Assignment Two

Matthew C. Scicluna
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal
Montréal, QC H3T 1J4
matthew.scicluna@umontreal.ca

March 7, 2018

1 Convolutions

We compute the full valid and same convolution with kernel flipping for the following matrices: $[1, 2, 3, 4] * [1, 0, 2]$

- The valid convolution is: $[1 \cdot 2 + 2 \cdot 0 + 3 \cdot 1,\ 2 \cdot 2 + 3 \cdot 0 + 4 \cdot 1] = [5, 8]$
- Likewise the same convolution is: $[0, 1, 2, 3, 4, 0] * [1, 0, 2] = [2, 5, 8, 6]$
- Finally the full convolution is: $[0, 0, 1, 2, 3, 4, 0, 0] * [1, 0, 2] = [1, 2, 5, 8, 6, 8]$

2 Convolutional Neural Networks

Consider a 3-layer CNN. We are given an input of size $3 \times 256 \times 256$. The first layer contains 64 8×8 kernels using a stride of 2 and no padding. The shape of its output is $64 \times 125 \times 125$ using relationship 6 from [1]:

$$\text{output length} = \left\lfloor \frac{256 + 2 \cdot 0 - 8}{2} \right\rfloor + 1 = 125$$

The second layer subsamples this using 5×5 non-overlapping max pooling. It is easy to see that the size of its output is $64 \times 25 \times 25$, since $\frac{125}{5} = 25$. The final layer convolves 128 4×4 kernels with a stride of 1 and a zero-padding of size 1 on each border. Using the formula we have that $\left\lfloor \frac{25+2 \cdot 1-4}{1} \right\rfloor + 1 = 24$, and so the output of the last layer has shape $128 \times 24 \times 24$.

(a) The output of the last layer will be of size: $128 \times 24 \times 24 = 73725$

(b) Ignoring biases, we would need $64 \times 25 \times 25 \times 128 = 5120000$ weights

3 Kernel configuration for CNNs

We are given an input shape of $3 \times 64 \times 64$ and the output shape is $64 \times 32 \times 32$ for a convolutional layer.

(a) Assuming no dilation and kernel size of 8×8, we can solve for the stride length s and the padding p by solving the relationship with the given kernel size (setting $s = 2$ for simplicity):

$$\begin{aligned} \left\lfloor \frac{64 + 2 \cdot p - 8}{2} \right\rfloor + 1 &= 32 \\ 32 + p - 4 + 1 &= 32 \\ p &= 3 \end{aligned}$$

Setting 3 padding with 2 stride satisfies the convolution dimensions. Assuming dilatation $d = 6$ and stride of $s = 2$, we can use relationship 15 from [1] to get:

$$\begin{aligned} \left\lfloor \frac{64 + 2 \cdot p - k - (k-1)(6-1)}{2} \right\rfloor + 1 &= 32 \\ \left\lfloor \frac{69 + 2 \cdot p - 6 \cdot k}{2} \right\rfloor &= 31 \end{aligned}$$

This is satisfied when $69 + 2 \cdot p - 6 \cdot k = 63$. We simplify further to get $2p - 6k + 6 = 0$, for which one possible solution is: $p = 3$, $k = 2$. Therefore, setting padding to be 3 and kernel size 2×2 satisfies the convolution dimensions.

(b) Given an input shape of $64 \times 32 \times 32$ and the output shape is $64 \times 8 \times 8$ a configuration assuming no overlapping of pooling windows or padding would have kernel size 4 and stride 1. This is easily seen since $\frac{32}{8} = 4$.

(c) Without any padding and given input shape $64 \times 32 \times 32$ and kernel of size 8×8 and stride 4 we can use the relation to get:

$$\text{output length} = \left\lfloor \frac{32 + 2 \cdot 0 - 8}{4} \right\rfloor + 1 = 7$$

And so the output size would be 7×7.

(d) We are given input shape $64 \times 8 \times 8$ and output $128 \times 4 \times 4$

   (i) Assuming no padding and no dilation and using the relation above, we can easily solve to get kernel size 4 and stride 2.

(ii) Assuming dilatation of 1 and padding of 2, kernel size 6 and stride 2 satisfies the input/output dimensions.

(iii) Assuming padding of 1 and no dilatation, kernel size 4 and stride 2 satisfies the input/output dimensions.

4 Dropout as weight decay

We consider a linear regression problem with input data $X \in \mathbb{R}^{n\times d}$, weights $w \in R^{d\times 1}$ and and targets $y \in R^{n\times 1}$. We also suppose that dropout is being applied to the input units with probability p.

