CSC411 HW4

Casey Juanxi Li - 998816973

October 2018

1 AlexNet.

a) Counting units, weights, and connections. Referencing:

- `http://ttic.uchicago.edu/~shubhendu/Pages/Files/Lecture7_flat.pdf`
- `http://cs231n.github.io/convolutional-networks/`

I assume that the following statements hold. Referencing TA comments from Piazza where necessary:

For fully connected layers:

- **Units** are explicitly stated.
- **Weights** = **Connections** = (# of neurons in previous layer) ×(# of neurons in current layer)

For Conv layers:

- **Dimension of feature map** = # of different receptive fields that the kernel can "scan" during convolution
- **Units** = neurons = dimension of feature map × kernel count
- **Weights** = # of parameters per kernel, assuming no biases × kernel count (Piazza @585)
- **Connections** = weights ×dimension of feature map. (Piazza @583)
- Ignore pooling and bias.

	# Units	# Weights	# Connections
Convolution Layer 1	290,400	34,848	105,415,200
Convolution Layer 2	186,624	614,400	447,897,600
Convolution Layer 3	64,896	884,736	149,520,384
Convolution Layer 4	64,896	1,327,104	224,280,576
Convolution Layer 5	43,264	884,736	149,520,384
Fully Connected Layer 1	4,096	177,209,344	177,209,344
Fully Connected Layer 2	4,096	16,777,216	16,777,216
Output Layer	1,000	4,096,000	4,096,000

Showing work:
Note that we've had to take some guesses about stride and padding to make the numbers match Figure 2. It's not evident from Fig. 2 that they are using $\frac{K-1}{2}$ as padding. These calculations also don't match the 60 million parameters stated, due to ignoring pooling.

We're also going to pretend that they used one GPU, so some kernel depths here will be double what appears in Fig 2.

Image -> C1:
Side dimension of feature map = $\frac{224+2\times padding-K}{stride} = \frac{224+2\times ?-11}{4} = 55$
Kernel count = 96
Kernel: 11 ×11 ×3

Units = 55 ×55 ×96 = 290,400   (does not match paper, discussed in @587)
Weights = params in kernel × kernel count = 11 × 11 × 3 × 96 = 34,848
Connections = weights × dim. of feature map = 34,848 × 55 × 55 = 105,415,200

C1→C2:
Side dimension of feature map = $\frac{55+2\times?-5}{stride}+1=27$ (unstated stride. 2 seems likely)
Kernel count = 256
Kernel: $5\times5\times96$

Units = $27\times27\times256 = 186{,}624$
Weights = params in kernel × kernel count = $5\times5\times96\times256 = 614{,}400$
Connections = weights × dim. of feature map = $614{,}400\times27\times27 = 447{,}897{,}600$

C2→C3:
Side dimension of feature map = $\frac{27+2\times?-3}{stride}+1=13$ (unstated stride. Again, 2 seems likely)
Kernel count = 384
Kernel: $3\times3\times256$

Units = $13\times13\times384 = 64{,}896$
Weights = params in kernel × kernel count = $3\times3\times256\times384 = 884{,}736$
Connections = weights × dim. of feature map = $884{,}736\times13\times13 = 149{,}520{,}384$

C3→C4:
Side dimension of feature map = $\frac{13+2\times?-3}{stride}+1=13$ (unstated stride. 1 seems likely)
Kernel count = 384
Kernel: $3\times3\times384$

Units = $13\times13\times384 = 64{,}896$
Weights = params in kernel × kernel count = $3\times3\times384\times384 = 1{,}327{,}104$
Connections = weights × dim. of feature map = $1{,}327{,}104\times13\times13 = 224{,}280{,}576$

C4→C5:
Side dimension of feature map = $\frac{13+2\times?-3}{stride}+1=13$ (unstated stride. 1 seems likely)
Kernel count = 256
Kernel: $3\times3\times384$

Units = $13\times13\times256 = 43{,}264$
Weights = params in kernel × kernel count = $3\times3\times384\times256 = 884{,}736$
Connections = weights × dim. of feature map = $884{,}736\times13\times13 = 149{,}520{,}384$

C5→F1:
Units = 4096
For a fully connected layer, every neuron in F1 needs to apply a weight for every neuron in C5. Therefore,
Weights = Connections = (units in C5) × (units in F1) = $43{,}264\times4096 = 177{,}209{,}344$

F1→F2: Units = 4096
Weights = Connections = (units in F1) × (units in F2) = $4096\times4096 = 16{,}777{,}216$

F2→Output: Units = 1000
Weights = Connections = (units in F2) × (units in output) = $4096\times1000 = 4{,}096{,}000$

b) Suggest architecture changes:

i) Run network on cell phone, reduce memory usage at test time, reduce number of parameters:

Use fewer fully connected layers. Parameters are **weights** and **biases**. As pointed out by the paper, and as shown in the table in a), "most of the net's parameters are in the first fully-connected layer". The fully connected layers are particularly costly in terms of adding parameters; I'd see if we can do with removing one of them.

