb{E}_{q(w|\theta)}\left[\log p\left(\mathcal{D}|w\right)\right]\right)$$

BNNs

We will construct a new distribution for $q(w|\theta)$, for approximating $p(w|\mathcal{D})$. So we need to do the following optimization:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\ KL\left(q\left(w|\theta\right)||p\left(w|\mathcal{D}\right)\right)$$

$$= \underset{\theta}{\operatorname{argmin}} \left(KL\left(q\left(w|\theta\right)||p\left(w\right)\right) - \mathbb{E}_{q(w|\theta)}\left[\log p\left(\mathcal{D}|w\right)\right]\right)$$

We will assume that prior $p(w)$ is mixture of two Gaussians:

$$p\left(w\right) = \prod_j \left(\alpha\mathcal{N}\left(w_j|0,\sigma_1^2\right) + \left(1-\alpha\right)\mathcal{N}\left(w_j|0,\sigma_2^2\right)\right)$$

where the first mixture component of the prior is given a larger variance than the second: $\sigma_1 > \sigma_2$.

BNNs

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Let $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \tag{3}$$

6. Calculate the gradient with respect to the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \tag{4}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \Delta_\mu \tag{5}$$

$$\rho \leftarrow \rho - \alpha \Delta_\rho. \tag{6}$$

Outline

Facial Recognition System

- Suppose there is an organisation and it wants a facial recognition system to allow access to the building for its employees and you are given the task of building such a system.

Facial Recognition System

- Suppose there is an organisation and it wants a facial recognition system to allow access to the building for its employees and you are given the task of building such a system.
- The problem with this task is that the organisation might not have more than ten images for each of the employee.

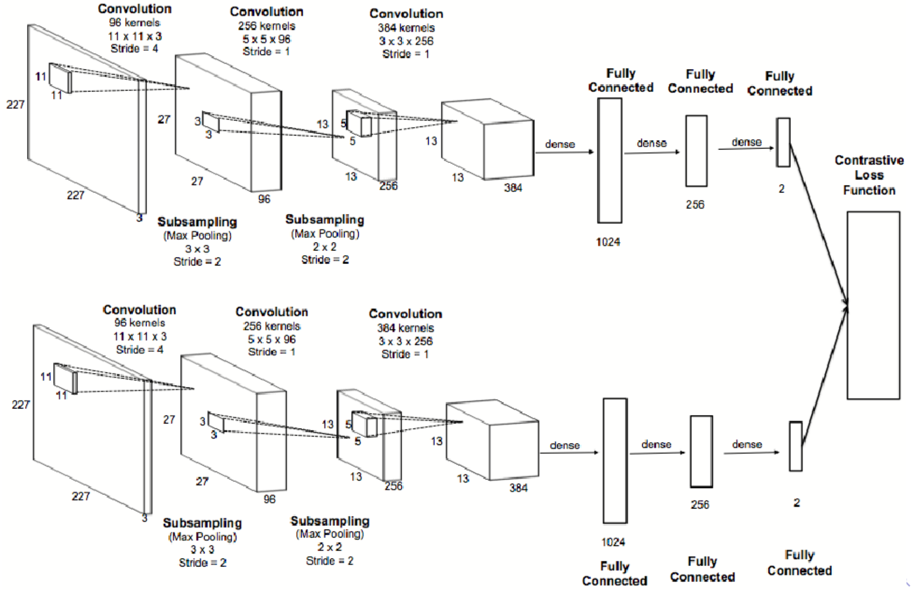
Facial Recognition System

- Suppose there is an organisation and it wants a facial recognition system to allow access to the building for its employees and you are given the task of building such a system.
- The problem with this task is that the organisation might not have more than ten images for each of the employee.
- Therefore, building and training a typical convolutional neural network will not work as it cannot learn the features required with the given amount of data.

Facial Recognition System

- Suppose there is an organisation and it wants a facial recognition system to allow access to the building for its employees and you are given the task of building such a system.
- The problem with this task is that the organisation might not have more than ten images for each of the employee.
- Therefore, building and training a typical convolutional neural network will not work as it cannot learn the features required with the given amount of data.
- What to do?

Siamese NN


Convolution
96 kernels
11 x 11 x 3
Stride = 4
Convolution
256 kernels
5 x 5 x 96
Stride = 1
Convolution
384 kernels
3 x 3 x 256
Stride = 1
227
11
11
227
3
27
3
3
27
96
13
5
5
13
256
13
13
384
Subsampling
(Max Pooling)
3 x 3
Stride = 2
Subsampling
(Max Pooling)
2 x 2
Stride = 2
dense
dense
dense
Fully Connected
Fully Connected
Fully Connected
1024
256
2
Contrastive Loss Function
Convolution
96 kernels
11 x 11 x 3
Stride = 4
Convolution
256 kernels
5 x 5 x 96
Stride = 1
Convolution
384 kernels
3 x 3 x 256
Stride = 1
227
11
11
227
3
27
3
3
27
96
13
5
5
13
256
13
13
384
Subsampling
(Max Pooling)
3 x 3
Stride = 2
Subsampling
(Max Pooling)
2 x 2
Stride = 2
dense
dense
dense
1024
256
2
Fully Connected
Fully Connected
Fully Connected

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),
- a represents a sample of the dataset,

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),
- a represents a sample of the dataset,
- p represents a random positive sample,

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),
- a represents a sample of the dataset,
- p represents a random positive sample,
- n represents a negative sample,

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),
- a represents a sample of the dataset,
- p represents a random positive sample,
- n represents a negative sample,
- m is an arbitrary margin and is used to further the separation between the positive and negative scores,

Triplet Loss Function

$$L = \max\left(d\left(a, p\right) - d\left(a, n\right) + m, 0\right),$$

where

- d is a distance function (e.g. the L_2 loss),
- a represents a sample of the dataset,
- p represents a random positive sample,
- n represents a negative sample,
- m is an arbitrary margin and is used to further the separation between the positive and negative scores,
- if $m = 0.2$ and $d\left(a, p\right) = 0.5$ then $d\left(a, n\right)$ should at least be equal to 0.7.