Deep Learning: Assignment Two

Aditi Nair (asn264) and Akash Shah (ass502)

April 4, 2017

1 Batch Normalization

1. Let $x_1, ...x_n$ be scalar features. Then we define the mean μ_n as:

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and the variance σ_n^2 as:

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_n)^2$$

Now to normalize the feature x_i (for $1 \leq i \leq n$), we compute:

$$\hat{x_i} = \frac{x_i - \mu_n}{\sigma_n}$$

Then for all $\hat{x_i}$, the expected value is 0:

$$\sum_{i=1}^{n}\hat{x_i} = \sum_{i=1}^{n}\frac{x_i - \mu_n}{\sigma_n} = \frac{1}{\sigma_n}\sum_{i=1}^{n}(x_i - \mu_n) = \frac{1}{\sigma_n}\left[\left(\sum_{i=1}^{n} x_i\right) - n \cdot \mu_n\right] = \frac{1}{\sigma_n}\left[\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i\right] = 0$$

Since $\sum_{i=1}^{n}\hat{x_i} = 0$, the expected value $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}\hat{x_i}$ is also 0.

Then for all $\hat{x_i}$ the variance is 1 since:

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{x_i} - \hat{\mu})^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{x_i} - 0)^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{x_i}^2 = \frac{1}{n}\sum_{i=1}^{n}\Big(\frac{x_i - \mu_n}{\sigma_n}\Big)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - \mu_n)^2}{\sigma_n^2} = \frac{1}{n \cdot \sigma_n^2}\sum_{i=1}^{n}(x_i - \mu_n)^2$$

$$= \frac{n}{n \cdot \sum_{i=1}^{n}(x_i - \mu_n)^2}\sum_{i=1}^{n}(x_i - \mu_n)^2 = 1$$

2. For scalar features $x_1, ...x_n$ the output of the BN module can be written as:

$$y_i = BN_{\gamma,\beta}(x_i) = \gamma\hat{x_i} + \beta$$

with

$$\hat{x_i} = \frac{x_i - \mu_n}{\sqrt{\sigma_n^2 + \epsilon}}$$

μ_n and σ_n are defined as above. For numerical stability, the BN algorithm adds ϵ to σ_n^2 in the denominator before taking the square root.

$$GRADIENTFORMULAS$$

2 Convolution

1. Assuming a stride of 1 and no zero padding, we will get a 3×3 matrix of 9 values.
2. SHOW WORK?

   Assuming an additive filter, the result of this convolution is:

$$\begin{bmatrix} 158 & 183 & 172 \\ 229 & 237 & 238 \\ 195 & 232 & 244 \end{bmatrix}$$

3.

3 Variants of Pooling

1. Three types of pooling are max-pooling, average-pooling and L-P pooling. Max-pooling is implemented in different dimensions by the MaxPool1d, MaxPool2d and MaxPool3d classes in PyTorch. Average-pooling is implemented in different dimensions by the AvgPool1d, AvgPool2d, and AvgPool3d classes in PyTorch. L-P pooling is implemented in by the LPPool2d module in PyTorch.
2. Does this require the output of a full max-pooling operation on an image/matrix, or just one snapshot?
3. Max-pooling essentially applies a maximum function over (sometimes non-overlapping) regions of input matrices. For inputs with several layers of depth, max-pooling is generally applied separately for each layer's matrix. The pooling operation is used to reduce the size of the input matrices in each layer. Applying max-pooling will select only the maximum value in each region of the input matrix, discarding smaller values in each region. This can prevent overfitting by discarding information from the smaller values. In comparison, operations like average-pooling weigh all values in the region equally - and are therefore more conservative in discarding additional information. An average pooling operation in a region with many small values and one large value will have a relatively small output compared to the max pooling operation over the same region, which ignores the overall tendency of values in the region. Therefore the aggressive approach of the max pooling operation can be used as a regularizer.

4 t-SNE

1. The crowding problem refers to the tendency of dimensionality reduction techniques to crowd points of varying similarity close together.

   Maarten and Hinton (2008) provide the following examples. Given a set of points which lie on a two-dimensional manifold in high-dimensional space, it is fairly straightforward to effectively describe their pairwise distances with a two-dimensional map. On the other hand, suppose these points lie on a higher-dimensional manifold - then it becomes much more difficult to model pairwise distances in two dimensions. Maarten and Hinton provide an example on a 10-dimensional manifold, where it is possible to have ten points which are mutually equidistant - this is clearly impossible to map into a two-dimensional space.

