

Advanced techniques

Practice Session 5

Changho Suh

January 23, 2024

Outline

Will learn how to apply the advanced techniques:

Generalization techniques

regularization, early stopping, dropout

Weight initialization

He's initialization

Techniques for training stability

batch normalization, learning rate decaying

Start with a two-layer DNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

```
model = Sequential()
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
opt = Adam(learning_rate=0.01,
            beta_1 = 0.9,
            beta_2 = 0.999)
```

```
model.compile(optimizer=opt,
               loss='binary_crossentropy',
               metrics=['acc'])
```

Training

```
model.fit(X_train,y_train, epochs=10)
```

Epoch 1/10

814/814 [=====] - 1s 527us/step - loss: 0.3559 - acc: 0.8342

Epoch 2/10

814/814 [=====] - 0s 528us/step - loss: 0.3224 - acc: 0.8507

Epoch 3/10

814/814 [=====] - 0s 526us/step - loss: 0.3198 - acc: 0.8505

Epoch 4/10

814/814 [=====] - 0s 524us/step - loss: 0.3158 - acc: 0.8520

Epoch 5/10

814/814 [=====] - 0s 519us/step - loss: 0.3155 - acc: 0.8516

Epoch 6/10

814/814 [=====] - 0s 523us/step - loss: 0.3134 - acc: 0.8545

Epoch 7/10

814/814 [=====] - 0s 522us/step - loss: 0.3111 - acc: 0.8567

Epoch 8/10

814/814 [=====] - 0s 527us/step - loss: 0.3102 - acc: 0.8555

Epoch 9/10

814/814 [=====] - 0s 522us/step - loss: 0.3081 - acc: 0.8576

Epoch 10/10

814/814 [=====] - 0s 544us/step - loss: 0.3043 - acc: 0.8594

Evaluation

```
val_hist = model.evaluate(X_val, y_val)\n\nprint(val_hist)
```

```
102/102 [=====] - 0s 559us/step - loss: 0.3249 - acc:\n0.8566\n[0.32488954067230225, 0.8565725088119507]
```

Regularization

```
from tensorflow.keras.regularizers import l2
```

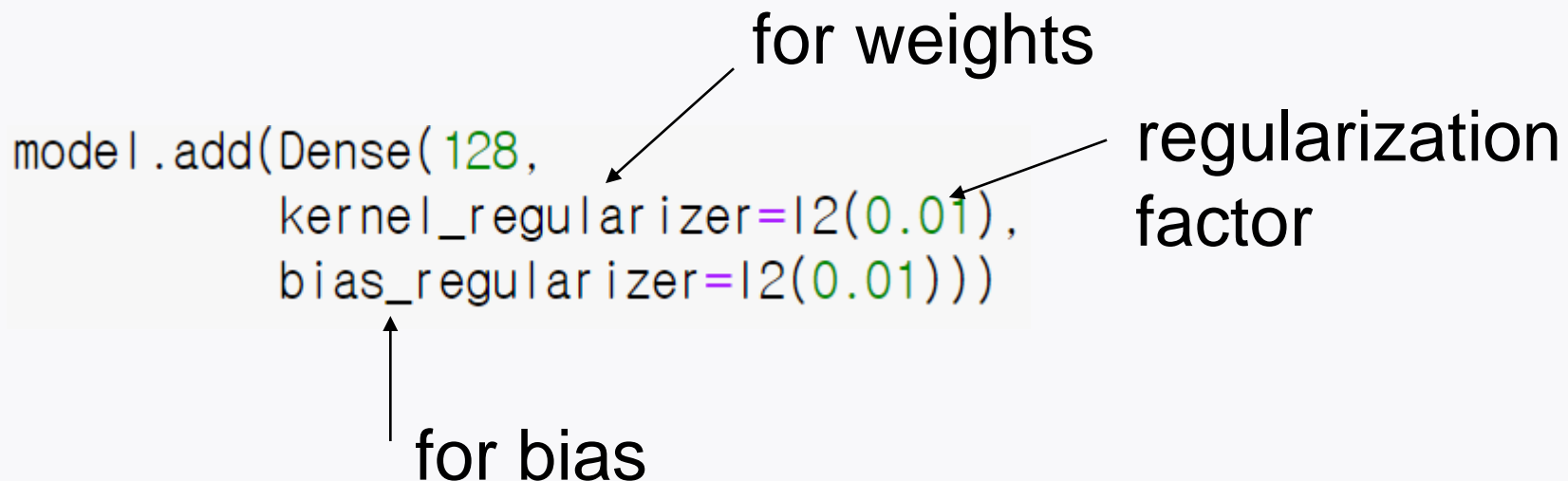
```
model = Sequential()
```

```
model.add(Dense(128,  
                kernel_regularizer=l2(0.01),  
                bias_regularizer=l2(0.01)))
```

