Hyperparameter tuning

Tuning process
Hyperparameters

$\alpha$

$\beta$, $\gamma$

$\beta_1$, $\beta_2$, $\epsilon$

$0.9$, $10^{-4}$, $10^{-8}$

#layers

#hidden units

learning rate decay

mini-batch size
Try random values: Don’t use a grid
Coarse to fine

Hyperparameter 1

Hyperparameter 2

Andrew Ng
Hyperparameter tuning

Using an appropriate scale to pick hyperparameters
Picking hyperparameters at random

\[ n^{\text{test}} = 50, \ldots, 100 \]

\[ \begin{array}{cccccccc}
\times & \times & \times & \times & \times & \times & \times & \times \\
\end{array} \]

\[ 50 \quad 100 \]

\[ \Rightarrow \text{\#layers } L : 2 - 4 \]

\[ 2, 3, 4 \]
Appropriate scale for hyperparameters

\[ \begin{align*}
\theta & = 0.0001, \\
\epsilon & = 0.0001, \\
\gamma & = -4, \\
a & = \log_{10} 0.0001 \\
x & = 10^a \\
r & = -4 \times \text{np.random.rand()} \\
\beta & = \log_{10} 1 \\
f & = \text{[a,b]} \\
d & = 10^d \\
\end{align*} \]
Hyperparameters for exponentially weighted averages

\[ \begin{align*}
\beta &= 0.9 \quad \ldots \quad 0.999 \\
1 - \beta &= 0.1 \quad \ldots \quad 0.001 \\
\beta &= 0.9000 \to 0.9005 \sim 10 \\
1 - \beta &= 10^{-1} \\
\beta &= 0.999 \to 0.9995
\end{align*} \]
Hyperparameters tuning in practice: Pandas vs. Caviar
Re-test hyperparameters occasionally

- NLP, Vision, Speech, Ads, logistics, ....

- Intuitions do get stale. Re-evaluate occasionally.
Babysitting one model

Training many models in parallel

Panda

Caviar
Batch Normalization

Normalizing activations in a network
Normalizing inputs to speed up learning

\[ \mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \]
\[ X = X - \mu \]
\[ \sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2 \]
\[ X = \frac{X}{\sigma^2} \]

Can we normalize \( \frac{a}{w^{[2]}, b^{[2]}} \) so as to turn \( w^{[2]}, b^{[2]} \) faster.

Normalized \( z^{[2]} \)
Implementing Batch Norm

Given some intermediate values in NN $z^{(i)}, \ldots, z^{(m)}$

$$\mu = \frac{1}{m} \sum_{i=1}^{m} z^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (z^{(i)} - \mu)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\gamma(i) = \sum_{i=1}^{m} z_{\text{norm}}^{(i)} + \beta$$

$$\tilde{z}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

If $\gamma(i)$ is learned parameter of model.

Use $\tilde{z}$ instead of $z^{(i)}$.
Batch Normalization

Fitting Batch Norm into a neural network
Adding Batch Norm to a network

\[ x_1, x_2, x_3 \rightarrow z_1, z_2, z_3 \rightarrow a_1, a_2, a_3 \rightarrow \hat{y} \]

Parameters: \( w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \ldots, w^{[L]}, b^{[L]} \)

\( \frac{d\beta}{dL^2} = \beta - \eta \frac{dL^2}{d\beta} \)
Working with mini-batches

Parameters: $W$, $b$, $\beta$, $\gamma$, $\mu$, $\nu$. 

$X \xrightarrow{\times n} 2 \xrightarrow{\beta, \gamma} \sim \nu \xrightarrow{\text{BN}} 2 \rightarrow \ldots$

$X \rightarrow \ldots$

$\rightarrow \hat{z} = \frac{z - \mu}{\nu}$
Implementing gradient descent

for $t = 1, \ldots, \text{num\ Mini\ Batchs}$

Compute forward prop on $X^{t\times3}$

In each hidden layer, use BN to output $Z^{(l)}$ with $Z^{(l)}$

Use backprop to compute $dL^{(l+1)} / dt^{(l+1)}, \frac{dL^{(l+1)}}{dp^{(l+1)}}, \frac{dL^{(l+1)}}{d\beta^{(l+1)}}$

Update params

$W^{(l)} := W^{(l)} - \alpha \frac{dL^{(l+1)}}{dW^{(l)}}$

$\beta^{(l)} := \beta^{(l+1)} - \alpha \frac{dL^{(l+1)}}{d\beta^{(l+1)}}$

$p^{(l)} := \ldots$

Works w/ moment, RMSprop, Adam.
Why does Batch Norm work?
Learning on shifting input distribution

\[ x_1 \xrightarrow{} \hat{y} \]

Cat \( y = 1 \)
Non-Cat \( y = 0 \)

\[ y = 1 \leftarrow \text{Cat} \]
\[ y = 0 \leftarrow \text{Non-Cat} \]

"Coordinate shift"
\[ x \rightarrow y \]
Why this is a problem with neural networks?
Batch Norm as regularization

• Each mini-batch is scaled by the mean/variance computed on just that mini-batch.

• This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer’s activations.

• This has a slight regularization effect.
Multi-class classification

Softmax regression
Recognizing cats, dogs, and baby chicks
Softmax layer
Softmax examples
Programming Frameworks

Deep Learning frameworks

deeplearning.ai
Deep learning frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Theano
- Torch

Choosing deep learning frameworks
- Ease of programming (development and deployment)
- Running speed
- Truly open (open source with good governance)
Motivating problem

\[ J(\omega) = \begin{cases} 
\omega^2 - 10\omega + 25 \\
(\omega - 5)^2 
\end{cases} \]

\[ \omega = 5 \]

\[ J(\omega, b) \]

Andrew Ng
import numpy as np
import tensorflow as tf

coefficients = np.array([[1], [-20], [25]])

w = tf.Variable([0], dtype=tf.float32)
x = tf.placeholder(tf.float32, [3,1])

cost = x[0][0]*w**2 + x[1][0]*w + x[2][0]  # (w-5)**2
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()

with tf.Session() as session:
    session.run(init)
    print(session.run(w))
    for i in range(1000):
        session.run(train, feed_dict={x:coefficients})
        print(session.run(w))