ASSIGNMENT 2: THEORY OF CNNS AND REGULARIZATION [IFT6135]

JOSEPH D. VIVIANO

1. Convolutions

$\boldsymbol{x} = (..., x_{-1}, x_0, x_1, x_2, x_3, x_4, ...) = (..., 0, 1, 2, 3, 4, 0, ...)$
$\boldsymbol{w} = (..., w_{-1}, w_0, w_1, w_2, w_3, w_4, ...) = (..., 0, 1, 0, 2, 0, 0, ...)$

$$[\boldsymbol{x} * \boldsymbol{w}](i) = \sum_{k=-\infty}^{\infty} x_k w_{i-k} \tag{1.1}$$

$[\boldsymbol{x} * \boldsymbol{w}](0) = x_0 w_0 = 1 \cdot 1 = 1$

$[\boldsymbol{x} * \boldsymbol{w}](1) = x_0 w_1 + x_1 w_0 = 1 \cdot 0 + 2 \cdot 1 = 2$

$[\boldsymbol{x} * \boldsymbol{w}](2) = x_0 w_2 + x_1 w_1 + x_2 w_0 = 1 \cdot 2 + 2 \cdot 0 + 3 \cdot 1 = 5$

$[\boldsymbol{x} * \boldsymbol{w}](3) = x_0 w_3 + x_1 w_2 + x_2 w_1 + x_3 w_0 = 1 \cdot 0 + 2 \cdot 2 + 3 \cdot 0 + 4 \cdot 1 = 8$

$[\boldsymbol{x} * \boldsymbol{w}](4) = x_2 w_2 + x_3 w_1 = 3 \cdot 2 + 4 \cdot 0 = 6$

$[\boldsymbol{x} * \boldsymbol{w}](5) = x_3 w_2 = 4 \cdot 2 = 8$

Let's $y_i = [\boldsymbol{x} * \boldsymbol{w}](i)$. Thus we have $\boldsymbol{y} = (1, 2, 5, 8, 6, 8)$ where the null elements are not shown.

2. Convolutional Neural Networks

(a) NB: zero padding in layer three, so size of layer 3 is $128 \times 6 \times 6$. The image is RGB (i.e., 3 channels), therefore the last layer is a fully connected layer of the size

$$3 \times 128 \times 6 \times 6 = 13824$$

.

(b) The last convolution has a kernel size 4×4 and there are 128 filters with 3 channels, so

$$n\_params = 4 \times 4 \times 128 \times 3 = 6144$$

.

Date: March 2018.

3. Kernel Configurations for CNNs

(a): $\boldsymbol{i}$: input is $W_1 \times H_1$ and output is $W_2 \times H_2$. Kernel size is K, zero padding is P and stride is S. Therefore

$$W_2 = \frac{W_1 - K + 2P}{S} + 1, \tag{3.1}$$

plugging our numbers in, we get

$$32 = \frac{64 - 8 + 2P}{S} + 1$$

. Either $\boldsymbol{P = 3}$ and $\boldsymbol{S = 2}$ would produce a proper convolution.

$\boldsymbol{ii}$: Dilatation size is D,

$$W_2 = \frac{W_1 - K + 2P + (W_1 - 1)D}{S} + 1 \tag{3.2}$$

So plugging in,

$$32 = \frac{64 - K + 2P + 63.6}{2} + 1.$$

If we set $\boldsymbol{K = 400}$ and $\boldsymbol{P = 10}$, then our convolution operation works.

(b): If the kernel size of the pooling layer is $\boldsymbol{K = 4 \times 4}$ with no overlap, and the stride size is $\boldsymbol{S = 4}$, the pooling operation works.

(c): $K = 8$, $W_1 = 32$ and $S = 4$, we plug them in and presto

$$W_2 = \frac{32 - K}{4} + 1 = 7.$$

The output is $\mathbf{7 \times 7}$.

(d): $\boldsymbol{i}$ $W_2 = 4$, $W_1 = 8$ and $P = 0$, plugging in, we get

$$4 = \frac{8 - K + 0}{S} + 1.$$

Therefore $\boldsymbol{K = 2}$ and $\boldsymbol{S = 2}$ are appropriate.