for weights

regularization factor

for bias



Early stopping

```

from tensorflow.keras.callbacks import EarlyStopping

model = Sequential()
model.add(Dense(128, kernel_regularizer=l2(0.01),
               bias_regularizer=l2(0.01),
               activation='relu'))
model.add(Dense(1, kernel_regularizer=l2(0.01),
               bias_regularizer=l2(0.01),
               activation='sigmoid'))
opt = Adam(learning_rate=0.01, beta_1 = 0.9, beta_2 = 0.999)
model.compile(optimizer=opt,
              loss='binary_crossentropy')
es_callback = EarlyStopping(monitor='val_loss', patience=2)
hist = model.fit(X_train, y_train,
                 validation_data=(X_val, y_val),
                 epochs=100, callbacks=[es_callback])

```

of epochs with no improvement after which training will stop.

↓

Early stopping: patience=2

```
Epoch 1/100
814/814 [=====] - 1s 659us/step - loss: 0.4573 - val_loss: 0.4273
Epoch 2/100
814/814 [=====] - 0s 590us/step - loss: 0.4204 - val_loss: 0.4095
Epoch 3/100
814/814 [=====] - 0s 586us/step - loss: 0.4115 - val_loss: 0.3940
Epoch 4/100
814/814 [=====] - 0s 582us/step - loss: 0.4028 - val_loss: 0.4456
Epoch 5/100
814/814 [=====] - 0s 586us/step - loss: 0.3998 - val_loss: 0.3956
```

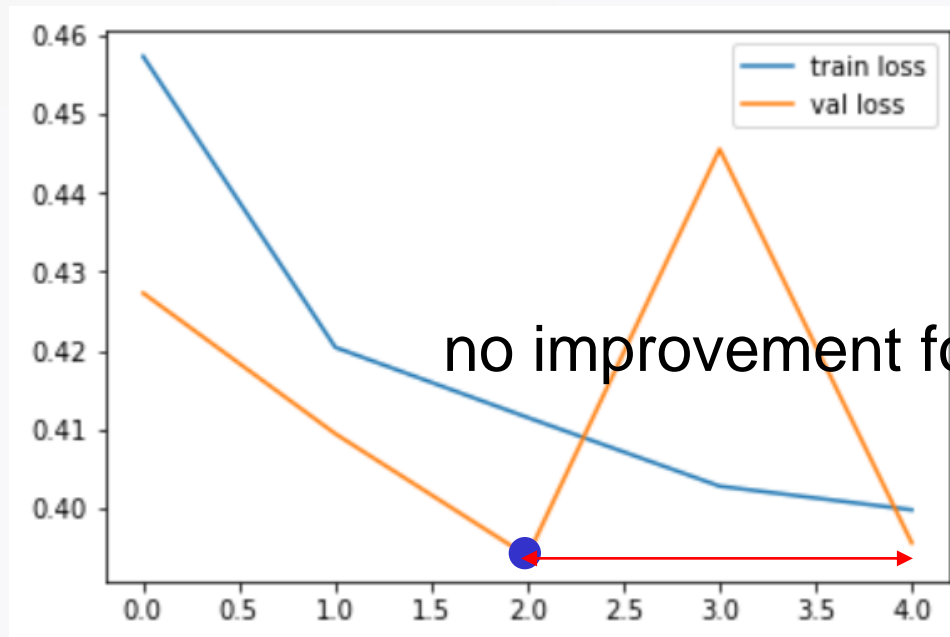
no improvement for 2 epochs

Early stopping: patience=2

```
train_loss = hist.history['loss']  
val_loss = hist.history['val_loss']
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(train_loss, label='train loss')  
plt.plot(val_loss, label='val loss')  
plt.legend()  
plt.show()
```



