CSCE 636 Neural Networks (Deep Learning)

Lecture 6: Fundamentals of Machine Learning

Anxiao (Andrew) Jiang

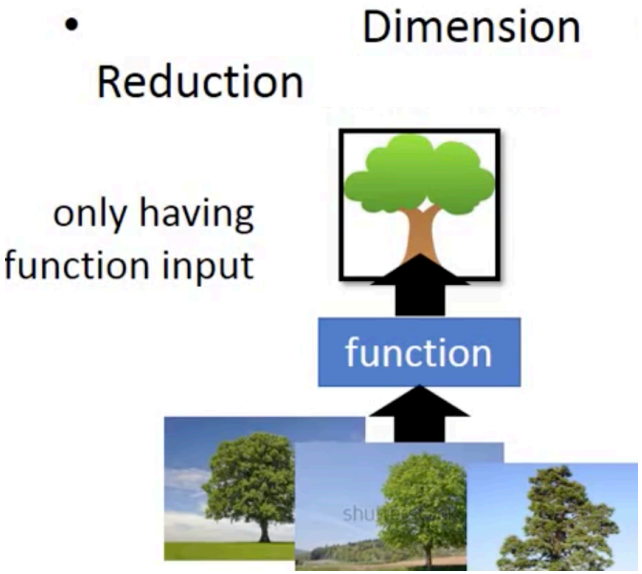
Supervised Learning

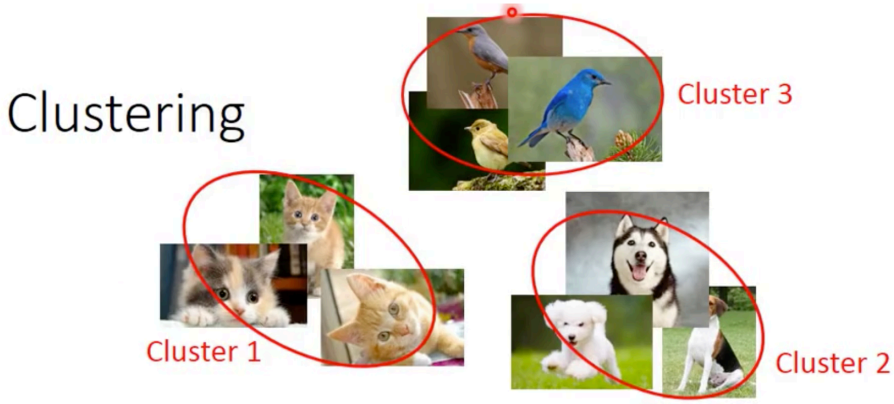
- Input and output are both known. Just learn the function.

- The four applications introduced so far in our class are all supervised learning.

Unsupervised Learning

- Output is unknown. Learn the relationship between data.

