**neural\_net.py**

1. overall explanation

: two-layer fully-connected neural network를 구하기 위한 클래스 TwoLayerNet 안에 모듈 \_\_init\_\_(), loss(), train(), predict() 가 정의되어 있다.

2. code

z1 = X.dot(W1) + b1 # First layer pre-activation

a1 = np.maximum(0, z1) # First layer activation(relu)

z2 = a1.dot(W2) + b2 # Second layer pre-activation

scores = z2

# Second layer activation (softmax)

exp\_scores = np.exp(scores)

a2 = exp\_scores / np.sum(exp\_scores, axis=1, keepdims=True) #softmax activation function

log\_prob = -np.log(a2[range(N), y]) #log-likelihood probability (y is ground truth)

data\_loss = np.sum(log\_prob) / N #data loss

reg\_loss = 0.5 \* reg \* np.sum(W1 \* W1) + 0.5 \* reg \* np.sum(W2 \* W2) #regularization loss(L2)

loss = data\_loss + reg\_loss # compute totall loss

indices = np.random.choice(np.arange(num\_train), batch\_size)

X\_batch = X[indices]

y\_batch = y[indices]

# Let z be the result after W \* X, z after activation function becomes y.

#dscores: ∂L/∂z2

dscores = a2

dscores[range(N),y] -= 1

dscores /= N

# gradients on W2, b2

grads['W2'] = np.dot(a1.T, dscores) #∂L/∂W2 = (∂L/∂z2) a1^T

grads['b2'] = np.sum(dscores, axis=0) #∂L/∂b2 = ∂L/∂z2

# z2에서 a1을 역으로 구한다: s=Wx+b, L is loss 일 때 ∂L/∂x = W^T (∂L/∂s) 임을 이용

# ∂L/∂z1 = W2^T (∂L/∂z2)

dhidden = np.dot(dscores, W2.T)

dhidden[a1 <= 0] = 0 #ReLu function 역으로

# gradients on W1, b1

grads['W1'] = np.dot(X.T, dhidden) #∂L/∂W1 = (∂L/∂z1) x^T

grads['b1'] = np.sum(dhidden, axis=0) #∂L/∂b1 = ∂L/∂z1

#regularize the gradients

grads['W2'] += reg \* W2

grads['W1'] += reg \* W1

self.params['W1'] += -learning\_rate \* grads['W1']

self.params['b1'] += -learning\_rate \* grads['b1']

self.params['W2'] += -learning\_rate \* grads['W2']

self.params['b2'] += -learning\_rate \* grads['b2']

z1 = X.dot(self.params['W1']) + self.params['b1'] # z1 = W1X + b1

a1 = np.maximum(0, z1) # relu funtion

scores = a1.dot(self.params['W2']) + self.params['b2'] # z2 = W2X + b2

y\_pred = np.argmax(scores, axis=1) # score에서 최대값 index 리턴-> lable for each elements of X가 된다

3. code explanation & analysis

이 network의 구조는 다음과 같다.

Input(x) - fully connected layer - ReLU - fully connected layer - softmax

First fully connected layer: z1 = W1x+b1

ReLU: a1 = max(0, z1)

Second fully connected layer: z2(scores) = W2a1 + b2

Softmax: a2 = exp(scores) / np.sum(exp\_scores, axis=1, keepdims=True)

Input data를 받아 second fully connected layer의 결과로 각 class에 대한 score를 구한 후, 이 score와 softmax classifier를 이용하여 loss function을 구한다. 그리고 data loss와 L2 regularization의 합인 total loss를 구하여 network를 train한다.

**two\_layer\_net.ipynb**

1. overall explanation

: neural\_net.py를 이용하여 two layer neural network를 구현한 뒤, CIFAR-10 dataset을 이용하여 classifier를 train시킨다. 그리고 test accuracy가 0.36 이상인 최종 trained network, 즉 best model을 구한다.

