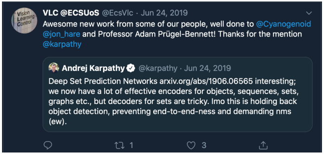
Make a forward pass before the backward pass

Backpropagation: Understanding the implications of the chain rule

Jonathon Hare

Vision, Learning and Control
University of Southampton

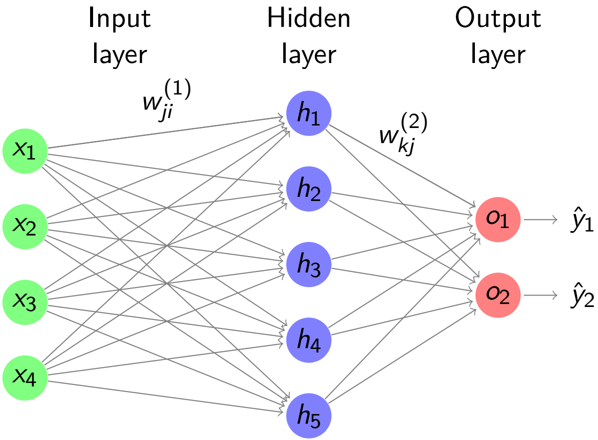
A lot of the ideas in this lecture come from Andrej Karpathy's blog post on backprop (`https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b`) and his CS231n Lecture Notes (`http://cs231n.github.io/optimization-2/`)



Topics

- A quick look at an MLP again
- The chain rule (again)
- Uninititive gradient effects
- A closer look at basic stochastic gradient descent algorithms

The unbiased Multilayer Perceptron (again)...



Without loss of generality, we can write the above as:

$$\hat{\boldsymbol{y}} = g(f(\boldsymbol{x}; \boldsymbol{W}^{(1)}); \boldsymbol{W}^{(2)}) = g(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x}))$$

where f and g are activation functions.

Gradients of our simple unbiased MLP

- Let's assume MSE Loss

$$\ell_{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$$

Gradients of our simple unbiased MLP

- Let's assume MSE Loss

$$\ell_{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$$

- What are the gradients?

$$\nabla_{\boldsymbol{W}^*} \ell_{MSE}(g(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x})), \boldsymbol{y})$$

Gradients of our simple unbiased MLP

- Let's assume MSE Loss

$$\ell_{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$$

- What are the gradients?

$$\nabla_{\boldsymbol{W}^*} \ell_{MSE}(g(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x})), \boldsymbol{y})$$

- Clearly we need to apply the chain rule (vector form) multiple times

Gradients of our simple unbiased MLP

- Let's assume MSE Loss

$$\ell_{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$$

- What are the gradients?

$$\nabla_{\boldsymbol{W}^*} \ell_{MSE}(g(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x})), \boldsymbol{y})$$

- Clearly we need to apply the chain rule (vector form) multiple times
- We could do this by hand

Gradients of our simple unbiased MLP

- Let's assume MSE Loss

$$\ell_{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$$

- What are the gradients?

$$\nabla_{\boldsymbol{W}^*} \ell_{MSE}(g(\boldsymbol{W}^{(2)} f(\boldsymbol{W}^{(1)} \boldsymbol{x})), \boldsymbol{y})$$

- Clearly we need to apply the chain rule (vector form) multiple times
- We could do this by hand
- (But we're not that crazy!)

Let's go back to a simpler expression

$$
\begin{aligned}
f(x, y, z) &= (x + y)z \\
&\equiv qz \text{ where } q = (x + y)
\end{aligned}
$$

Let's go back to a simpler expression

$$
\begin{aligned}
f(x, y, z) &= (x + y)z \\
&\equiv qz \text{ where } q = (x + y)
\end{aligned}
$$

Clearly the partial derivatives of the subexpressions are trivial:

$$
\begin{aligned}
\partial f/\partial z &= q \qquad & \partial f/\partial q &= z \\
\partial q/\partial x &= 1 \qquad & \partial q/\partial y &= 1
\end{aligned}
$$

Let's go back to a simpler expression

$$\begin{aligned} f(x, y, z) &= (x + y)z \\ &\equiv qz \text{ where } q = (x + y) \end{aligned}$$