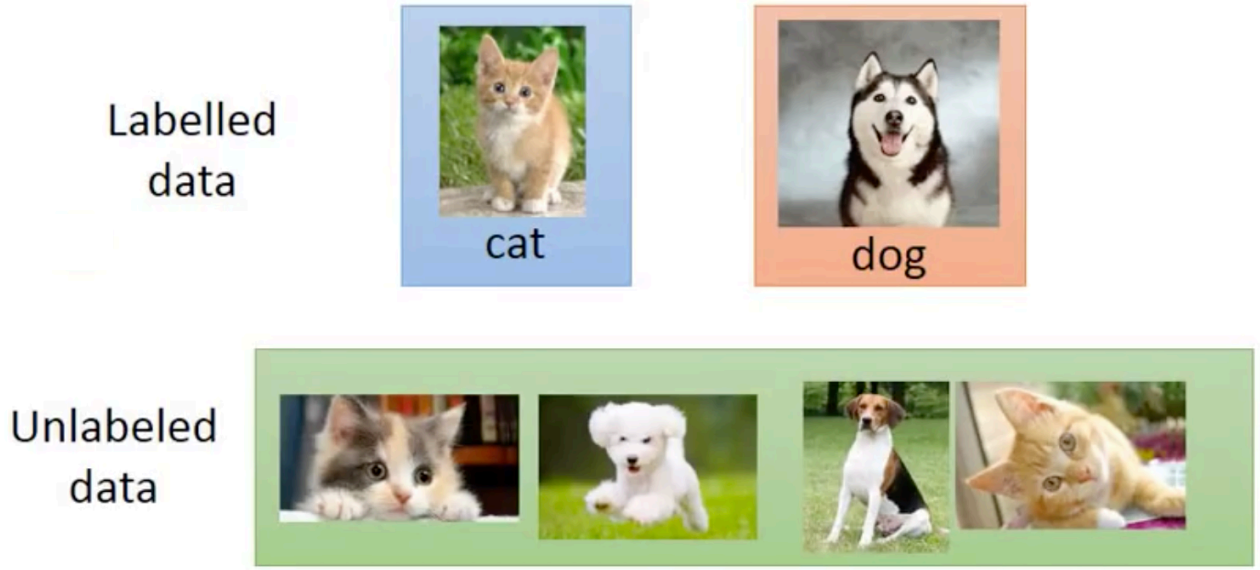




Figures by Prof. Hung-yi Lee.

Semi-supervised Learning

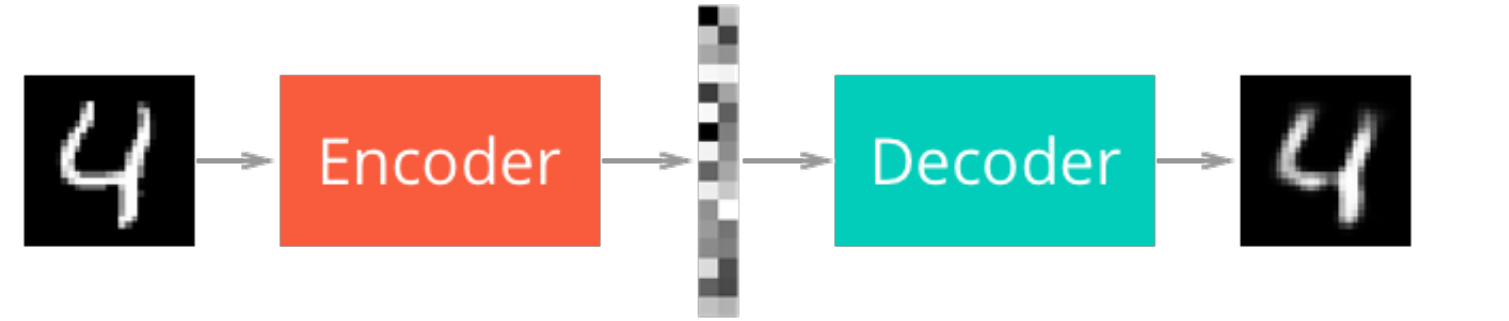
- Some outputs are known, but not all. (Most data are unlabeled.)



Figures by Prof. Hung-yi Lee.

Self-supervised Learning

- Output is generated from input data, without human help.
- Example: auto-encoder



Reinforcement Learning

- Learn from feedback (penalty or reward) from environment.
- But the environment does not tell what to do.

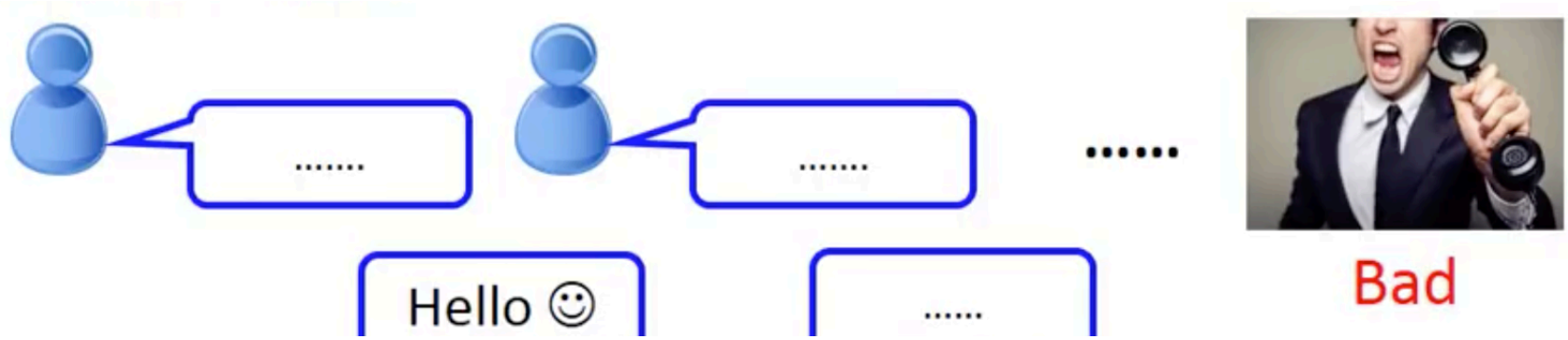


Figure by Prof. Hung-yi Lee.

