Fuzzy Set Theory in Computer Vision: **Example 6**

Derek T. Anderson and James M. Keller

FUZZ-IEEE, July 2017

Batch normalization: authors words

- Paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Batch normalization: authors words

- Paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change

Batch normalization: authors words

- Paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change
- This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities - which is referred to as **internal covariate shift**

Batch normalization: authors words

- Paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change
- This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities - which is referred to as **internal covariate shift**
- Ioffe and Szegedy address this by **normalizing layer inputs**

Batch normalization: authors words

- Paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change
- This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities - which is referred to as **internal covariate shift**
- Ioffe and Szegedy address this by **normalizing layer inputs**
- This method draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch

Batch normalization: authors words

- Batch Normalization allows one to use higher learning rates and be less careful about initialization (the claim)

Batch normalization: authors words

- Batch Normalization allows one to use higher learning rates and be less careful about initialization (the claim)
- It also acts as a regularizer, in some cases eliminating the need for dropout ... (the claim)

Batch normalization: authors words

- Batch Normalization allows one to use higher learning rates and be less careful about initialization (the claim)
- It also acts as a regularizer, in some cases eliminating the need for dropout ... (the claim)
- It was applied to a state-of-the-art image classification model and Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin

Batch normalization: authors words

- Batch Normalization allows one to use higher learning rates and be less careful about initialization (the claim)
- It also acts as a regularizer, in some cases eliminating the need for dropout ... (the claim)
- It was applied to a state-of-the-art image classification model and Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin
- Using an ensemble of batch-normalized networks, improvement on ImageNet classification: reaching 4.9 percent top-5 validation error (and 4.8 percent test error), exceeding accuracy of human raters

Helpful links

- http://cs231n.stanford.edu/slides/2016/winter1516_lecture5.pdf
- http://cs231n.stanford.edu/syllabus.html
- https://gab41.lab41.org/batch-normalization-what-the-hey-d480039a9e3b
- https://gist.github.com/shagunsodhani/4441216a298df0fe6ab0
- https://medium.com/towards-data-science/batch-normalization-8a2e585775c9
- https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html
- https://r2rt.com/implementing-batch-normalization-in-tensorflow.html

Algorithmic description

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N^{\mathrm{inf}}_{\mathrm{BN}}$

1: $N^{\mathrm{tr}}_{\mathrm{BN}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N^{\mathrm{tr}}_{\mathrm{BN}}$ (Alg. 1)
4: Modify each layer in $N^{\mathrm{tr}}_{\mathrm{BN}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N^{\mathrm{tr}}_{\mathrm{BN}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N^{\mathrm{inf}}_{\mathrm{BN}} \leftarrow N^{\mathrm{tr}}_{\mathrm{BN}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu^{(k)}_{\mathcal{B}}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma^2_{\mathcal{B}}]$$

11: In $N^{\mathrm{inf}}_{\mathrm{BN}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

Synopsis

- More-or-less, the idea is instead of normalizing the inputs to the network we normalize the inputs to layers in the network

Synopsis

- More-or-less, the idea is instead of normalizing the inputs to the network we normalize the inputs to layers in the network
- Its called "batch" normalization because during training, we normalize each layers inputs by using the mean and variance of the values in the current mini-batch

Synopsis

- More-or-less, the idea is instead of normalizing the inputs to the network we normalize the inputs to layers in the network
- Its called "batch" normalization because during training, we normalize each layers inputs by using the mean and variance of the values in the current mini-batch
- Claimed benefits; network training faster, allows higher learning rates, makes weights easier to initialize, makes more activation functions viable, simplifies the creation of deeper networks, provides a bit of regularlization, MAY give better results overall

Synopsis

- BN2015 paper proposes BN of the input to the activation function of each neuron (e.g., each sigmoid or ReLU function) during training, so that the input to the activation function across each training batch has a mean of 0 and a variance of 1

Synopsis

- BN2015 paper proposes BN of the input to the activation function of each neuron (e.g., each sigmoid or ReLU function) during training, so that the input to the activation function across each training batch has a mean of 0 and a variance of 1
- Is this really where we want to be normalizing?

Synopsis

- BN2015 paper proposes BN of the input to the activation function of each neuron (e.g., each sigmoid or ReLU function) during training, so that the input to the activation function across each training batch has a mean of 0 and a variance of 1
- Is this really where we want to be normalizing?
- $BN(x_i) = \gamma \left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \beta$ (where we learn γ and β)

Synopsis

- BN2015 paper proposes BN of the input to the activation function of each neuron (e.g., each sigmoid or ReLU function) during training, so that the input to the activation function across each training batch has a mean of 0 and a variance of 1
- Is this really where we want to be normalizing?
- $BN(x_i) = \gamma \left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \beta$ (where we learn γ and β)
- Note, the γ and β allows the system to learn a zero mean and unit variance or to undo batch normalization all together

Synopsis

- At test time, BN layer behaves differently

Synopsis

- At test time, BN layer behaves differently
- The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used (e.g. can be estimated during training with running averages)

Synopsis

- At test time, BN layer behaves differently
- The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used (e.g. can be estimated during training with running averages)
- See Section 3.1 of the BN2015 paper. Testing the model above only worked because the entire test set was predicted at once, so the "batch mean" and "batch variance" of the test set provided good estimates for the population mean and population variance