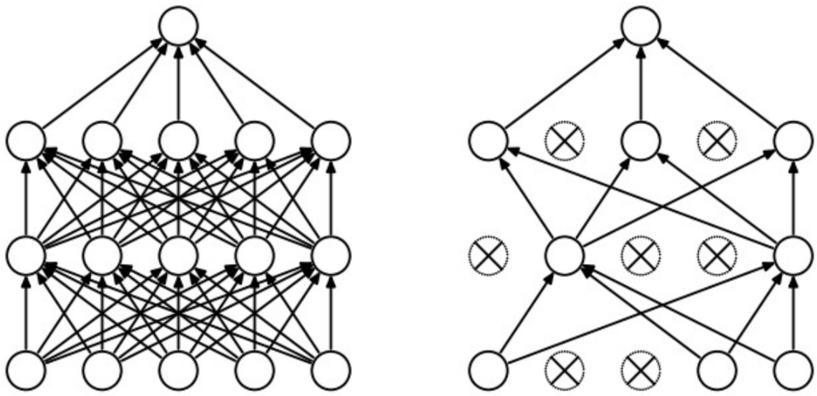
Dropout是一种简单有效的正则化方式
在**训练**的时候，随机失活的实现方法是让神经元以超参数p的概率被设置为0。



(a) Standard Neural Net (b) After applying dropout.

在训练过程中，随机失活可以被认为是对完整神经网络抽样出一些子集，每次基于输入数据值只更新子网络的参数（然而，数量巨大的子网络们并不是相互独立的，因为它们都共享参数）。
在测试（或者验证）时候不使用随机失活，可以理解为对数量巨大的子网络们做了模型集成，以此来计算出一个平均的预测。

理解了dropout的基本原理，下面我们来实现下dropout层。
主要内容如下：

1 Dropout层

1.1 前向传播

代码如下：

```
def dropout_forward(x, dropout_param):

    p, mode = dropout_param['p'], dropout_param['mode']
    if 'seed' in dropout_param:
```

```
        np.random.seed(dropout_param['seed'])

    mask = None
    out = None

    if mode == 'train':

 ########################################################################
###
        # TODO: Implement the training phase forward pass for inverted
dropout.   #
        # Store the dropout mask in the mask variable.
         #

 ########################################################################
###
        mask = (np.random.rand(*x.shape) >= p) / (1 - p)
        # mask = (np.random.rand(x.shape[1]) >= p) / (1 - p)
        out = x * mask
        # pass

 ########################################################################
###
        #                            END OF YOUR CODE
        #

 ########################################################################
###
    elif mode == 'test':

 ########################################################################
###
        # TODO: Implement the test phase forward pass for inverted
dropout.       #

 ########################################################################
###
        out = x
        # pass

 ########################################################################
###
        #                            END OF YOUR CODE
        #
```

```
    ###########################################################################
    
    cache = (dropout_param, mask)
    out = out.astype(x.dtype, copy=False)

    return out, cache
```

我们来检验下，检验代码如下：

```
x = np.random.randn(500, 500) + 10

for p in [0.3, 0.6, 0.75]:
  out, _ = dropout_forward(x, {'mode': 'train', 'p': p})
  out_test, _ = dropout_forward(x, {'mode': 'test', 'p': p})

  print ('Running tests with p = ', p)
  print ('Mean of input: ', x.mean())
  print ('Mean of train-time output: ', out.mean())
  print ('Mean of test-time output: ', out_test.mean())
  print ('Fraction of train-time output set to zero: ', (out == 0).mean())
  print ('Fraction of test-time output set to zero: ', (out_test == 0).mean())
  print()
```

输出结果如下：

```
Running tests with p = 0.3
Mean of input: 9.99902471947
Mean of train-time output: 9.98399347022
Mean of test-time output: 9.99902471947
Fraction of train-time output set to zero: 0.301064
Fraction of test-time output set to zero: 0.0

Running tests with p = 0.6
Mean of input: 9.99902471947
Mean of train-time output: 10.00544314
Mean of test-time output: 9.99902471947
Fraction of train-time output set to zero: 0.599708
Fraction of test-time output set to zero: 0.0