Underfitting: training performance is bad

- This is the first thing we should worry about.

Overfitting: training performance is good, but test performance is bad

- This is the next thing we should worry about.

To prevent underfitting: change our model or make it general (to include the correct function in our solution set). Then train our NN well to find that correct function.

To prevent overfitting: after we have found a NN with good training performance, simplify the NN model (e.g., reduce NN size, or use regularization techniques). Or get more data.

Regularization techniques

- Weight regularization: add a function of weights to the loss function, to prevent the weights from becoming too large.

L2 regularization $\quad \text{new loss function} = \text{old loss function} + \lambda \sum_i w_i^2$

L1 regularization $\quad \text{new loss function} = \text{old loss function} + \lambda \sum_i |w_i|$

A reason for weight regularization: large weight can make the model more sensitive to noise/variance in data.

L2 regularization: it tends to make all weights small.

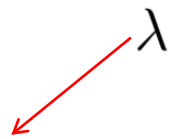
L1 regularization: it tends to make weights sparser (namely, more 0s).

L2 regularization: example

Listing 4.6 Adding L2 weight regularization to the model

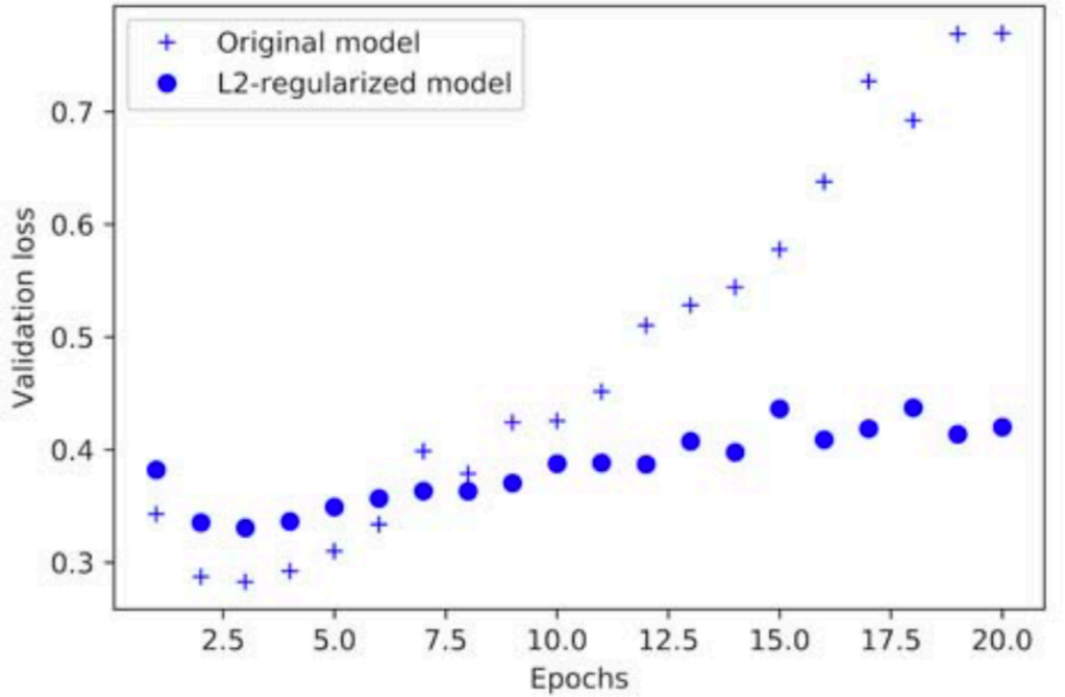
```
from keras import regularizers

model = models.Sequential()
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```



The above example uses L2 regularization on first layer and second layer, but not third layer.

L2 regularization: example



More resistant to overfitting

Figure 4.7 Effect of L2 weight regularization on validation loss

Note: with weight regularization, the loss will become higher during training (due to the extra term in loss function). In testing, the extra term is not used, so loss will get lower.

L1 regularization, and combine L1 and L2

Listing 4.7 Different weight regularizers available in Keras

