**Regularization**

Welcome to the second assignment of this week. Deep Learning models have so much flexibility and capacity that **overfitting can be a serious problem**, if the training dataset is not big enough. Sure it does well on the training set, but the learned network **doesn't generalize to new examples** that it has never seen!

**Observations**:

* The value of λλ is a hyperparameter that you can tune using a dev set.
* L2 regularization makes your decision boundary smoother. If λλ is too large, it is also possible to "oversmooth", resulting in a model with high bias.

**What is L2-regularization actually doing?**:

L2-regularization relies on the assumption that a model with small weights is simpler than a model with large weights. Thus, by penalizing the square values of the weights in the cost function you drive all the weights to smaller values. It becomes too costly for the cost to have large weights! This leads to a smoother model in which the output changes more slowly as the input changes.

**What you should remember** -- the implications of L2-regularization on:

* The cost computation:
  + A regularization term is added to the cost
* The backpropagation function:
  + There are extra terms in the gradients with respect to weight matrices
* Weights end up smaller ("weight decay"):
  + Weights are pushed to smaller values.

## 3 - Dropout

Finally, **dropout** is a widely used regularization technique that is specific to deep learning. **It randomly shuts down some neurons in each iteration.**

When you shut some neurons down, you actually modify your model. The idea behind drop-out is that at each iteration, you train a different model that uses only a subset of your neurons. With dropout, your neurons thus become less sensitive to the activation of one other specific neuron, because that other neuron might be shut down at any time.

* A **common mistake** when using dropout is to use it both in training and testing. You should use dropout (randomly eliminate nodes) only in training.
* Deep learning frameworks like [tensorflow](https://www.tensorflow.org/api_docs/python/tf/nn/dropout" \t "_blank), [PaddlePaddle](http://doc.paddlepaddle.org/release_doc/0.9.0/doc/ui/api/trainer_config_helpers/attrs.html" \t "_blank), [keras](https://keras.io/layers/core/" \l "dropout" \t "_blank) or [caffe](http://caffe.berkeleyvision.org/tutorial/layers/dropout.html" \t "_blank) come with a dropout layer implementation. Don't stress - you will soon learn some of these frameworks.

**What you should remember about dropout:**

* Dropout is a regularization technique.
* You only use dropout during training. Don't use dropout (randomly eliminate nodes) during test time.
* Apply dropout both during forward and backward propagation.
* During training time, divide each dropout layer by keep\_prob to keep the same expected value for the activations. For example, if keep\_prob is 0.5, then we will on average shut down half the nodes, so the output will be scaled by 0.5 since only the remaining half are contributing to the solution. Dividing by 0.5 is equivalent to multiplying by 2. Hence, the output now has the same expected value. You can check that this works even when keep\_prob is other values than 0.5.

## 4 - Conclusions

**Here are the results of our three models**:

|  |  |  |
| --- | --- | --- |
| **model** | **train accuracy** | **test accuracy** |
| 3-layer NN without regularization | 95% | 91.5% |
| 3-layer NN with L2-regularization | 94% | 93% |
| 3-layer NN with dropout | 93% | 95% |

Note that regularization hurts training set performance! This is because it limits the ability of the network to overfit to the training set. But since it ultimately gives better test accuracy, it is helping your system.

Congratulations for finishing this assignment! And also for revolutionizing French football. :-)

**What we want you to remember from this notebook**:

* Regularization will help you reduce overfitting.
* Regularization will drive your weights to lower values.
* L2 regularization and Dropout are two very effective regularization techniques.