```

```
Running tests with p = 0.75
Mean of input: 9.99902471947
Mean of train-time output: 10.037004918
Mean of test-time output: 9.99902471947
Fraction of train-time output set to zero: 0.749104
Fraction of test-time output set to zero: 0.0
```

我们来对结果分析下，当`p=0.3`时，在训练过程中，我们的神经元有0.3的概率失活，也就是被置0，我们可以看到神经元为0的概率为0.301064，约等于0.3；而在测试过程中，将所有的神经元集成，也就是使用所有的神经元，所以神经元为0的概率为0。

1.2 反向传播

下面我们来看下dropout层的反向传播，代码如下：

```
def dropout_backward(dout, cache):

    dropout_param, mask = cache
    mode = dropout_param['mode']

    dx = None
    if mode == 'train':
        ###########################################################################
        # TODO: Implement the training phase backward pass for inverted dropout.  #
        ###########################################################################
        dx = dout * mask
        # pass
        ###########################################################################
        #                            END OF YOUR CODE                             #
        ###########################################################################
    elif mode == 'test':
        dx = dout
    return dx
```

同样，我们对梯度进行检验。
代码如下：

```
x = np.random.randn(10, 10) + 10
dout = np.random.randn(*x.shape)

dropout_param = {'mode': 'train', 'p': 0.8, 'seed': 123}
out, cache = dropout_forward(x, dropout_param)
dx = dropout_backward(dout, cache)
dx_num = eval_numerical_gradient_array(lambda xx: dropout_forward(xx, dropout_param)[0], x, dout)

print ('dx relative error: ', rel_error(dx, dx_num))
```

