CSE 152: Computer Vision

Hao Su

Lecture 8: Network Optimization

Credit: Manmohan Chandraker

Gradient descent

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

parameters to compute

# of input samples

$$\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2$$

Small changes in parameters to leads to small changes in output

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2} \right)^T$$

Gradient vector!

$$\Delta v = -\eta \nabla C$$

Change the parameter using learning rate (positive) and gradient vector!

$$v \rightarrow v' = v - \eta \nabla C$$

Update rule!

Stochastic gradient descent

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

parameters to compute

# of input samples

Update rules for each parameter:

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

Cost function is a sum over all the training samples:

$$C(w,b) = \sum_x C_x(w,b), \text{ where } C_x(w,b) = \frac{1}{2}\|y(x) - a\|^2$$

Gradient from entire training set:

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x$$

Usually, *n* is very large.

Stochastic gradient descent

Gradient from entire training set:

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x$$

- For large training data, gradient computation takes a long time
  - Leads to “slow learning”

- Instead, consider a mini-batch with *m* samples
- If sample size is large enough, properties approximate the dataset

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C$$

Stochastic gradient descent

What if the loss function has a **local minima** or **saddle point**?

Zero gradient, gradient descent gets stuck

Stochastic gradient descent

What if loss changes quickly in one direction and slowly in another?
What does gradient descent do?
Very slow progress along shallow dimension, jitter along steep direction

Loss function has high **condition number**: ratio of largest to smallest singular value of the Hessian matrix is large

Stochastic gradient descent

Our gradients come from minibatches so they can be noisy!

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W)$$

Stochastic gradient descent

Momentum update:

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

Build up velocity as a running mean of gradients.

Layer to layer relationship

$$a_j^l = \sigma(z_j^l)$$

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

- b_j^l is the bias term in the jth neuron in the lth layer.
- a_j^l is the activation in the jth neuron in the lth layer.
- z_j^l is the weighted input to the jth neuron in the lth layer.

Cost and gradient computation

Groundtruth for each input

Output activation vector for a specific training sample x.

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$

# of input samples

for each input sample

- The goal of the backpropagation algorithm is to compute the gradients $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ of the cost function C with respect to each and every weight and bias parameters. Note that backpropagation is only used to compute the gradients.

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$

- Stochastic gradient descent is the training algorithm.

Chain rule of differentiation

- In order to differentiate a function $z = f\big(g(x)\big)$ w.r.t x, we can do the following:

Let $y = g(x)$, $z = f(y)$, $\frac{dz}{dx} = \frac{dz}{dy} \times \frac{dy}{dx}$

Let $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, g maps from $\mathbb{R}^m$ to $\mathbb{R}^n$, and f maps from $\mathbb{R}^n$ to $\mathbb{R}$. If $y = g(x)$ and $z = f(y)$, then

$$\frac{\partial z}{\partial x_i} = \sum_k \frac{\partial z}{\partial y_k} \frac{\partial y_k}{\partial x_i}$$

This is all you need to know to get the gradients in a neural network!

Backpropagation

- In order to differentiate a function $z = f\big(g(x)\big)$ w.r.t x, we can do the following:

Let $y = g(x)$, $z = f(y)$, $\frac{dz}{dx} = \frac{dz}{dy} \times \frac{dy}{dx}$

Let $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, g maps from $\mathbb{R}^m$ to $\mathbb{R}^n$, and f maps from $\mathbb{R}^n$ to $\mathbb{R}$. If $y = g(x)$ and $z = f(y)$, then

$$\frac{\partial z}{\partial x_i} = \sum_k \frac{\partial z}{\partial y_k} \frac{\partial y_k}{\partial x_i}$$

This is all you need to know to get the gradients in a neural network!

Backpropagation: application of chain rule in certain order, taking advantage of forward propagation to efficiently compute gradients.

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

[Slides credit: Fei-Fei Li]

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial y}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation example

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial x}$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Backpropagation example

Backpropagation example

Backpropagation example

Backpropagation example

Over-fitting

- Instead of 60000 training images, we use only 1000 training images and check the performance on the test data.

More data prevents over-fitting

But not always feasible to have more data that is relevant.

Regularization reduces over-fitting

- If not enough data, can instead limit model complexity.
- Regularization places constraints on the model, so reduces its complexity.

L2 regularization

L2 regularization:

Let C_0 be original cost. Define $C = C_o + \frac{\lambda}{2n}\sum_w w^2$

- The first term is just the usual $C_o \equiv \frac{1}{2n}\sum \|y(x) - a\|^2$
- Here λ is the regularization parameter and n is the size of our training set.

Partial derivatives:

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w$$

Update rule:

$$w \rightarrow w - \frac{\eta \partial C_0}{\partial w} - \frac{\eta \lambda}{n} w$$

Regularization reduces over-fitting

Before

After

Dropout as a regularization

- Modify the network itself
  - Randomly delete half the hidden neurons in the network
  - Repeat several times to learn weights and biases
  - At runtime, twice as many neurons, so halve the weights outgoing from a neuron

Dropout as a regularization

- Averaging or voting scheme to decide output
  - Forces neurons to learn independent of others
  - Same training data, but random initializations
  - Each network over-fits in a different way
  - Robustness: average output not sensitive to particular mode

Data augmentation as regularization

Data augmentation as regularization

Horizontal flips

Data augmentation as regularization

Random crops and scales

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch

Data augmentation as regularization

Color jitter

Simple: Randomize contrast and brightness

More Complex:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a “color offset” along principal component directions
3. Add offset to all pixels of a training image

Data augmentation as regularization

Color jitter

Simple: Randomize contrast and brightness

More Complex:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a "color offset" along principal component directions
3. Add offset to all pixels of a training image

Can do a lot more: rotation, shear, non-rigid, motion blur, lens distortions, ….