2. code

best\_val = -1 #validation accuracy of best network

best\_stats = None

learning\_rates = [1e-2, 1e-3]

regularization\_strengths = [0.4, 0.5, 0.6]

results = {}

iters = 2000

# find the best network

for lr in learning\_rates:

for rs in regularization\_strengths:

net = TwoLayerNet(input\_size, hidden\_size, num\_classes)

# training the network

stats = net.train(X\_train, y\_train, X\_val, y\_val,

num\_iters=iters, batch\_size=200,

learning\_rate=lr, learning\_rate\_decay=0.95,

reg=rs)

# compute train accuracy and validation accuracy

y\_train\_pred = net.predict(X\_train)

acc\_train = np.mean(y\_train == y\_train\_pred)

y\_val\_pred = net.predict(X\_val)

acc\_val = np.mean(y\_val == y\_val\_pred)

results[(lr, rs)] = (acc\_train, acc\_val)

# if the validation accuracy is largest, then it is the best network

if best\_val < acc\_val:

best\_stats = stats

best\_val = acc\_val

best\_net = net

# Print out results.

for lr, reg in sorted(results):

train\_accuracy, val\_accuracy = results[(lr, reg)]

print('lr %e reg %e train accuracy: %f val accuracy: %f' % (

lr, reg, train\_accuracy, val\_accuracy))

print('best validation accuracy achieved during cross-validation: %f' % best\_val)

3. code explanation & analysis

1) Create a small net and toy data

2) Implement neural network

* Forward pass: open the file classifiers/neural\_net.py and use loss function to compute the scores for all inputs. After computing score, use loss function again and compute the data and L2 regularization loss.
* Backward pass: compute the gradient of the loss with respect to W1, W2, b1, b2 using the loss function

3) Train the network

* We will train two-layer neural network on toy data. To train the network we will use stochastic gradient descent (SGD).

4) Load the data from CIFAR-10 dataset so we can use it to train a classifier on a real dataset

5) Train a network using the data from CIFAR-10 dataset

* we train the network using train function and predict on the validation set using predict function.

6) Debug the training

7) Tune your hyperparameters

* classification accuracy should be greater than 36% on the validation set. Best network gets over 39% on the validation set.

8) Run on the test set and evaluate the final trained network on the test set.

4. Results analysis

Forward pass: compute scores

-compute scores for all inputs and compute the difference between our scores and correct scores

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

Forward pass: compute loss

-compute total loss and difference between our loss and correct loss

텍스트이(가) 표시된 사진

자동 생성된 설명

Backward pass

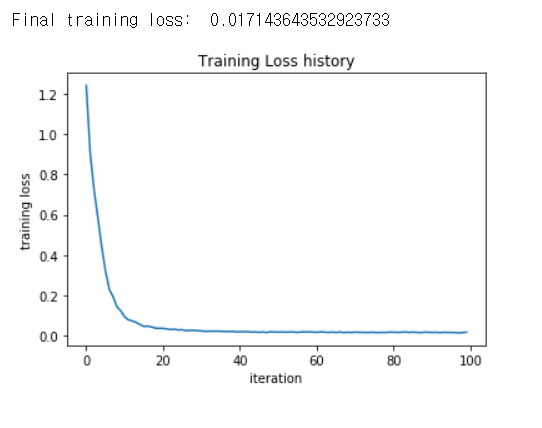
-compute relative error(difference between the numeric and analytic gradients) for W1, W2, b1, and b2.

텍스트이(가) 표시된 사진

자동 생성된 설명

Train the network

-plot the training loss history and compute the final training loss



Load the data

-load CIFAR-10 data and print shape of the data

텍스트이(가) 표시된 사진

자동 생성된 설명

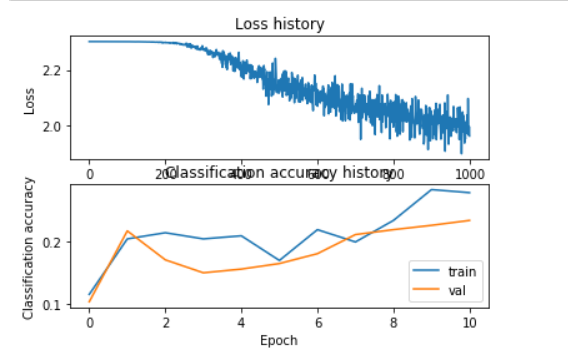
Train a network

텍스트이(가) 표시된 사진

자동 생성된 설명

Debug the training

-plot the loss history and classification accuracy history



-visualize the weights of the network

건물이(가) 표시된 사진

자동 생성된 설명

Tune your hyperparameters

-find the best network and its accuracy

텍스트이(가) 표시된 사진

자동 생성된 설명-visualize the weights of the best network

화이트보드이(가) 표시된 사진

자동 생성된 설명

Run on the test set

-evaluate final trained network on the test set

텍스트이(가) 표시된 사진

자동 생성된 설명