```
from keras import regularizers
regularizers.l1(0.001)
regularizers.l1_l2(l1=0.001, l2=0.001)
```

L1 regularization

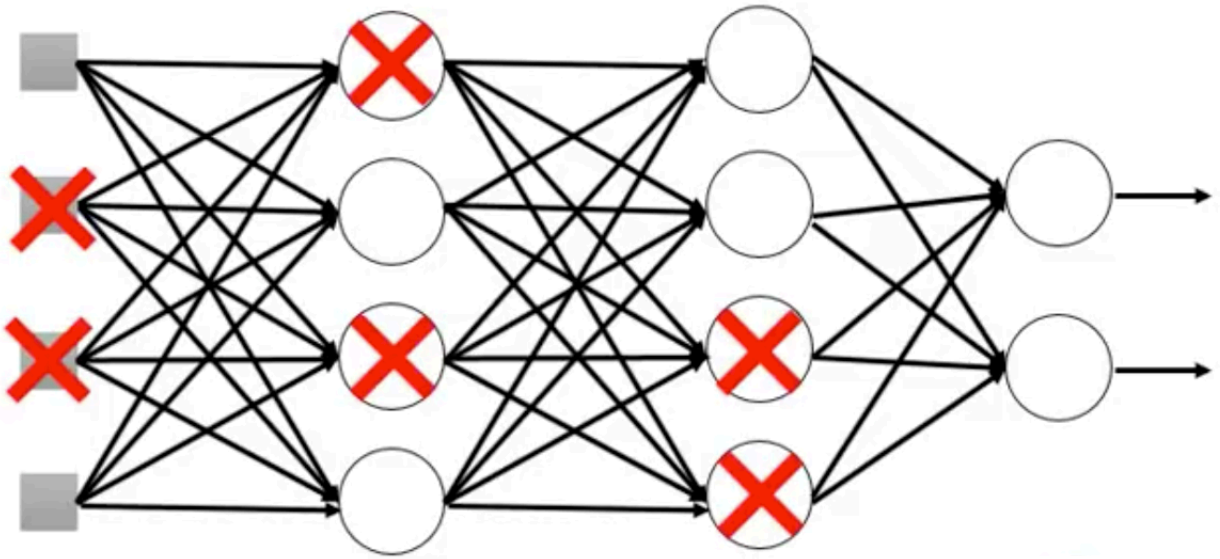
Simultaneous L1 and L2 regularization

Another regularization technique: Dropout

The following slides are based on Prof. Hung-yi Lee's interesting lecture on machine learning:
"Tips for Training DNN"
https://www.youtube.com/watch?v=xki61j7z-30&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=16

Dropout

Training:

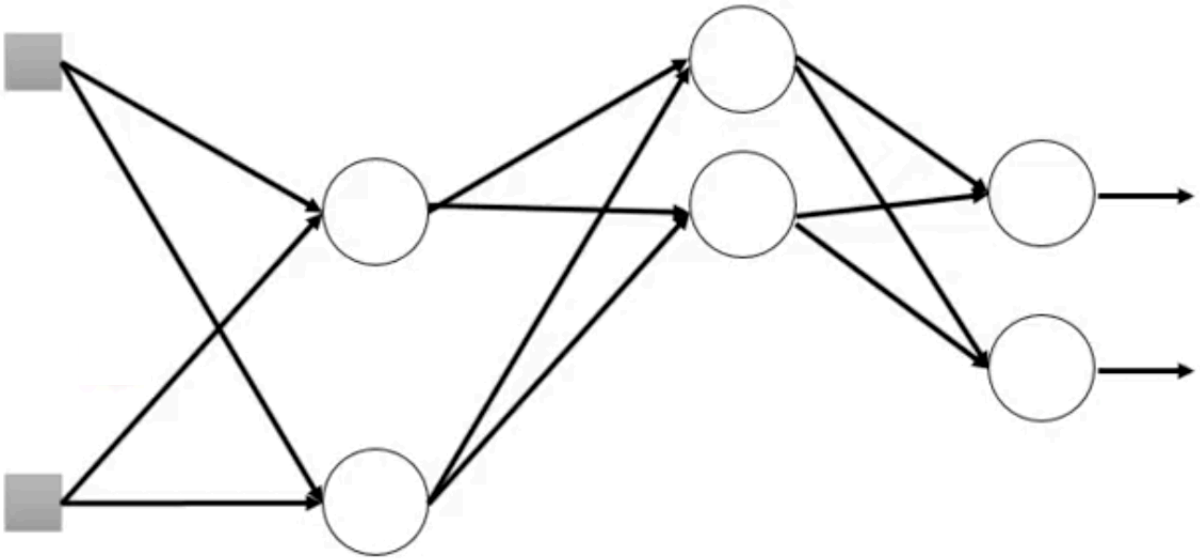


- **Each time before updating the parameters**
  - Each neuron has p% to dropout

When a node drops out, its output and its outgoing edges are all removed for this weight update.