Early stopping: patience=15

```

Epoch 1/100
814/814 [=====] - 1s 672us/step - loss: 0.4479 - val_loss: 0.4187
Epoch 2/100
814/814 [=====] - 0s 600us/step - loss: 0.4099 - val_loss: 0.3990
Epoch 3/100
814/814 [=====] - 0s 573us/step - loss: 0.4047 - val_loss: 0.3962
Epoch 4/100
814/814 [=====] - 0s 593us/step - loss: 0.4034 - val_loss: 0.3978
Epoch 5/100
814/814 [=====] - 0s 577us/step - loss: 0.4031 - val_loss: 0.4119
Epoch 6/100
814/814 [=====] - 0s 604us/step - loss: 0.3965 - val_loss: 0.4034
Epoch 7/100
814/814 [=====] - 1s 679us/step - loss: 0.3961 - val_loss: 0.3951
Epoch 8/100
814/814 [=====] - 1s 890us/step - loss: 0.3987 - val_loss: 0.3963
Epoch 9/100
814/814 [=====] - 1s 671us/step - loss: 0.3964 - val_loss: 0.3857
Epoch 10/100
814/814 [=====] - 1s 652us/step - loss: 0.3967 - val_loss: 0.3919
Epoch 11/100
814/814 [=====] - 0s 587us/step - loss: 0.4005 - val_loss: 0.4409
Epoch 12/100
814/814 [=====] - 1s 711us/step - loss: 0.3986 - val_loss: 0.3909
Epoch 13/100
814/814 [=====] - 1s 616us/step - loss: 0.4018 - val_loss: 0.3870
Epoch 14/100
814/814 [=====] - 1s 653us/step - loss: 0.3980 - val_loss: 0.3882
Epoch 15/100
814/814 [=====] - 0s 578us/step - loss: 0.3986 - val_loss: 0.3857
Epoch 16/100
814/814 [=====] - 0s 593us/step - loss: 0.3982 - val_loss: 0.3855
Epoch 17/100
814/814 [=====] - 0s 590us/step - loss: 0.3973 - val_loss: 0.3941
Epoch 18/100
814/814 [=====] - 0s 589us/step - loss: 0.3989 - val_loss: 0.3909
Epoch 19/100
814/814 [=====] - 0s 576us/step - loss: 0.3985 - val_loss: 0.4083
Epoch 20/100
814/814 [=====] - 0s 584us/step - loss: 0.3987 - val_loss: 0.3938
Epoch 21/100
814/814 [=====] - 0s 592us/step - loss: 0.3998 - val_loss: 0.3907
Epoch 22/100
814/814 [=====] - 0s 593us/step - loss: 0.3977 - val_loss: 0.3935
Epoch 23/100
814/814 [=====] - 0s 580us/step - loss: 0.3966 - val_loss: 0.4174
Epoch 24/100
814/814 [=====] - 0s 586us/step - loss: 0.3982 - val_loss: 0.3892

```

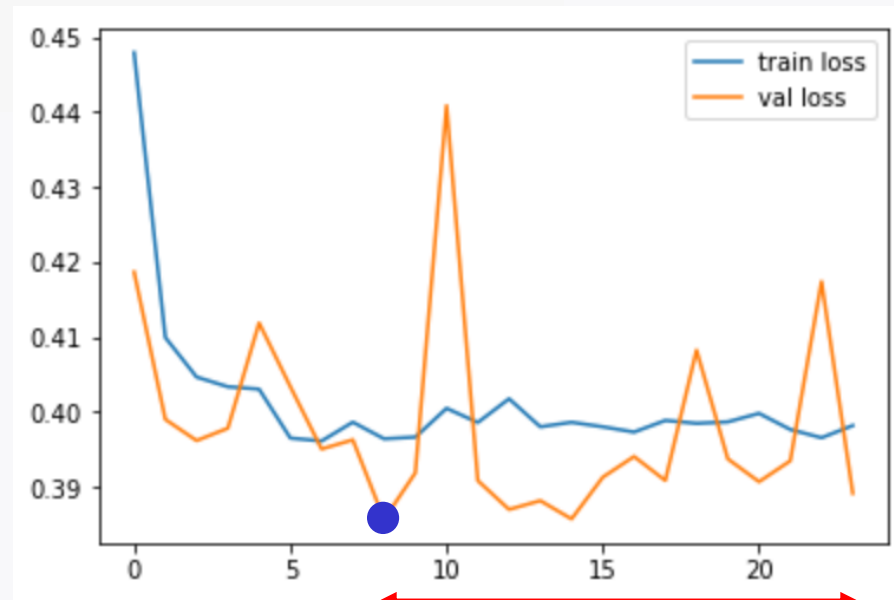
no improvement for 15 epochs

Early stopping: patience=15

```
train_loss = hist.history['loss']  
val_loss = hist.history['val_loss']
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(train_loss, label='train loss')  
plt.plot(val_loss, label='val loss')  
plt.legend()  
plt.show()
```



no improvement for 15 epochs

Dropout

```
from tensorflow.keras.layers import Dropout
```

```
model = Sequential()  
model.add(Dense(128, kernel_regularizer=l2(0.01),  
                bias_regularizer=l2(0.01),  
                activation='relu'))
```

```
model.add(Dropout(0.9))
```

↑
dropout rate

Weight initialization

```
from tensorflow.keras.initializers import HeNormal
```

```
init = HeNormal()
```

```
model.add(Dense(128, kernel_regularizer=l2(0.01),  
                bias_regularizer=l2(0.01),  
                kernel_initializer=init,  
                activation='relu'))
```

initializer setup



Batch normalization

```
from tensorflow.keras.layers import BatchNormalization
```

```
from tensorflow.keras.layers import ReLU
```

```
init = HeNormal()
```

```
model.add(Dense(128, kernel_regularizer=l2(0.01),  
                bias_regularizer=l2(0.01),  
                kernel_initializer=init))
```

```
model.add(BatchNormalization())
```