(a) We can let $\tilde{X} = P \odot X$ where $P_{ij} \sim Bernoulli(p)$

(b) The cost function of this would be

$$\begin{aligned}
\mathbb{E}_P\left\{\|y-\tilde{X}w\|^2\right\} &= \mathbb{E}_P\left\{(y-\tilde{X}w)^T(y-\tilde{X}w)\right\} \\
&= y^Ty - 2w^T\mathbb{E}_P\left\{\tilde{X}^T\right\}y + \mathbb{E}_P\left\{w^T\tilde{X}^T\tilde{X}w\right\} \\
&= y^Ty - 2pw^TX^Ty + \left(p^2(Xw)^T(Xw) - p^2(Xw)^T(Xw)\right) + \mathbb{E}_P\left\{w^T\tilde{X}^T\tilde{X}w\right\} \\
&= \|y-pXw\|^2 + \mathbb{E}_P\left\{w^T\tilde{X}^T\tilde{X}w\right\} - p^2(Xw)^T(Xw) \\
&= \|y-pXw\|^2 + \mathbb{E}_P\left\{w^T\tilde{X}^T\tilde{X}w\right\} - \mathbb{E}_P\left\{(\tilde{X}w)^T(\tilde{X}w)\right\} \\
&= \|y-pXw\|^2 + w^T\left(\mathbb{E}_P\left\{\tilde{X}^T\tilde{X}\right\} - \mathbb{E}_P\left\{\tilde{X}\right\}^T\mathbb{E}_p\left\{\tilde{X}\right\}\right)w
\end{aligned}$$

We evaluate the matrix in the rightmost term:

$$\begin{aligned}
\mathbb{E}_P\left\{\tilde{X}^T\tilde{X}\right\} - \mathbb{E}_P\left\{\tilde{X}\right\}^T\mathbb{E}_p\left\{\tilde{X}\right\} &= \left[\sum_k \mathbb{E}\left\{p_{ki}p_{kj}\right\}x_{ki}x_{kj} - \mathbb{E}\left\{p_{ki}\right\}\mathbb{E}\left\{p_{kj}\right\}x_{ki}x_{kj}\right]_{ij} \\
&= \sum_k Cov(p_{ki},p_{kj})x_{ki}x_{kj} \\
&= \begin{cases} \sum_k p(1-p)x_{ki}^2 & \text{if } i=j \\ 0 & \text{o.w.} \end{cases} \\
&= Diag(X^TX)p(1-p)
\end{aligned}$$

Inserting this into the cost function gives us:

$$\begin{aligned}
\mathbb{E}_P\left\{\|y-\tilde{X}w\|^2\right\} &= \|y-pXw\|^2 + p(1-p)w^TDiag(X^TX)w \\
&= \|y-pXw\|^2 + p(1-p)\|\Gamma w\|^2
\end{aligned}$$

Where $\Gamma = Diag(X^TX)^{\frac{1}{2}}$

(c) We show that applying dropout to the linear regression problem can be seen as using L2 regularization in the loss function. Let $\tilde{w} = pw$. The optimal value of the cost function is:

$$\frac{\partial}{\partial \tilde{w}} \left(\|y - X\tilde{w}\|^2 + \frac{1-p}{p}\|\Gamma\tilde{w}\|^2 \right) = -2X^T y + 2X^T X\tilde{w} + 2\frac{1-p}{p}\Gamma^2\tilde{w}$$

And setting this to zero yields:

$$\begin{aligned} &\frac{1-p}{p}\Gamma^2\tilde{w} + X^T X\tilde{w} = X^T y \\ &\Rightarrow \left(\frac{1-p}{p}\Gamma^2 + X^T X\right)\tilde{w} = X^T y \\ &\Rightarrow \tilde{w} = \left(\lambda\Gamma^2 + X^T X\right)^{-1} X^T y \end{aligned}$$

where $\lambda = \frac{1-p}{p}$. Notice that the solution to the regularized least squares problem is identical except the $\lambda\Gamma^2$ term is replaced by λI. In dropout, the Γ term adds additional cost to weights which are in directions where the data varies, wheras in ordinary L2 regularized least squares, the directions are penalized by the same amount.