Dropout

Training:



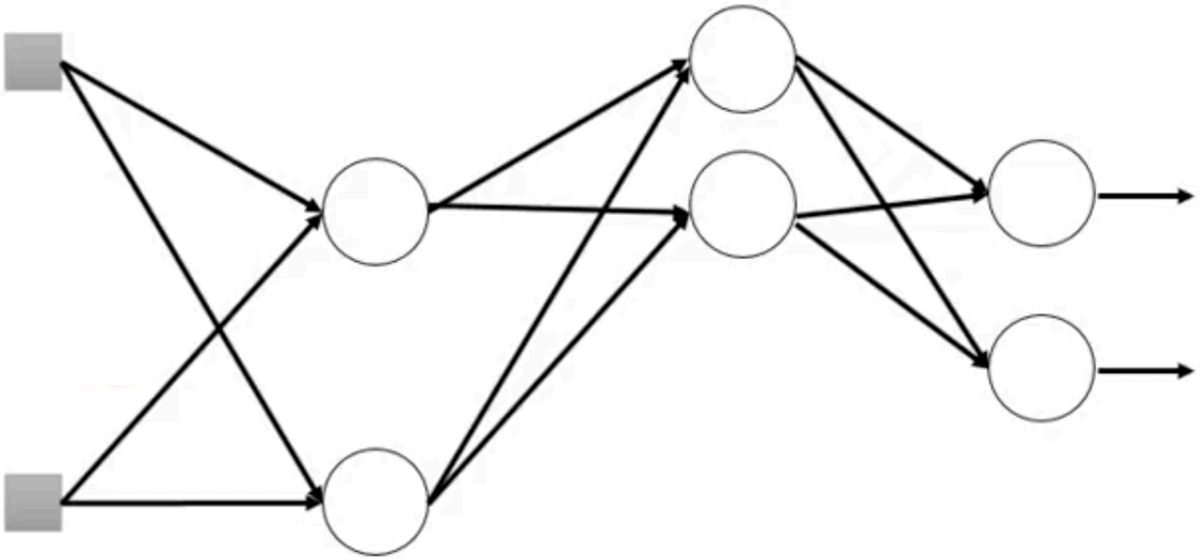
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    → **The structure of the network is changed.**

Use the new network to train its weights, for this mini-batch of data.

Dropout

Training:



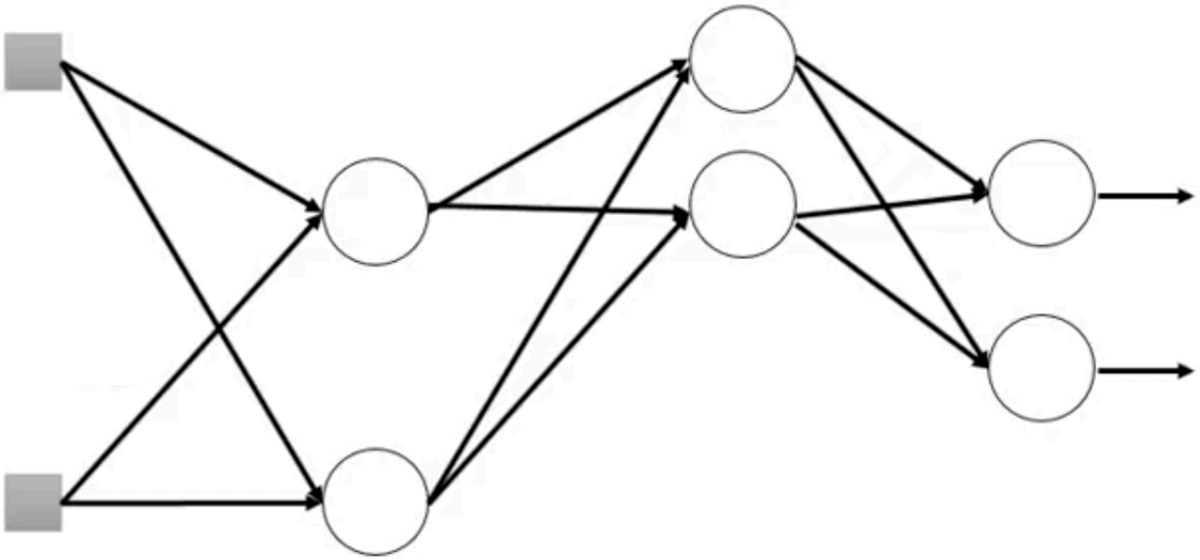
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    → **The structure of the network is changed.**

For each mini-batch, we re-sample the nodes to drop out.

Dropout

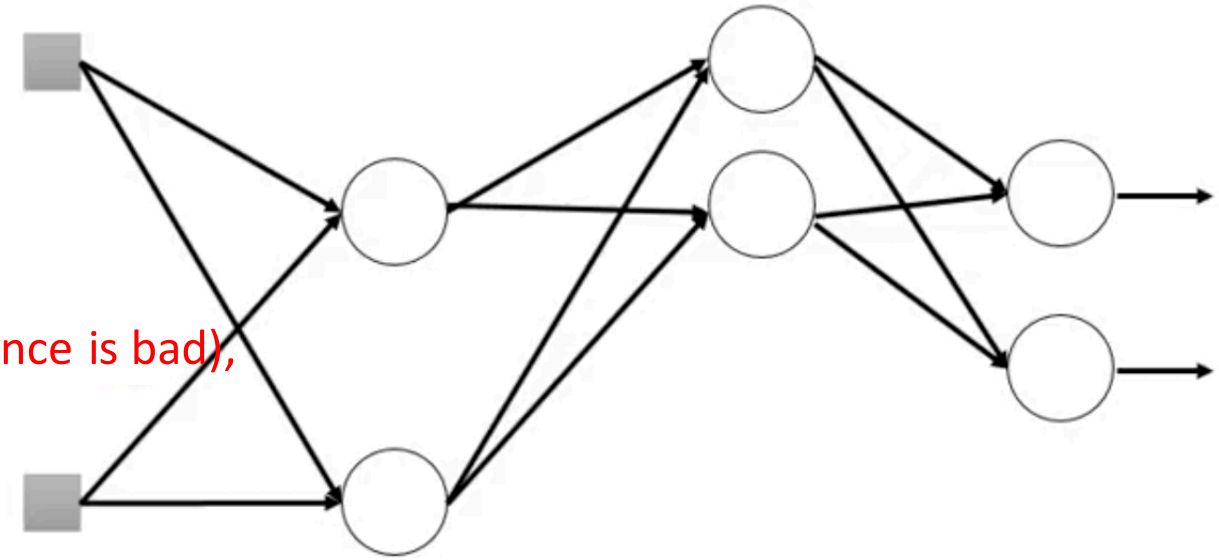
Training:



With dropout, training performance will worsen.