$\boldsymbol{ii}$: $W_2 = 4$, $W_1 = 8$, $P = 2$ and $D = 1$. Plugging in, we get

$$4 = \frac{8 - K + 4 + 7}{S} + 1.$$

So $\boldsymbol{K = 13}$ and $\boldsymbol{S = 2}$ are appropriate.

$\boldsymbol{iii}$:

$$4 = \frac{8 - K + 2}{S} + 1,$$

so $\boldsymbol{K = 4}$ and $\boldsymbol{S = 2}$ are appropriate.

4. Dropout as Weight Decay

(a) Let $\tilde{X} = X \odot \boldsymbol{\delta}$ where $\odot$ is the element wise product of matrices and $\boldsymbol{\delta} = (\delta_1, ..., \delta_n)$, with

$$\delta_i = \begin{cases} \mathbf{0}, & p \\ \mathbf{1}, & 1-p, \end{cases} \tag{4.1}$$

and $\mathbf{0}$ and $\mathbf{1}$ are1 $\times\, d$ vectors of elements 0 and 1.

(b) Let L_{MSE} be the loss function. The general formula for the loss function is as follows

$$L_{MSE}(w) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2 \tag{4.2}$$

where $\hat{y}^{(i)}$ is the the dropout prediction when we add dropout so using part (a) we have

$$L_{MSE}(w) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)}\delta_i w - y^{(i)})^2 \tag{4.3}$$

(c) Assume dropout probability is $1-p$ for eacn input unit. Let $J_D(w, \delta)$ be the network's loss function with dropout, so we minimize

$$E\left[J_D(w,\delta)\right] = E_{\delta \sim Bernoulli(p)}\left[\frac{1}{2}||(\delta \odot x)w - y||_2^2\right],$$

We define $\delta \odot x := \tilde{x}$, so

$$\begin{aligned} E\left[J_D(w.\delta)\right] &= E_\delta\left[\frac{1}{2}\left(\tilde{x}w - y\right)^T\left(\tilde{x}w - y\right)\right] \\ &= E_\delta\left[\frac{1}{2}w^T\tilde{x}^T\tilde{x}w - w\tilde{x}^T y + \frac{1}{2}yy^T\right] \\ &= \frac{1}{2}w^T E_\delta\left[\tilde{x}^T\tilde{x}\right]w - wE_\delta[\tilde{x}^T]y + \frac{1}{2}y^T y \end{aligned} \tag{4.4}$$

Using the covariance of two matrices, we can see that

$$\begin{aligned} E\left[J_D(w.\delta)\right] &= \frac{1}{2}w^T\left(E_\delta[\tilde{x}^T]E_\delta[\tilde{x}] + Cov_\delta\left[(\tilde{x}^T, \tilde{x})\right]\right)w - wE_\delta[\tilde{x}^T]y + \frac{1}{2}y^T y \\ &= \frac{1}{2}w^T p^2 x^T x w + \frac{1}{2}w^T p(1-p)\left(diag(x^T x)^{1/2}\right)^2 w - wpx^T y + \frac{1}{2}y^T y, \end{aligned} \tag{4.5}$$

Where the $Cov_\delta(\tilde{x}, \tilde{x}^T)$ is shown in the following matrix

$$\begin{bmatrix} x_1^2 var(\delta_1) \dots & 0 & \\ \vdots & \ddots & \\ 0 & \dots & x_n^2 var(\delta_n) \end{bmatrix}$$

where $var(\delta_i) = p(1-p)$ for each i. So we get a diagonal of the matrix $(x^T x)$. To find the minimizing point we put the derivative (with respect to w) equal to zero, so we have

$$\left[\frac{\partial E[J_D]}{\partial w}\right] = p^2 x^T x w + p(1-p)\Gamma^2 w - p x^T y = 0 \tag{4.6}$$

Thus

$$pw\left(x^T x + \frac{(1-p)}{p}\Gamma^2\right) - x^T y = 0$$

and this gives us

$$w^* = pw = \left(x^T x + \frac{(1-p)}{p}\Gamma^2\right)^{-1} x^T y$$

The weight that minimize the cost when using $L2$-regularization(deep learning book) is given by

$$w = \left(X^\top X + \alpha I\right)^{-1} X^\top y \tag{4.7}$$

where I is the identity matrix and α is the regularization parameter. We can see that both parameter have very similar form. Hence, dropout and L2 regularization should lead to similar regularization effect. The difference between is the diagonal matrix *Gamma*. With dropout, it is equal to the covariance matrix where with L2 regularization it is identity matrix.