Use more max-pooling. The paper indicates that max-pooling isn't used between layers C3$\rightarrow$C4 and C4$\rightarrow$C5. "Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map." In other words, max pooling allows us to "combine" the signal from many neighbouring neurons into a single neuron, thus reducing the number of weights that are passed into the next layer. If we choose a reasonable max pooling method and pool size, we can hopefully preserve some useful signal while significantly reducing the number of parameters.

ii) Make very rapid predictions at test time. Reduce the number of connections, since there is approximately one add-multiply operation per connection.

Note that connections are a function of **weights**, so the suggestions from i) which reduce parameters would help here as well.

Additionally, the number of connections is a function of:

- The # of weights in the kernel: **use smaller kernels**
- How many kernels we are using: **use fewer kernels**
- The # of receptive fields that kernel "scans" on the input (aka feature map dimensions):
  - **Use a larger stride**: fewer hops to cover the same size image
  - **Use a smaller input image**: downsample train and test data

2 Gaussian Naive Bayes.

a) Use Bayes' rule to derive an expression for $p(y = k|x, \mu, \sigma)$.

$$p(y = k|x, \mu, \sigma) = \frac{p(x|y = k, \mu, \sigma) \cdot p(y = k)}{p(x)}$$

Note that:

- $p(y = k) = \alpha_k$ by definition.
- The generative expression for $p(x|y = k, \mu, \sigma)$ was given in the question.
- Using the law of total probability, the denominator is the sum of the numerator over all possible k.

So the expression is:

$$\boxed{p(y = k|x, \mu, \sigma) = \frac{(\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2} \cdot \alpha_k}{\sum_{k=1}^{K} \big((\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i - \mu_{ki})^2} \cdot \alpha_k\big)}}$$

b) Write down an expression for the negative likelihood function (NLL) of a particular dataset $D = \big\{(y^{(1)}, x^(1))...(y^{(N)}, x^(N))\big\}$ with parameters $\theta = \{\alpha, \mu, \sigma\}$.

$$p(D|\mu, \sigma, \alpha) = \prod_{j=1}^{N} p(x^{(j)}|\mu, \sigma, \alpha) \cdot p(y^{(j)}|\mu, \sigma, \alpha) = \prod_{j=1}^{N} p(x^{(j)}|y^j, \mu, \sigma) \cdot p(y^{(j)}|\alpha)$$

(Note that once y is set, x only depends on μ and σ. y is not at all conditioned on μ or σ.)

$$= \prod_{j=1}^{N} p(x^{(j)}|y^j, \mu, \sigma) \cdot \alpha_y$$

(Note that we calculated this in the numerator of part a, we're now just making it a product over all N)

$$= \prod_{j=1}^{N} \Big((\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2} \cdot \alpha_k \Big)$$

Taking negative log likelihood:

$$NLL = -ln\Bigg(\prod_{j=1}^{N} \Big((\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2} \cdot \alpha_k \Big) \Bigg)$$

$$= \sum_{j=1}^{N} -ln\Bigg(\Big((\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2} \cdot \alpha_k \Big) \Bigg)$$

$$= \sum_{j=1}^{N} \Bigg(-ln\Big((\prod_{i=1}^{D} 2\pi\sigma_i^2)^{-\frac{1}{2}} \Big) - ln\Big(e^{\sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2} \Big) - ln\Big(\alpha_k\Big) \Bigg)$$

$$\boxed{= \sum_{j=1}^{N} \Bigg(\frac{1}{2} ln\Big((\prod_{i=1}^{D} 2\pi\sigma_i^2) \Big) - \sum_{i=1}^{D} \frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2 - ln\Big(\alpha_k\Big) \Bigg)}$$

c) MLE for μ and σ^2:

For any individual μ_{KI}:

$$\frac{\partial NLL}{\partial \mu_{KI}} = \sum_{j=1}^{N}(-1)\sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki}) \cdot 2 \cdot (-1) \qquad \text{(but only where i=I and k=K)}$$
$$= \sum_{j|k=K}\frac{1}{\sigma_I^2}(x_I^{(j)} - \mu_{KI})$$

Set this derivative equal to 0. σ is shared across classes so once we've fixed i=I, we can pull σ out of the sum:

$$0 = \sum_{j|k=K}\frac{1}{\sigma_I^2}(x_I^{(j)} - \mu_{KI}) = \frac{1}{\sigma_I^2}\big(\sum_{j|k=K} x_I^{(j)} - \sum_{j|k=K}\mu_{KI}\big)$$

In order for the derivative to be 0 it must be the case that:

$$\sum_{j|k=K} x_I^{(j)} = \sum_{j|k=K}\mu_{KI} = \text{(count of class K in D)} \cdot \mu_{KI}$$

$$\boxed{\mu_{KI} = \frac{\sum_{j|k=K} x_I^{(j)}}{\text{(count of class K in D)}}}$$