(But we hope the test performance will become better due to less overfitting).

Dropout

Training:

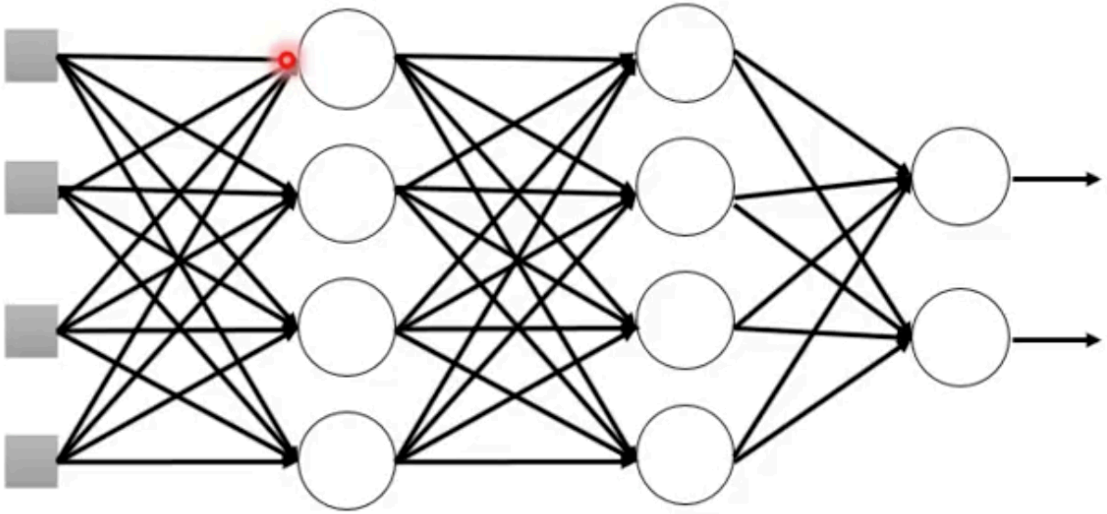


If the NN is underfitting
(namely, training performance is bad),
don't use dropout.

With dropout, training performance will worsen.
(But we hope the test performance will become better due to less overfitting).

Dropout

Testing:



- **No dropout**
  - If the dropout rate at training is p%, all the weights times (1-p)%

Keras will take care of it automatically for us. So don't worry about it.

Listing 4.8 Adding dropout to the IMDB network

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

dropout rate is 0.5

Note: Dropout is considered a layer in Keras.