5 Dropout as Geometric Ensemble

We show that weight scaling with a factor of 0.5 corresponds exactly to the inference of a conditional probability distribution proportional to the geometric mean over all dropout masks:

$$p_{\text{ens}}(y = j|v) \propto \left(\prod_{i=1}^{N} \hat{y}_j^{(i)}\right)^{\frac{1}{N}}$$

Where N is the number of dropout masks, $\hat{y}_j^{(i)} = softmax\left(W^T(m_i \odot v) + b\right)_j$ and m_i is a dropout mask configuration, for which there are N of. We expand the geometric mean:

$$\begin{aligned}
p_{\text{ens}}(y = j|v) &\propto \left(\prod_{i=1}^{N} softmax\left(W^T(m_i \odot v) + b\right)_j\right)^{\frac{1}{N}} \\
&= \left(\prod_{i=1}^{N} \frac{\exp\left\{W^T(m_i \odot v) + b\right\}_j}{\sum_{j'} \exp\left\{W^T(m_i \odot v) + b\right\}_{j'}}\right)^{\frac{1}{N}} \\
&= \frac{\left(\prod_{i=1}^{N} \exp\left\{W^T(m_i \odot v) + b\right\}_j\right)^{\frac{1}{N}}}{\left(\prod_{i=1}^{N} \sum_{j'} \exp\left\{W^T(m_i \odot v) + b\right\}_{j'}\right)^{\frac{1}{N}}} \\
&\propto \left(\prod_{i=1}^{N} \exp\left\{W^T(m_i \odot v) + b\right\}_j\right)^{\frac{1}{N}} \\
&= \exp\left\{\frac{1}{N}\sum_{i=1}^{N} W^T(m_i \odot v) + b\right\}_j \\
&= \exp\left\{\frac{1}{2}W^T v + b\right\}_j
\end{aligned}$$

6 Normalization

We investigate Weight Normalization (WN). We decouple the weight vector into two terms :

$$w = \frac{g}{\|u\|}u$$

where $g \in \mathbb{R}$ is a scaling factor. Doing so has similar effects as implementing Batch Normalization (BN), but has a lower computational overhead.

(a) We consider the simplest model, where we only have one single output layer conditioned on one input feature x. Additionally, we assume $\mathbb{E}\{x\} = 0, Var\{x\} = 1$. We show that in this simple case WN is equivalent to BN (ignoring the learned scale and shift terms) that normalizes the linearly transformed feature $a = w^T x + b$. From 8.35 of [2] we have that:

$$BN(a) = \frac{a - \mathbb{E}\{a\}}{\sqrt{Var\{a\}}} = \frac{w^T x}{\|w\|}$$

since $\mathbb{E}\left\{w^T x\right\} = w^T \mathbb{E}\{x\} = 0$ and $Var\left\{w^T x\right\} = w^T Var\{x\}\, w = w^T w = \|w\|^2$. Ignoring the scale and shift terms, we see that this is equivalent to weight normalization.

(b) Show that the gradient of a loss function L with respect to the new parameters u can be expressed in the form $sW^*\nabla_w L$, where s is a scalar and W^* is the orthogonal complement projection matrix. We compute the gradient. Using the multivariate chain rule:

$$\begin{aligned}
\nabla_u L &= \nabla_u w \nabla_w L \\
&= g\nabla_u \frac{u}{\|u\|}\nabla_w L \\
&= \frac{g}{\|u\|}\left(I - \frac{uu^T}{\|u\|^2}\right)\nabla_w L
\end{aligned}$$

Since

$$\frac{\partial w_i}{\partial u_j} = \frac{\partial}{\partial u_j}\frac{u_i}{\|u\|} = \frac{1_{i=j} - \frac{u_j}{\|u\|^2}u_i}{\|u\|}$$

And so multiplying and dividing the matrix by $g^2 = \|w\|^2$ gives us

$$\begin{aligned}
\nabla_u L &= \frac{g}{\|u\|}\left(I - \frac{ww^T}{\|w\|^2}\right)\nabla_w L \\
&= sW^*\nabla_w L
\end{aligned}$$

Where $s = \frac{g}{\|w\|}$ and $W^* = \left(I - \frac{ww^T}{\|w\|^2}\right)$ is a projection matrix that projects onto the complement of w.

(c) The effect in the figure is a consequence of (b). Let $u' = u + \alpha\nabla_u L$, which is standard gradient descent with α learning rate. Since W^* projects $\nabla_u L$ orthogonal to w, we have that $w \perp \nabla_u L$ and so $u \perp \nabla_u L$ (since $u \propto w$). Let $c = \frac{\|\nabla_u L\|}{\|u\|}$. We can use Pythagorean theorem (due to the orthogonality of u and $\nabla_u L$) to get that

$$\begin{aligned}
\|u'\| = \|u + \alpha\nabla_u L\| &= \sqrt{\|u\|^2 + \alpha^2\|\nabla_u L\|^2} \\
&= \sqrt{\|u\|^2 + \alpha^2 c^2\|u\|^2} \\
&= \sqrt{1 + \alpha^2 c^2}\|u\| \\
&\geq \|u\|
\end{aligned}$$

We see that $\|u\|$ grows monotonically, and this growth is proportional to α. This explains what is happening in the graph.

References

[1] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning.," *CoRR*, vol. abs/1603.07285, 2016.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.