5. Dropout as a Geometric Ensemble

Consider the case of a single linear layer model with a softmax output. Prove that weight scaling by 0.5 corresponds exactly to the inference of a conditional probability distribution proportional to a geometric mean over all dropout masks.

First, observe the single linear layer with softmax output with n input variables represented by the vector v with dropout mask d:

$$P(y = \mathrm{y}|v; d) = \mathbf{softmax}\left(W^T(d \odot v) + b\right)_y \tag{5.1}$$

and the ensemble conditional probability distribution which represents the geometric mean over all dropout masks:

$$p_{ens}(y = \mathrm{y}|v; d) \propto \left(\prod_{i=1}^{N} \hat{y}_v^{(i)}\right)^{\frac{1}{N}}. \tag{5.2}$$

Aren't they nice? Recall the alternative formulation of the softmax:

$$\textbf{softmax}_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \tag{5.3}$$

Which we now rewrite, subbing in our vector representation of the softmax and replacing e^x with $exp(x)$:

$$\textbf{softmax}_y = \frac{exp\left(W_y^T(d \odot v) + b\right)}{\sum_{k=1}^{K} exp\left(W_{y'}^T(d \odot v) + b\right)} \tag{5.4}$$

Now we show that the ensemble predictor is defined by re-normalizing the geometric mean over all the individual ensemble members' predictions:

$$P_{ens}(y = \text{y}|v) = \frac{\tilde{P}_{ens}(y = \text{y}|v)}{\sum y' \tilde{P}_{ens}(y = \text{y'}|v)} \tag{5.5}$$

Where each $\tilde{P}_{ens}$ is the geometric mean over all dropout masks for a single y:

$$\tilde{P}_{ens}(y = \text{y}|v) = 2^n \sqrt{\prod_{d \in \{0,1\}^n} P(y = \text{y}|v; d)}. \tag{5.6}$$

Now we simply sub in our definition of $softmax$ for P:

$$\tilde{P}_{ens}(y = \text{y}|v) = 2^n \sqrt{\prod_{d \in \{0,1\}^n} \frac{exp\left(W_y^T(d \odot v) + b\right)}{\sum_{k=1}^{K} exp\left(W_{y'}^T(d \odot v) + b\right)}}. \tag{5.7}$$

Since the denominator is a constant under this normalization scheme we ignore it and simplify:

$$\tilde{P}_{ens}(y = \text{y}|v) \propto 2^n \sqrt{\prod_{d \in \{0,1\}^n} exp\left(W_y^T(d \odot v) + b\right)} \tag{5.8}$$

We convert the product to the sum by taking exp of the entire equation:

$$\tilde{P}_{ens}(y = \text{y}|v) \propto exp\left(\frac{1}{2^n} \sum_{d \in \{0,1\}^n} W_y^T(d \odot v) + b\right) \tag{5.9}$$

And finally the sum and exponent n cancel:

$$\tilde{P}_{ens}(y = \mathrm{y}|v) \propto exp\left(\frac{1}{2}W_y^T(d \odot v) + b\right) \tag{5.10}$$

Finally, we sub this back into our earlier formulation of the softmax to show that the weights W are scaled by $\frac{1}{2}$:

$$\mathbf{softmax}_y = \frac{exp\left(\frac{1}{2}W_y^T(d \odot v) + b\right)}{\sum_{k=1}^{K} exp\left(\frac{1}{2}W_{y'}^T(d \odot v) + b\right)} \tag{5.11}$$

Therefore, weight scaling by 0.5 is exactly equivilant to a conditional probability distribution proportional to a geometric mean over all dropout masks.