Dropout applies to the layer right before it.
The dropout rate is usually between 0.2 and 0.5.

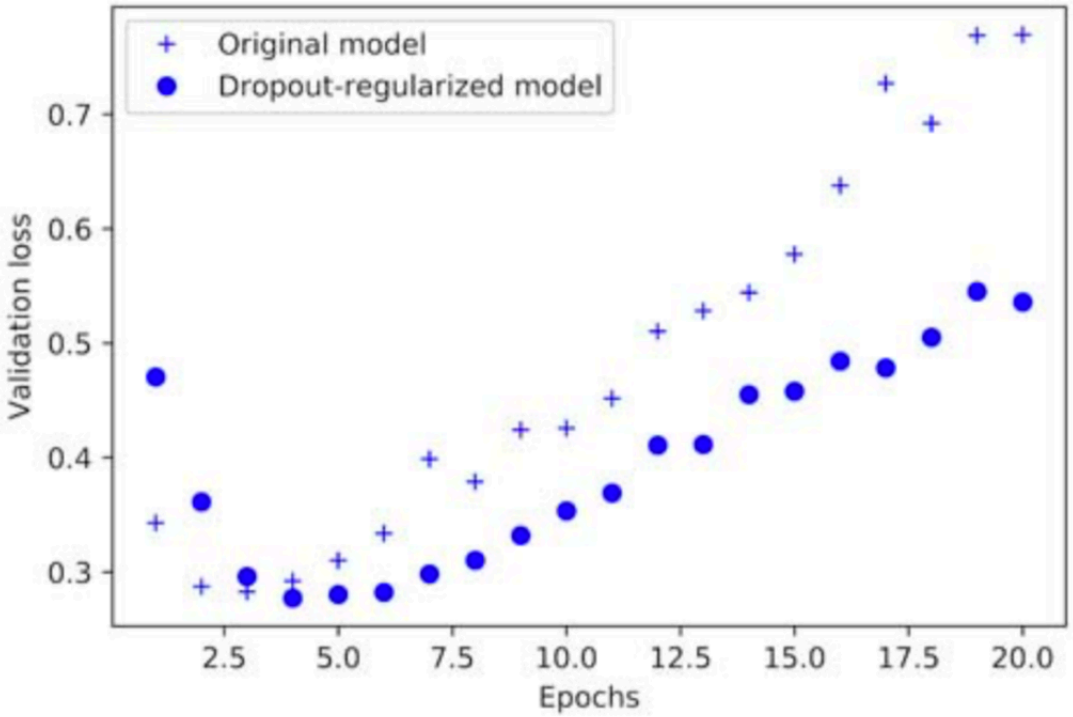


Figure 4.9 Effect of dropout on validation loss

In the above figure, the model with dropout is more resistant to overfitting.
Remember that dropout makes training performance worse.
So its test performance will likely be better than training without dropout.

Intuitions for dropout

- Weights have less chance of “collusion” for overfitting.

- Each weight “trains harder” to capture a feature, since other weights may dropout during training. (If my teammates do not work hard, then I have to work harder.)