With PyTorch 0.2.0_2

PYTORCH

Lab 6

Softmax Classifier

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

With PyTorch 0.2.0_2

PYTORCH

Lab 6-1

Softmax Classifier

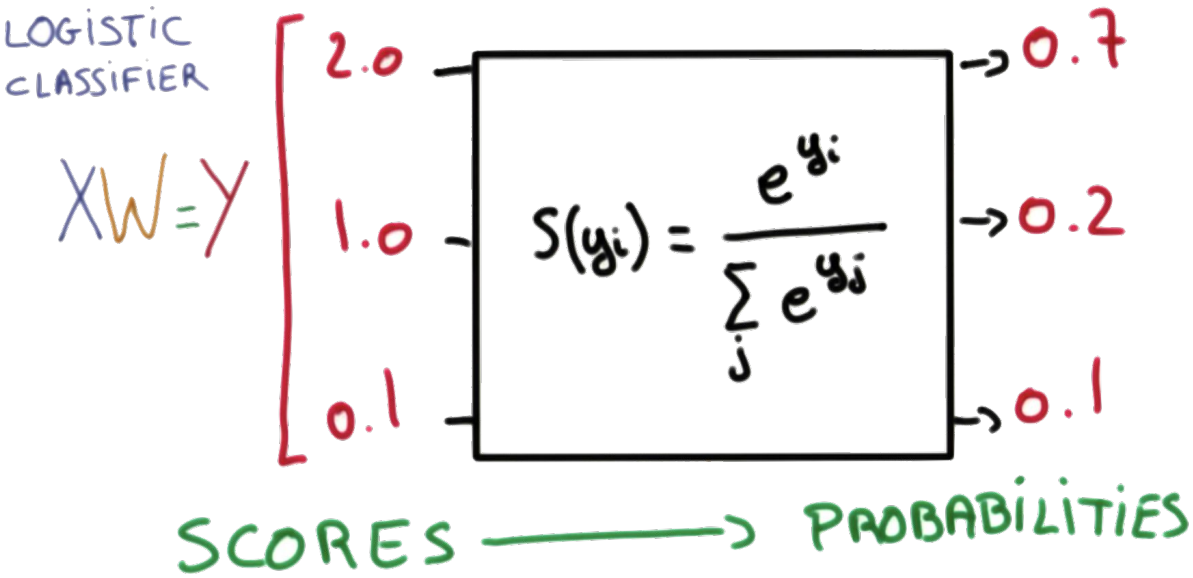
Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

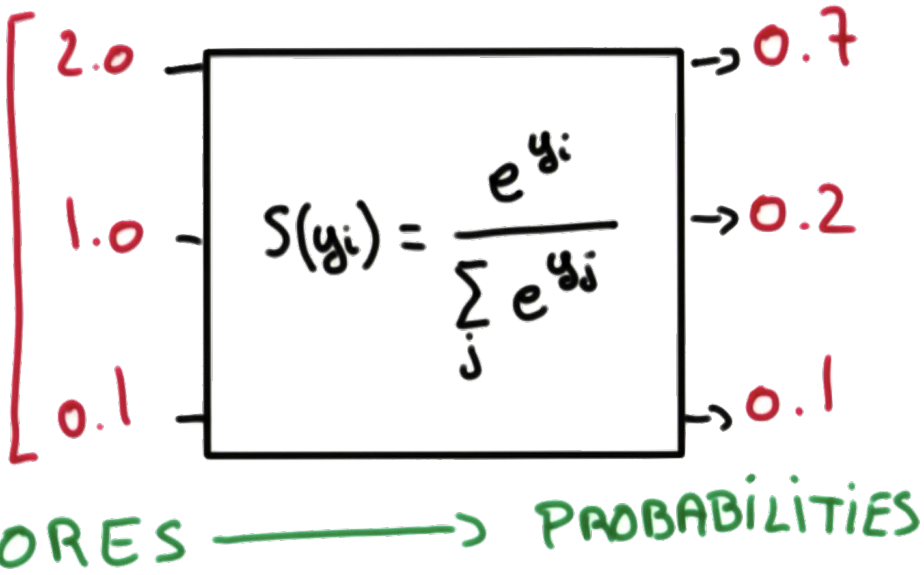
Softmax function



https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

LOGISTIC CLASSIFIER

$XW = Y$




```
# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
softmax = torch.nn.Softmax()
linear = torch.nn.Linear(4, nb_classes, bias=True)
model = torch.nn.Sequential(linear, softmax)
```

