Logistic regression and layers

Let's look at how we can specify logistic regression as layers. The ability to specify such models as layers is key to designing neural networks. We will also discuss backpropagation.
Example: 2-class softmax classifier

**Negative log likelihood**

\[
l(\theta) = -\sum_{i=1}^{N} \Pi_0 (y^{(i)}) \log \frac{e^{\mathbf{x}^{(i)T} \theta_1}}{e^{\mathbf{x}^{(i)T} \theta_1} + e^{\mathbf{x}^{(i)T} \theta_2}} + \Pi_1 (y^{(i)}) \log \frac{e^{\mathbf{x}^{(i)T} \theta_2}}{e^{\mathbf{x}^{(i)T} \theta_1} + e^{\mathbf{x}^{(i)T} \theta_2}} \]

Define cost \( C(\theta) \) that we want to minimize to be the negative log likelihood \( l(\theta) \).

**Layer representation**

```
1
|--|---|--|---|--|
x^{(i)} | z^2_1 | z^3_1 | log S(x^{(i)})_1
x_1^{(i)} | x^{(i)T} \theta_1 | log S(x^{(i)})_2
x^{(i)}_M | z^2_2 | z^3_2
z^1

linear layer  log softmax layer  negative log likelihood layer
```

C(\theta)
Chain rule

\[
\frac{\partial f(g(u,v), h(u,v))}{\partial u} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial u} + \frac{\partial f}{\partial h} \frac{\partial h}{\partial u}
\]
Example: 2-class softmax classifier

We can use the *chain rule* to compute $\frac{\partial z^4}{\partial \theta_1}$ and $\frac{\partial z^4}{\partial \theta_2}$.

\[
\frac{\partial z^4}{\partial \theta_1} = \frac{\partial z^4}{\partial z^3_1} \frac{\partial z^3_1}{\partial \theta_1} + \frac{\partial z^4}{\partial z^3_2} \frac{\partial z^3_2}{\partial \theta_1} \\
= \frac{\partial z^4}{\partial z^3_1} \left( \frac{\partial z^3_1}{\partial z^2_1} \frac{\partial z^2_1}{\partial \theta_1} + \frac{\partial z^3_1}{\partial z^2_2} \frac{\partial z^2_2}{\partial \theta_1} \right) \\
+ \frac{\partial z^4}{\partial z^3_2} \left( \frac{\partial z^3_2}{\partial z^2_1} \frac{\partial z^2_1}{\partial \theta_1} + \frac{\partial z^3_2}{\partial z^2_2} \frac{\partial z^2_2}{\partial \theta_1} \right)
\]

We can similarly compute $\frac{\partial z^4}{\partial \theta_2}$.

Recall that $z^4 = l(\theta)$, and we can minimize the $l(\theta)$ using gradient descent using the gradients computed above.
Example: 2-class softmax classifier (layered view)

**Forward pass**
- $z^1 = f(x)$ (input data)
- $z^2 = f(z^1)$ (linear function)
- $z^3 = f(z^2)$ (log softmax)
- $z^4 = f(z^3) = l(\theta)$ (negative log likelihood, cost)

**Backward pass**
- $\delta^l = \frac{\partial l(\theta)}{\partial z^L}$
Example: 2-class softmax classifier (layered view)

Computing $\delta^l$

\[
\delta^4 = \frac{\partial C(\theta)}{\partial z^4} = \frac{\partial z^4}{\partial z^4} = 1
\]

\[
\delta^3_1 = \frac{\partial C(\theta)}{\partial z^3_1} = \frac{\partial C(\theta)}{\partial z^4} \frac{\partial z^4}{\partial z^3_1} = \delta^4 \frac{\partial z^4}{\partial z^3_1}
\]

\[
\delta^3_2 = \frac{\partial C(\theta)}{\partial z^3_2} = \frac{\partial C(\theta)}{\partial z^4} \frac{\partial z^4}{\partial z^3_2} = \delta^4 \frac{\partial z^4}{\partial z^3_2}
\]

\[
\delta^2_1 = \frac{\partial C(\theta)}{\partial z^2_1} = \sum_k \frac{\partial C(\theta)}{\partial z^3_k} \frac{\partial z^3_k}{\partial z^2_1} = \sum_k \delta^3_k \frac{\partial z^3_k}{\partial z^2_1}
\]

\[
\delta^2_2 = \frac{\partial C(\theta)}{\partial z^2_2} = \sum_k \frac{\partial C(\theta)}{\partial z^3_k} \frac{\partial z^3_k}{\partial z^2_2} = \sum_k \delta^3_k \frac{\partial z^3_k}{\partial z^2_2}
\]
For any differentiable layer \( l \)

For a given layer \( l \), with inputs \( z^l_i \) and outputs \( z^l_{k+1} \)

\[
\delta^l_i = \sum_k \delta^{l+1}_k \frac{\partial z^{l+1}_k}{\partial z^l_i}
\]

Similarly, for layer \( l \) that depends upon parameters \( \theta^l \),

\[
\frac{\partial C(\theta)}{\partial \theta^l} = \sum_k \frac{\partial C(\theta)}{\partial z^{l+1}_k} \frac{\partial z^{l+1}_k}{\partial \theta^l} = \sum_k \delta^{l+1}_k \frac{\partial z^{l+1}_k}{\partial \theta^l}
\]

In our 2-class softmax classifier only layer 1 has parameters \((\theta_0 \text{ and } \theta_1)\).
Layered architectures

As long as we have differentiable layers, i.e., we can compute $\frac{\partial z_{k}^{l+1}}{\partial z_{i}^{l}}$, we can use backpropagation to update the parameters $\theta$ to minimize the cost $C(\theta)$.
Backpropagation

- Set $z^1$ equal to input $x$.
- Forward pass: compute $z^2, z^3, \ldots$ layers 1, 2, \ldots activations.
- Set $\delta$ at the last layer equal to 1
- Backward pass: backpropagate deltas all the way to first layer.
- Update $\theta$
- Repeat
Summary

- Layered view of logistic regression and softmax
- Backpropogation