   More generally, the distribution of possible pairwise distances in the high-dimensional space is very different than the distribution of possible pairwise distances in the two-dimensional space. Consider a set of points in high-dimensional space which are uniformly distributed around a central point. As the distance of the points from the central point grows, we see that in two dimensions there is less area to accommodate points which are moderately far from the center, than there is to accommodate points which are near to the center. Therefore, if we attempt to model small distances accurately in the two-dimensional space, we will be forced to place moderately large distances much further from the center point. Now, when trying to optimize the two-dimensional mapping, these too-far points will be pushed inward by the SNE objective function, effectively crowding all of the points together.

How does t-SNE alleviate this? (type up later because boring...)

2. Let

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \ \frac{p_{ij}}{q_{ij}} = \sum_i \sum_j p_{ij} log \ p_{ij} - p_{ij} log \ q_{ij}$$

We want to compute $\frac{\delta C}{\delta y_i}$. Note that p_{ij} is constant with respect to y_i but q_{ij} is not. Specifically:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Following the derivation of van der Maaten and Hinton, we define two intermediate values:

$$d_{ij} = ||y_i - y_j||$$

and

$$Z = \sum_{k \neq l}(1 + d_{kl}^2)^{-1}$$

Now we can rewrite q_{ij} as:

$$q_{ij} = \frac{(1 + d_{ij}^2)^{-1}}{\sum_{k \neq l}(1 + d_{kl}^2)^{-1}} = \frac{(1 + d_{ij}^2)^{-1}}{Z}$$

In the new notation, all terms are constant with respect to y_i except d_{ij} and d_{ji} for all j. Now, we can compute the partial derivative $\frac{\delta C}{\delta y_i}$ in terms of the partial derivatives $\frac{\delta C}{\delta d_{ij}}$ and $\frac{\delta C}{\delta d_{ji}}$ for all d_{ij} and d_{ji}. In particular, we notice that the terms d_{ji} and d_{ij} appears once each for all possible values of j. That is:

$$\frac{\delta C}{\delta y_i} = \sum_j \frac{\delta C}{\delta d_{ij}} \frac{\delta d_{ij}}{\delta y_i} + \sum_j \frac{\delta C}{\delta d_{ji}} \frac{\delta d_{ji}}{\delta y_i}$$

First, we compute $\frac{\delta d_{ij}}{\delta y_i}$. For $x, y \in \mathbb{R}^n$, notice that:

$$||x - y|| = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2}$$

Then it follows that:

$$\begin{aligned}
\frac{\delta\Big(||x - y||\Big)}{\delta x} &= \Big\langle \frac{\delta\Big(||x - y||\Big)}{\delta x_1}, ..., \frac{\delta\Big(||x - y||\Big)}{\delta x_n} \Big\rangle^T \\
&= \Big\langle \frac{2(x_1 - y_1)}{2||x - y||}, ..., \frac{2(x_n - y_n)}{2||x - y||} \Big\rangle^T \\
&= \frac{1}{||x - y||} \Big\langle x_1 - y_1, ..., x_n - y_n \Big\rangle^T = \frac{x - y}{||x - y||}
\end{aligned}$$

Since $d_{ij} = ||y_i - y_j||$, it follows that:

$$\frac{\delta d_{ij}}{\delta y_i} = \frac{\delta\Big(||y_i - y_j||\Big)}{\delta y_i} = \frac{y_i - y_j}{||y_i - y_j||}$$

Moreover since $d_{ji} = ||y_j - y_i|| = ||y_i - y_j|| = d_{ij}$, we can say that:

$$\frac{\delta d_{ji}}{\delta y_i} = \frac{y_i - y_j}{||y_i - y_j||}$$

Finally:

$$\frac{\delta C}{\delta y_i} = \sum_j \frac{\delta C}{\delta d_{ij}} \frac{y_i - y_j}{||y_i - y_j||} + \sum_j \frac{\delta C}{\delta d_{ji}} \frac{y_i - y_j}{||y_i - y_j||}$$

$$= \sum_j \left(\frac{\delta C}{\delta d_{ij}} + \frac{\delta C}{\delta d_{ji}} \right) \frac{y_i - y_j}{||y_i - y_j||} == 2 \sum_j \frac{\delta C}{\delta d_{ij}} \frac{y_i - y_j}{||y_i - y_j||}$$

5 Sentence Classification

5.1 ConvNet

5.2 RNN

5.3 Extra credit experiments of fastText

5.4 Extra credit question

6 Language Modeling