Clearly the partial derivatives of the subexpressions are trivial:

$$\begin{array}{ll} \partial f / \partial z = q & \partial f / \partial q = z \\ \partial q / \partial x = 1 & \partial q / \partial y = 1 \end{array}$$

and the chain rule tells us how to combine these:

$$\begin{aligned} \partial f / \partial x &= \partial f / \partial q \cdot \partial q / \partial x = z \\ \partial f / \partial y &= \partial f / \partial q \cdot \partial q / \partial y = z \end{aligned}$$

Let's go back to a simpler expression

$$
\begin{aligned}
f(x, y, z) &= (x + y)z \\
&\equiv qz \text{ where } q = (x + y)
\end{aligned}
$$

Clearly the partial derivatives of the subexpressions are trivial:

$$
\begin{aligned}
\partial f/\partial z &= q \qquad & \partial f/\partial q &= z \\
\partial q/\partial x &= 1 \qquad & \partial q/\partial y &= 1
\end{aligned}
$$

and the chain rule tells us how to combine these:

$$
\begin{aligned}
\partial f/\partial x &= \partial f/\partial q \cdot \partial q/\partial x = z \\
\partial f/\partial y &= \partial f/\partial q \cdot \partial q/\partial y = z
\end{aligned}
$$

so $\nabla_{[x,y,z]} f = [z, z, q]$

A computational graph perspective

$$f(x, y, z) = (x + y)z$$

An intuition of the chain rule

- Notice how every operation in the computational graph given its inputs can immediately compute two things:
  1. its output value
  2. the *local* gradient of its inputs with respect to its output value
- The chain rule tells us literally that each operation should take its local gradients and multiply them by the gradient that *flows* backwards into it

This is backpropagation

- The backprop algorithm is just the idea that you can perform the forward pass (computing and caching the local gradients as you go),
- and then perform a backward pass to compute the total gradient by applying the chain rule and re-utilising the cached local gradients

This is backpropagation

- The backprop algorithm is just the idea that you can perform the forward pass (computing and caching the local gradients as you go),
- and then perform a backward pass to compute the total gradient by applying the chain rule and re-utilising the cached local gradients
- Backprop is just another name for 'Reverse Mode Automatic Differentiation'...

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
    - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.
- This has implications for e.g. linear classifiers ($\boldsymbol{w}^\top \boldsymbol{x}_i$) where you perform many multiplications

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.
- This has implications for e.g. linear classifiers ($\boldsymbol{w}^\top \boldsymbol{x}_i$) where you perform many multiplications
  - the magnitude of the gradient is directly proportional to the magnitude of the data

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.
- This has implications for e.g. linear classifiers ($\boldsymbol{w}^\top \boldsymbol{x}_i$) where you perform many multiplications
  - the magnitude of the gradient is directly proportional to the magnitude of the data
  - multiple $\boldsymbol{x}_i$ by 1000, and the gradients also increase by 1000

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.
- This has implications for e.g. linear classifiers ($\boldsymbol{w}^\top \boldsymbol{x}_i$) where you perform many multiplications
  - the magnitude of the gradient is directly proportional to the magnitude of the data
  - multiple $\boldsymbol{x}_i$ by 1000, and the gradients also increase by 1000
  - if you don't lower the learning rate to compensate your model might not learn

Unintuitive effects I: Multiplication

- Consider the multiplication operation $f(a, b) = a * b$.
- The gradients are clearly $\partial f / \partial b = a$ and $\partial f / \partial a = b$.
  - (in a computational graph these would be the local gradients w.r.t the inputs)
- If a is large and b is tiny the gradient assigned to b will be large, and the gradient to a small.
- This has implications for e.g. linear classifiers ($\boldsymbol{w}^\top \boldsymbol{x}_i$) where you perform many multiplications
  - the magnitude of the gradient is directly proportional to the magnitude of the data
  - multiple $\boldsymbol{x}_i$ by 1000, and the gradients also increase by 1000
  - if you don't lower the learning rate to compensate your model might not learn
  - **Hence you need to always pay attention to data normalisation!**

Unintuitive effects II: vanishing gradients of the sigmoid

Unintuitive effects III: dying ReLUs

Unintuitive effects III: Exploding gradients in recurrent networks