6. Normalization

(a) *Show batchnorm and weightnorm are the same when you only have one layer and input feature x.*

To normalize the minibatch of activations B, we do

$$B' = \frac{B - \delta}{\sigma} \tag{6.1}$$

Where δ is the mean of B, and σ is the standard deviation of B (with a small positive value added for numerical stability).
We can replace B with $w^\top x$, where w is our weight matrix to see:

$$B' = \frac{w^\top x}{\sqrt{\mathrm{Var}[w^\top x]}} - \frac{\mathbb{E}w^\top x}{\sqrt{\mathrm{Var}[w^\top x]}} \tag{6.2}$$

Now notice the following about the unit vector

$$\frac{u}{||u||} = \frac{w^\top}{||w||} \tag{6.3}$$

And

$$g = \frac{||w||_2}{\sqrt{\mathrm{Var}[w^\top x]}} \tag{6.4}$$

Therefore if we assume x and w are independent and that x has 0 mean:

$$B' = \frac{||w||_2}{\sqrt{\mathrm{Var}w^\top x}} \frac{w^\top}{||w||_2} x - \frac{\mathbb{E}w^\top x}{\sqrt{\mathrm{Var}[w^\top x]}} \tag{6.5}$$

$$B' = g\frac{u}{||u||}x - \frac{\mathbb{E}w^\top x}{\sqrt{\mathrm{Var}[w^\top x]}} \tag{6.6}$$

The expectation and standard deviation are constant under these conditions, so we get:

$$B' = g\frac{u}{||u||}x - c \tag{6.7}$$

But we can ignore c for this question.

(b) *Show the gradients of L with respect to u can be expressed as* $sW^*\nabla_w L$.

From above:

$$B' = g\frac{u}{||u||}x \tag{6.8}$$

With weightnorm, we explicity reparameterize the model to perform gradient descent in the new parameters g and u directly. By decoupling the norm of the weight vector g and it's direction $\frac{u}{||u||}$, we can speed up convergence dramatically.

If we differentiate through the above with respect to some new parameters v, we get:

$$\nabla_g L = \frac{\nabla_u L \cdot u}{||u||}, \nabla_v L = \frac{g}{||u||}\nabla_u L - \frac{g\nabla_g L}{||u||^2}u \tag{6.9}$$

Where $\nabla_u L$ is the gradient with respect to the weights.
Let's sub in $\nabla_g L$ into $\nabla_v L$ to get:

$$\nabla_v L = \frac{g}{||u||}\nabla_u L - \frac{g\frac{\nabla_u L \cdot u}{||u||}}{||u||^2}u \tag{6.10}$$

Or,

$$\nabla_v L = \frac{g}{||u||}\nabla_u L - \frac{g\nabla_u L}{||u||^3}u^\top u \tag{6.11}$$

This leads us to the forumulation:

$$\nabla_v L = \frac{g}{||u||}M_u\nabla_u L \tag{6.12}$$

where,

$$M_u = Id - \frac{u^\top u}{||u||^2} \tag{6.13}$$

where where M_u is a projection matrix that projects onto the complement of the u vector, and Id is the identity matrix.

(c) *Explain a graph of different learning rates.*

Let λ be the learning rate. During learning we update u via at step k using $v_k \leftarrow v_k - \lambda \nabla_v L$.

As we said in the previous question, the matrix M_u project onto the complement of u. Therefore, $\nabla_v L$ is equal to a constant times M_u, i.e., $u \perp \lambda \nabla_v L$.

Since our update v is proportional to w, the update must be orthogonal to v and the norm increases by the Pythagorean theorem, which states that for any two orthogonal vectors v and v' the new weight vector must have the norm

$$||v'|| = \sqrt{||v||^2 + c^2||v||^2} \tag{6.14}$$

if

$$c = ||\lambda \nabla_v L||/||v|| \tag{6.15}$$

This tells us a few things. If the norm of the gradients is small, $\sqrt{1+c^2}$ is close to 1 and the norm of v stops increasing. As the norm of the gradients grow, the norm of v will also grow. Also, the norm of the updated parameter is proportional to the absoloute value of the learning rate. These observations explain the graph.

Université de Montréal
Email address: joseph@viviano.ca