 ← prior to activation

```
model.add(ReLU())
```

Learning rate decaying

```
from tensorflow.keras.callbacks import LearningRateScheduler
```

```
def scheduler(epoch, lr):  
    if epoch in [20,40,60]:  
        lr = 0.1*lr  
    else:  
        lr = lr  
    return lr
```

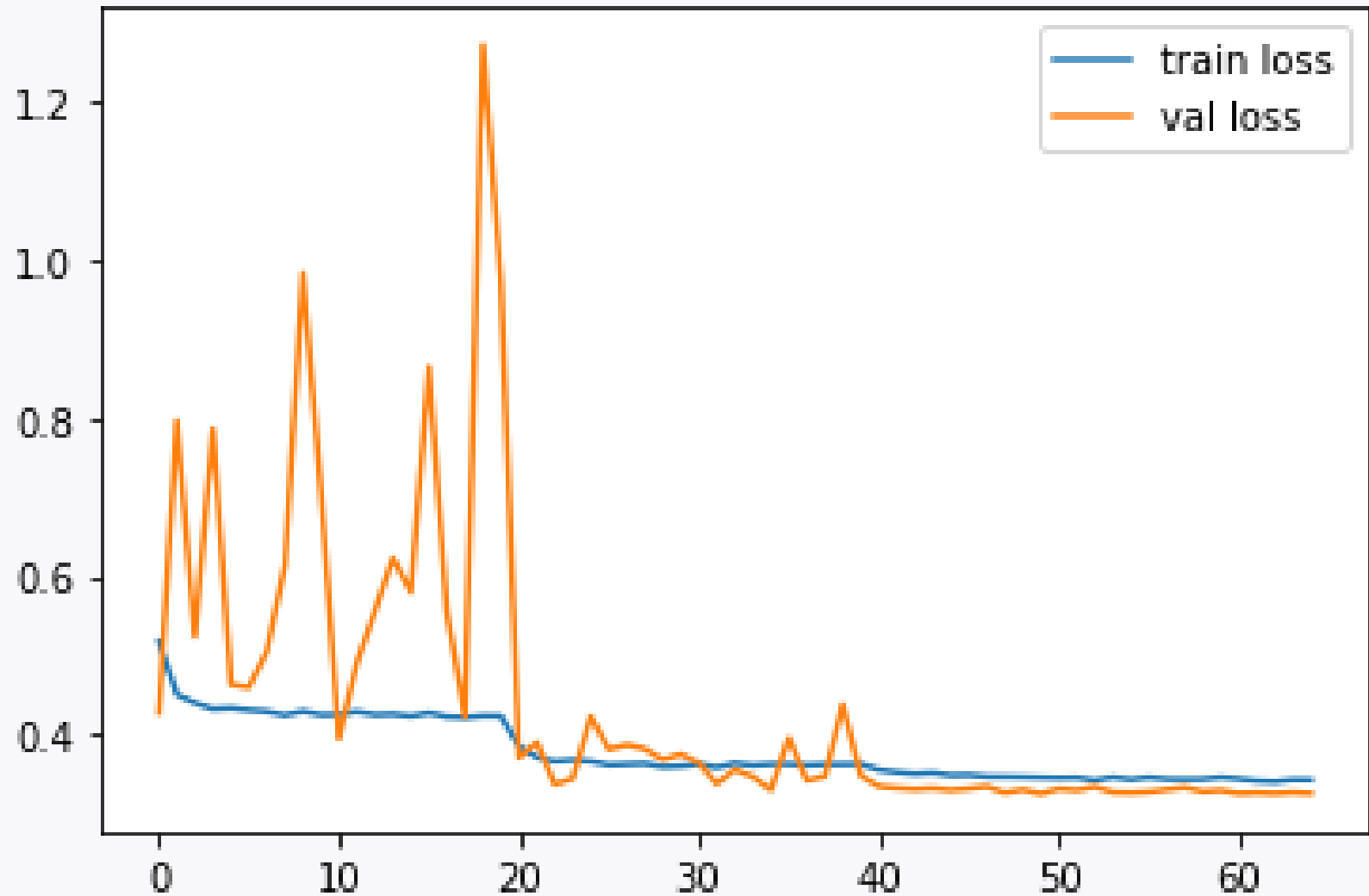
```
ls_callback = LearningRateScheduler(scheduler)
```

```
hist = model.fit(X_train, y_train,  
                 validation_data=(X_val, y_val),  
                 epochs=100, callbacks=[es_callback, ls_callback])
```

position



Learning rate decaying



Look ahead

Will learn how to implement **cross validation**.