得到结果如下：

```
dx relative error: 1.89290485329e-11
```

结果是1e-11，误差还是很小的，不错。

2 训练

完成了对dropout层的设计，下面我们就使用下dropout层来对cifar10数据进行训练，来看下dropout层的效果。

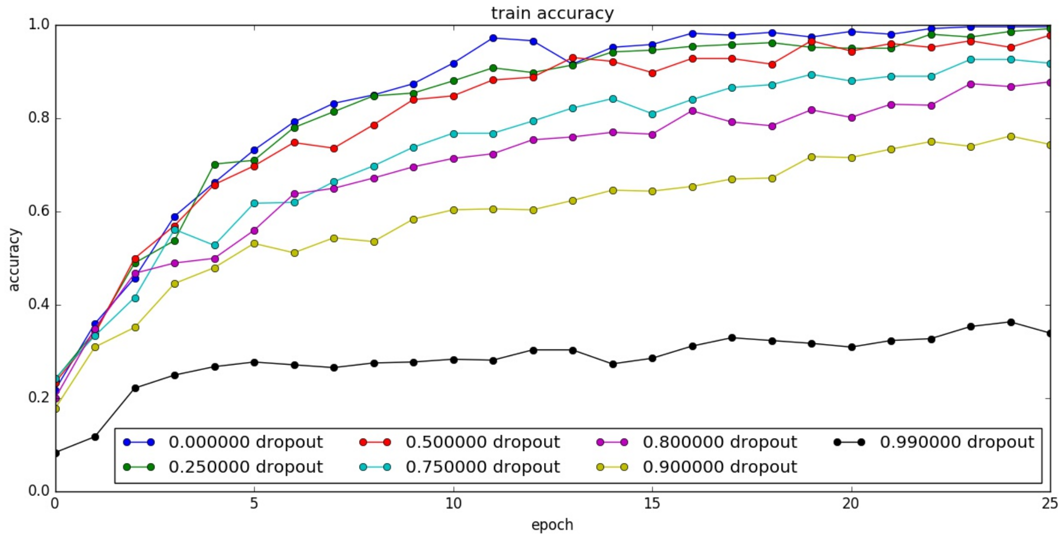
2.1 开始训练

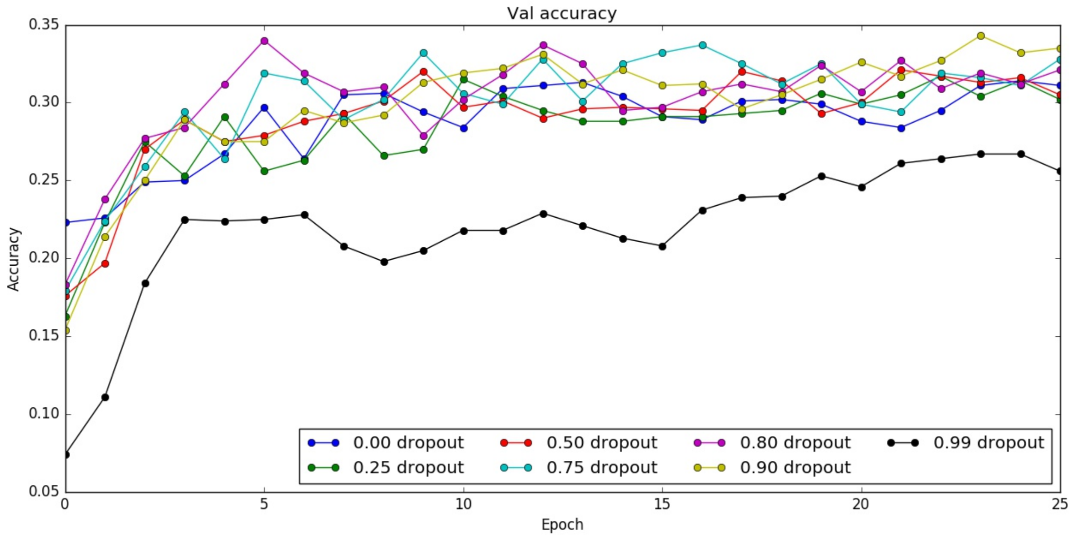
为了加快训练，这里我们只使用两层网络，即一个隐层。
代码如下：

```
num_train=500

small_data={
    'X_train':data['X_train'][:num_train],
    'y_train':data['y_train'][:num_train],
    'X_val':data['X_val'],
    'y_val':data['y_val']

}
```

```python

solvers={}
dropout_choices=[0,0.25,0.5,0.75,0.8,0.9,0.99]
for dropout in dropout_choices:
    model=FullyConnectedNet([500],dropout=dropout)
    print('dropout is:',dropout)
    solver=Solver(model,small_data,num_epochs=25,batch_size=100,update_rule='adam',
                  optim_config={'learning_rate':5e-4},verbose=True,print_every=100)
    solver.train()
    solvers[dropout]=solver

train_acc=[]
val_acc=[]
for dropout in dropout_choices:
    solver=solvers[dropout]
    train_acc.append(solver.train_acc_history[-1])
    val_acc.append(solver.val_acc_history[-1])

plt.subplot(311)
for dropout in dropout_choices:
    plt.plot(solvers[dropout].train_acc_history,'-o',label='%2f dropout' % dropout)
plt.title('train accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(ncol=4,loc='lower right')

plt.subplot(312)
for dropout in dropout_choices:
  plt.plot(solvers[dropout].val_acc_history, 'o', label='%.2f dropout' % dropout)
plt.title('Val accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(ncol=4, loc='lower right')

plt.gcf().set_size_inches(15, 15)
plt.show()
```

得到的可视化图如下：





2.2 结果分析

增加Dropout层可以防止过拟合，降低训练集和验证集之间的差距。但是随着dropout中p值的增加，模型的拟合能力也就变差，也就是降低了神经网络的容量，可能出现欠拟合。因此在实践中一般要谨慎选择p值。

同时，学习了BN层的读者，可能已经想到，在网络中增加BN层，可以减少对dropout层的依赖。