https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

Cost function: cross entropy

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(2001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # Cross entropy cost/loss
    cost = -Y * torch.log(hypothesis)
    cost = torch.sum(cost, 1).mean()
    cost.backward()
    optimizer.step()

    if step % 200 == 0:
        print(step, cost.data.numpy())
```

LOSS

$$\mathcal{L} = \frac{1}{N} \sum_i D(S(wx_i + b), L_i)$$

TRAINING SET

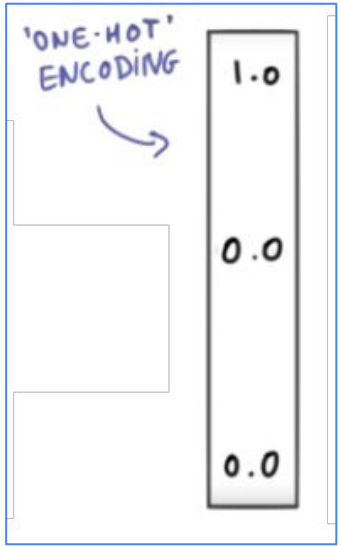
STEP

$$-\alpha \underbrace{\Delta \mathcal{L}(w_1, w_2)}_{\text{DERIVATIVE}}$$

https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

```python
# Lab 6 Softmax Classifier
import torch
from torch.autograd import Variable

torch.manual_seed(777)  # for reproducibility

x_data = [[1, 2, 1, 1], [2, 1, 3, 2], [3, 1, 3, 4], [4, 1, 5, 5],
          [1, 7, 5, 5], [1, 2, 5, 6], [1, 6, 6, 6], [1, 7, 7, 7]]
y_data = [[0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 1, 0],
          [0, 1, 0], [0, 1, 0], [1, 0, 0], [1, 0, 0]]

X = Variable(torch.Tensor(x_data))
Y = Variable(torch.Tensor(y_data))
nb_classes = 3

# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
softmax = torch.nn.Softmax()
linear = torch.nn.Linear(4, nb_classes, bias=True)
model = torch.nn.Sequential(linear, softmax)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(2001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # Cross entropy cost/loss
    cost = -Y * torch.log(hypothesis)
    cost = torch.sum(cost, 1).mean()
    cost.backward()
    optimizer.step()

    if step % 200 == 0:
        print(step, cost.data.numpy())
```

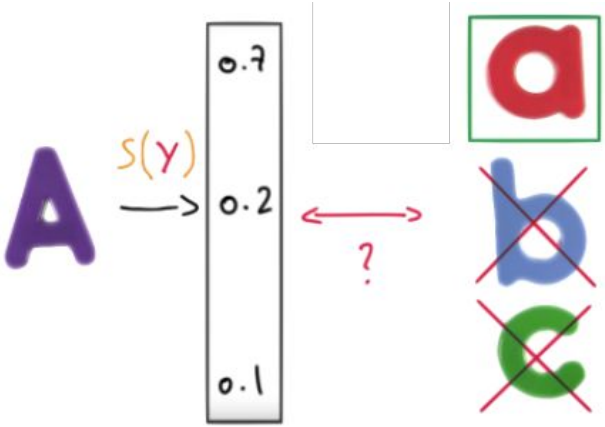
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py



Test & one-hot encoding

```
# Testing & One-hot encoding
print('--------------')
a = model(Variable(torch.Tensor([[1, 11, 7, 9]])))
print(a.data.numpy(), torch.max(a, 1)[1].data.numpy())
```

[[1.38904958e-03 9.98601854e-01 9.06129117e-06]] [1]



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

Test & one-hot encoding

```
all = model(Variable(torch.Tensor([[1, 11, 7, 9], [1, 3, 4, 3], [1, 1, 0, 1]])))
print(all.data.numpy(), torch.max(all, 1)[1].data.numpy())
```

```
[[ 1.38904958e-03  9.98601854e-01  9.06129117e-06]
[ 9.31192040e-01  6.29020557e-02  5.90589503e-03]
[ 1.27327668e-08  3.34112905e-04  9.99665856e-01]]
```

[1 0 2]

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

With PyTorch 0.2.0_2

PYTORCH

Lab 6-2

Fancy Softmax Classifier

cross_entropy, one_hot, reshape

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-2-softmax_zoo_classifier.py

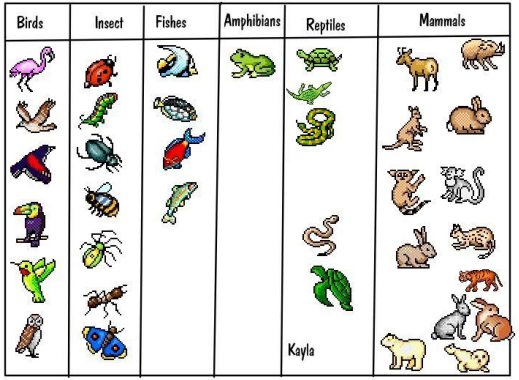
softmax_cross_entropy

