Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode
Hyperparameter tuning

Tuning process

deeplearning.ai
Hyperparameters

\[ \alpha \]

\[ \beta \]

\[ \beta_1, \beta_2, \gamma \]

- #layers
- #hidden units
- learning rate decay
- mini-batch size
Try random values: Don’t use a grid
Coarse to fine
Hyperparameter tuning

Using an appropriate scale to pick hyperparameters
Picking hyperparameters at random

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

\[ \begin{array}{c}
  50 \\
  \times \times \times \times \times \times \times \times \\
  100 
\end{array} \]

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

2, 3, 4
Appropriate scale for hyperparameters

\[
\alpha = 0.0001, \quad \ldots, \quad 1
\]

\[
\theta = \frac{1}{n-1} \sum_{i=1}^{n} (\alpha y_i - \theta x_i)^2
\]

\[
a = \log_{10} 0.0001, \quad r = -4 \times \text{np.random.rand()} \leq r \in [-4, 0]
\]

\[
d = 10^5, \quad 10^4 \ldots 10^6
\]

\[
10^a \ldots 10^b, \quad r \in [a, b], \quad d = 10^c
\]
Hyperparameters for exponentially weighted averages

\[ \beta = 0.9 \quad \ldots \quad 0.999 \]

\[ \sqrt{10} \quad \sqrt{1000} \]

\[ 1 - \beta = 0.1 \quad \ldots \quad 0.001 \]

\[ \beta : 0.9 \rightarrow 0.9005 \sim 10 \]

\[ \beta : 0.999 \rightarrow 0.9995 \sim 1000 \quad \sim 2000 \]

\[ \frac{1}{1 - \beta} \]

Andrew Ng
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

Andrew Ng
Batch Normalization

Normalizing activations in a network
Normalizing inputs to speed up learning

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

Can we normalize \( \text{a} \) to train \( w^{[2]}, b^{[2]} \) faster?

Normalizing \( Z \)
Implementing Batch Norm

Given some intermediate values in NN $z^{(1)}, \ldots, z^{(m)}$

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

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

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

$$\gamma^{(i)} = \frac{z_{\text{norm}}^{(i)}}{\epsilon + \sigma^2}$$

$$\text{Use } \gamma^{(i)} \text{ instead of } \frac{z^{(i)}}{\sigma}$$

If

$$\gamma = \frac{1}{\sqrt{\sigma_2}}$$

$$\beta = \frac{\mu}{\sigma}$$

Then $z^{(i)} = \gamma^{(i)} z^{(i)}$

Learnable parameters of model.
Batch Normalization

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

![Diagram of a neural network with Batch Normalization layers]

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

Gradient of Batch Norm:

$$d\beta^{[l]} = \frac{\partial L}{\partial \beta^{[l]}}$$

$$\beta = \beta - \alpha d\beta^{[l]}$$

Andrew Ng
Working with mini-batches

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

$X \times r^2 \xrightarrow{\omega \times b} z_1 \xrightarrow{x} \beta \xrightarrow{\gamma} z_2 \xrightarrow{\beta} \ldots$

$X \rightarrow \ldots$

$X \rightarrow \ldots$

Andrew Ng
Implementing gradient descent

for \( t = 1 \ldots \) numMiniBatches

    Compute forward pass on \( X^{\text{test}} \).

    In each hidden layer, use BN to renormalize \( \bar{z}_t \) with \( \overline{\bar{z}}_t \).

    Use backprop to compute \( \frac{\partial L}{\partial \bar{w}_t}, \frac{\partial L}{\partial \bar{b}_t}, \frac{\partial L}{\partial \bar{\beta}_t} \).

    Update parameters \( \overline{\bar{w}}_t := \overline{\bar{w}}_t - \alpha \frac{\partial L}{\partial \bar{w}_t} \), \( \overline{\bar{\beta}}_t := \overline{\bar{\beta}}_t - \alpha \frac{\partial L}{\partial \bar{\beta}_t} \), \( \overline{\bar{b}}_t := \ldots \).

Works w/ moment, RMSprop, Adam.
Batch Normalization

Why does Batch Norm work?
Learning on shifting input distribution

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

\[ x_2 \rightarrow \hat{y} \]

\[ x_3 \rightarrow \hat{y} \]

Cat \hspace{1cm} Non-Cat

\[ y = 1 \rightarrow \hat{y} \]

\[ y = 0 \rightarrow \hat{y} \]

"Coordinate shift"

\[ \hat{y} \rightarrow y \]
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)
Programming Frameworks

deeplearning.ai

TensorFlow
Motivating problem

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

\[\omega = 5\]

\[J(\omega, b)\]
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])

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))