

# Understanding model optimization

Puteaux, Fall/Winter 2020-2021

```
#####  
##                               ##  
## Deep Learning in Python  ##  
##                               ##  
#####
```

\$1 Introduction to Deep Learning in Python

\$1.4 Fine-tuning keras models

\$1.4.1 Understanding model optimization

## 1. Why is optimization hard?

- Simultaneously optimize 1000's of parameters with complex relationships.
- Updates may not improve the model meaningfully.
- Updates will be too small (if the learning rate is low) or too large (if the learning rate is high).

## 2. Code of stochastic gradient descent:

```
[1]: import pandas as pd  
from keras.layers import Dense  
from keras.models import Sequential  
from keras.utils.np_utils import to_categorical  
from keras.optimizers import SGD  
  
def data_preparation(df):  
    df = df.reindex(columns=[  
        'SHOT_CLOCK', 'DRIBBLES', 'TOUCH_TIME', 'SHOT_DIST', 'CLOSE_DEF_DIST',  
        'SHOT_RESULT'  
    ])  
    df['SHOT_CLOCK'] = df['SHOT_CLOCK'].fillna(0)  
    df['SHOT_RESULT'].replace('missed', 0, inplace=True)  
    df['SHOT_RESULT'].replace('made', 1, inplace=True)  
    df.columns = df.columns.str.lower()  
    return df
```

```
data = pd.read_csv('ref1. Basketball shot log.csv')
data = data_preparation(data)

predictors = data.drop(['shot_result'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(data.shot_result)
input_shape = (n_cols, )
```

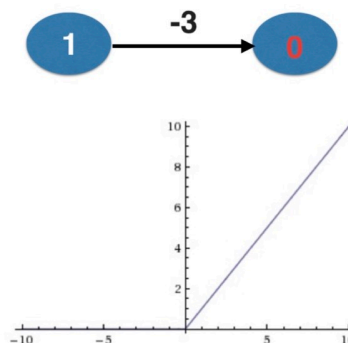
```
[2]: def get_new_model(input_shape=input_shape):
    model = Sequential()
    model.add(Dense(100, activation='relu', input_shape=input_shape))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(2, activation='softmax'))
    return (model)

lr_to_test = [.000001, 0.01, 1]

# loop over learning rates
for lr in lr_to_test:
    model = get_new_model()
    my_optimizer = SGD(lr=lr)
    model.compile(optimizer=my_optimizer, loss='categorical_crossentropy')
    model.fit(predictors, target)
```

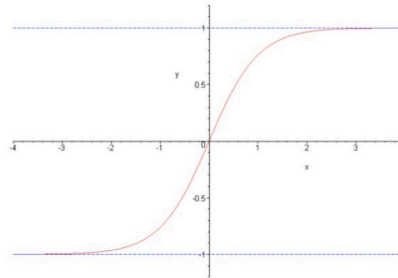
```
4003/4003 [=====] - 12s 3ms/step - loss: 0.7546: 0s -
loss
4003/4003 [=====] - 12s 3ms/step - loss: 0.6809
1/4003 [...] - ETA: 1:00:25 - loss:
0.8878WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0018s vs `on_train_batch_end` time: 0.0032s).
Check your callbacks.
4003/4003 [=====] - 11s 3ms/step - loss: nan
```

### 3. What is the dying neuron problem?



#### 4. What is the vanishing gradient problem?

- This occurs when many layers have very small slopes:
  - e.g., *due to being on at part of the hyperbolic tangent ( $\tanh$ ) curve*
- In deep networks, updates to backpropagation were close to 0.



tanh function

#### 5. Practice question for diagnosing optimization problems:

- Which of the following could prevent a model from showing an improved loss in its first few epochs?
  - ☐ Learning rate too low.
  - ☐ Learning rate too high.
  - ☐ Poor choice of the activation function.
  - ☒ All of the above.

#### 6. Practice exercises for understanding model optimization:

##### ► Package pre-loading:

```
[3]: import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
```

##### ► Data pre-loading:

```
[4]: df = pd.read_csv('ref4. Titanic.csv')

df.replace(False, 0, inplace=True)
df.replace(True, 1, inplace=True)

predictors = df.drop(['survived'], axis=1).to_numpy()
n_cols = predictors.shape[1]
target = to_categorical(df.survived)
input_shape = (n_cols, )
```

► Code pre-loading:

```
[5]: def get_new_model(input_shape=input_shape):  
    model = Sequential()  
    model.add(Dense(100, activation='relu', input_shape=input_shape))  
    model.add(Dense(100, activation='relu'))  
    model.add(Dense(2, activation='softmax'))  
    return (model)
```

► Changing optimization parameters practice:

```
[6]: # Import the SGD optimizer  
from keras.optimizers import SGD  
  
# Create list of learning rates: lr_to_test  
lr_to_test = [.000001, 0.01, 1]  
  
# Loop over learning rates  
for lr in lr_to_test:  
    print('\n\nTesting model with learning rate: %f\n' % lr)  
  
    # Build new model to test, unaffected by previous models  
    model = get_new_model()  
  
    # Create SGD optimizer with specified learning rate: my_optimizer  
    my_optimizer = SGD(lr=lr)  
  
    # Compile the model  
    model.compile(optimizer=my_optimizer, loss='categorical_crossentropy')  
  
    # Fit the model  
    model.fit(predictors, target)
```

Testing model with learning rate: 0.000001

28/28 [=====] - 1s 37ms/step - loss: 6.0089

Testing model with learning rate: 0.010000

28/28 [=====] - 1s 35ms/step - loss: 2.0812

Testing model with learning rate: 1.000000

28/28 [=====] - 1s 37ms/step - loss: 88345450.6148