```
softmax = torch.nn.Softmax()
model = torch.nn.Linear(16, nb_classes, bias=True)
```

```
# Cross entropy cost/loss
criterion = torch.nn.CrossEntropyLoss()

# Softmax is internally computed.
```

Animal classification

with *softmax_cross_entropy*



1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

```
# Predicting animal type based on various features
xy = np.loadtxt('data-04-zoo.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

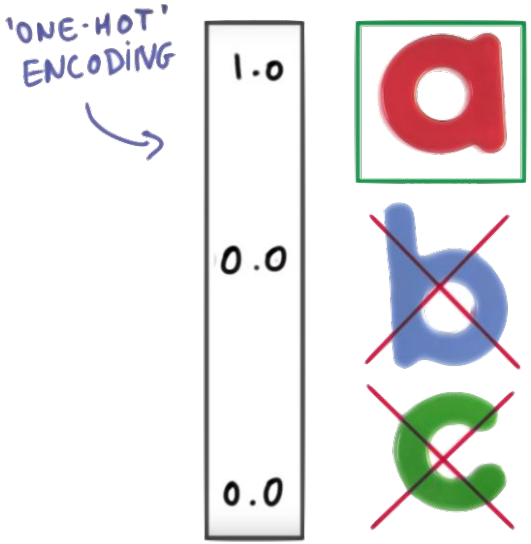
https://kr.pinterest.com/explore/animal-classification-activity/

tf.one_hot and reshape

1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

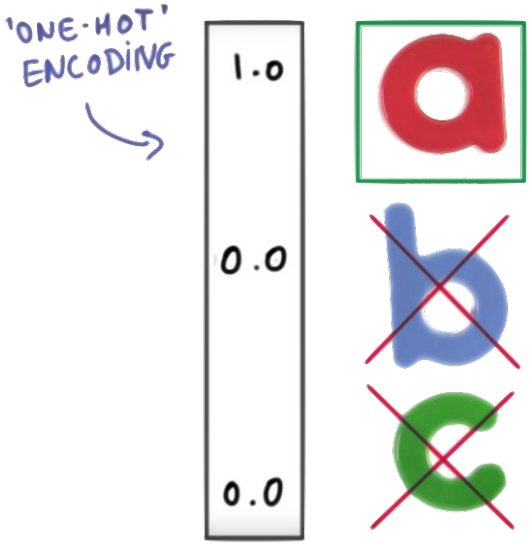
```
# one hot encoding
Y_one_hot = torch.zeros(Y.size()[0], nb_classes)
Y_one_hot.scatter_(1, Y.long().data, 1)
Y_one_hot = Variable(Y_one_hot)
print("one_hot", Y_one_hot.data)
```

```python
# Predicting animal type based on various features
xy = np.loadtxt('data-04-zoo.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

print(x_data.shape, y_data.shape)

nb_classes = 7  # 0 ~ 6

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))

# one hot encoding
Y_one_hot = torch.zeros(Y.size()[0], nb_classes)
Y_one_hot.scatter_(1, Y.long().data, 1)
Y_one_hot = Variable(Y_one_hot)
print("one_hot", Y_one_hot.data)

softmax = torch.nn.Softmax()
model = torch.nn.Linear(16, nb_classes, bias=True)
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-2-softmax_zoo_classifier.py

```python
# Cross entropy cost/loss
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(2001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # Label has to be 1D LongTensor
    cost = criterion(hypothesis, Y.long().view(-1))
    cost.backward()
    optimizer.step()

    prediction = torch.max(softmax(hypothesis), 1)[1].float()

    correct_prediction = (prediction.data == Y.data)
    accuracy = correct_prediction.float().mean()

    if step % 100 == 0:
        print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(step, cost.data[0], accuracy))


# Let's see if we can predict
pred = torch.max(softmax(hypothesis), 1)[1].float()

for p, y in zip(pred, Y):
    print("[{}] Prediction: {} True Y: {}".format(bool(p.data[0] == y.data[0]), p.data.int()[0], y.data.int()[0]))
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-2-softmax_zoo_classifier.py

With PyTorch 0.2.0_2

PYTORCH

Lab 9

NN for XOR

Sung Kim <hunkim+ml@gmail.com>