Note that this is another way of stating the MLE for μ_{ik} from Lec. 14, *Probabilistic Models II*, slide 38:

$$\mu_{ik} = \frac{\sum_{n=1}^{N}\mathbb{1}\left[t^{(n)} = k\right] \cdot x_i^{(n)}}{\sum_{n=1}^{N}\mathbb{1}\left[t^{(n)} = k\right]}$$

For any individual σ_I^2:

$$\frac{\partial NLL}{\partial \sigma_I^2} = \sum_{j=1}^{N}(-1)\sum_{i=1}^{D}\frac{1}{2\pi\sigma_i^2} \cdot 2\pi + \sum_{i=1}^{D}(x_i^{(j)} - \mu_{ki})^2 \cdot \frac{-1}{(\sigma^2)^2} \qquad \text{but only where i=I}$$
$$= \sum_{j=1}^{N}(-1)\big(\frac{1}{\sigma_I^2} - (x_I^{(j)} - \mu_{kI})^2 \cdot \frac{1}{(\sigma_I^2)^2}\big)$$

In order for the derivative to be 0 it must be the case that:

$$\sum_{j=1}^{N}\frac{1}{\sigma_I^2} = \sum_{j=1}^{N}\frac{(x_I^{(j)} - \mu_{kI})^2}{(\sigma_I^2)^2}$$
$$\sum_{j=1}^{N} 1 = \sum_{j=1}^{N}\frac{(x_I^{(j)} - \mu_{kI})^2}{(\sigma_I^2)} \qquad \text{(Multiply both sides by } \sigma_I^2\text{)}$$
$$N = \sum_{j=1}^{N}\frac{(x_I^{(j)} - \mu_{kI})^2}{(\sigma_I^2)}$$

Since σ_I^2 is shared across classes, we can take it out of the summation across all N since its value does not depend on j:

$$N = \frac{1}{(\sigma_I^2)}\sum_{j=1}^{N}(x_I^{(j)} - \mu_{kI})^2$$

$$\boxed{(\sigma_I^2) = \frac{\sum_{j=1}^{N}(x_I^{(j)} - \mu_{kI})^2}{N}}$$

Note that this is a special case of the MLE for σ^2 from Lec. 14, *Probabilistic Models II*, slide 38. We sum across all values of k for any given i, since σ^2 is shared across classes:

$$\sigma_{ik}^2 = \frac{\sum_{n=1}^{N}\mathbb{1}\left[t^{(n)} = k\right] \cdot \left(x_i^{(n)} - \mu_{ik}\right)^2}{\sum_{n=1}^{N}\mathbb{1}\left[t^{(n)} = k\right]}$$

d) Show the MLE for $\alpha_k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[y^{(i)} = k\right]$:

All priors must sum to 1. We'll use this as our Lagrangian constraint: $\alpha_1 + \alpha_2 + ... + \alpha_k - 1 = 0$

Then the Lagrangian of our NLL from b) is, with colors for clarity:

$$
\begin{aligned}
&- ln(p(D|\mu, \sigma, \alpha)) + \lambda(constraint) \\
=&\sum_{j=1}^{N}\left(\frac{1}{2}ln\Big((\prod_{i=1}^{D} 2\pi\sigma_i^2)\Big) - \sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{ki})^2 - ln\Big(\alpha_k\Big)\right) + \lambda(\alpha_1 + \alpha_2 + ... + \alpha_K - 1)
\end{aligned}
$$

The partial derivative of this with respect to μ and σ do not change from part c); what we are interested in is the partial with respect to any individual α_k, for k = 1 to K.

Recall that the $-ln(\alpha_k)$ term comes from the **prior** in the likelihood expression that we found in part b). Therefore for any specific set of data D, (α_k) only appears in the product as many times as the class k appears in the data D. Therefore in the NLL, $-ln(\alpha_k)$ only appears this many times in the sum.

$$\frac{\partial \mathcal{L}}{\partial \alpha_k} = \text{(count of class k in D)} \cdot \frac{-1}{\alpha_k} + \lambda \qquad \text{(and this is true for all k from 1 to K)}$$

Setting these all to zero, we note that:

$$\frac{\text{(count of class k in D)}}{\lambda} = \alpha_k \tag{1}$$

Substituting this into our Lagrangian constraint:

$$
\begin{aligned}
\frac{\text{(count of class 1 in D)}}{\lambda} + \frac{\text{(count of class 2 in D)}}{\lambda} + ... + \frac{\text{(count of class K in D)}}{\lambda} &= 1 \\
\frac{\sum_{k=1}^{K}\text{(count of class k in D)}}{\lambda} &= 1
\end{aligned}
$$

$$\lambda = \text{(sum of all class counts)} = N \tag{2}$$

We then get from substituting (2) into (1), for any individual class k:

$$\boxed{\begin{aligned} \alpha_k &= \frac{\text{(count of class k in D)}}{N} \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\left[y^{(i)} = k\right] \end{aligned}}$$