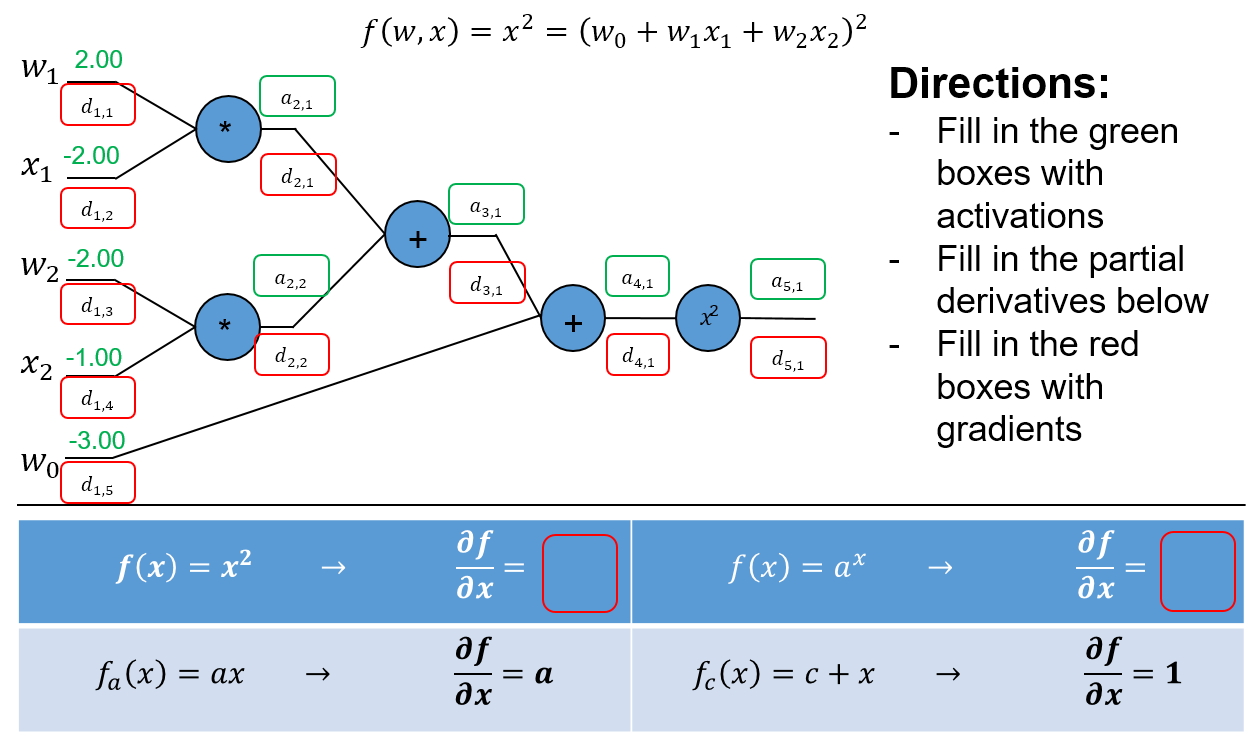
Assignment 1 (Due 4/26/2017)

Understanding Neural Networks

|  |  |
| --- | --- |
| **Name** |  |
| **Discussion partner** |  |
| **Comments** | Add comments for the grader here. E.g. How to run the code, or anything to note when grading the code. |
| **Feedback** | Note any feedback that you’d like to address in a future lecture. |

# 1. Neural Networks on paper (5 points)

Fill in the blanks below:



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | |  | |  | |
|  |  |  |  |  |  |
|  |  |  | |  |  |

# 2. Neural Networks in code (12 points)

Using the provided example code from Lecture 3, explore the items below and demonstrate how to improve the example code by showing plots of the loss and accuracy curves. For each plot, show the curves for the first 200 iterations (You can stop training after 200 iterations for this part of the assignment). Consider the visualizations for activations, weights and weight updates.

1. **Learning rate:** Adjust the learning rate variable (lr) to try to achieve the “fastest” possible training rate. Show your loss and accuracy curves. What should you generally look for in the visualizations to ensure a “good” learning rate?
2. **Activation function:** Try changing the sigmoid function to “1.0/(1.0 + np.e\*\*-(k\*x))”, where k is another training parameter. Explain what k does. What is the effect of a small k on training versus a larger value for k? Is there an optimal k for a given learning rate? Use the default learning rate “1e-5” for your experiments. Justify your position in words, and show up to 5 plots.
3. **Initialization:** In the sample code, the weights W1, W2, W3 are initialized uniformly from -1 to 1. Experiment with various kinds of initialization and report your findings. Justify why your proposed initialization is better than the default initialization. Show up to 5 plots. Hint: how do the visualizations differ for good and bad initializations?

# 3. Optimization in code (16 points)

Using the example code from lecture 3, demonstrate your understanding of the principles in lecture 4 by doing the following (submit your final code for these in part 4 below, but show your changes here):

1. **Activations and Gradients:** Examine the activations and gradients visualized during training. Justify why the mean and standard deviation of the activation and gradient matrices are “optimal” or not. Propose some ways to “fix” the activations/gradients to improve training. Show some plots to illustrate how your proposed “fix” improves training.
2. **Tanh:** Implement tanh(x) instead of the sigmoid. Explain why tanh(x) may be better, and show plots. Hint: what is the derivative of tanh(x)?
3. **Cross Entropy:** Implement cross entropy. Show plots of how “Cross-entropy” improves training.
4. **ReLU:** Implement rectified linear units and justify why they may be better. Show plots. Hint: what is the derivative of relu(x)? Did any of your neurons “die”? What do dead neurons look like in the visualizations? How can we “fix” dead neurons?

# 4. Understanding the weights (7 points)

Looking at the visualizations of the activations, weights and weight updates, explain what each plot means. Refer to the images in the “train” subfolder. Don’t forget to delete or rename old runs.

- How do the visualizations/plots differ for Tanh, ReLU and cross entropy?

- How does the weight/update magnitude change as training progresses? How are the magnitudes similar or different depending on the depth of the layer?

- What are signs that the network is “stuck”, and how should the plots look as the network reaches the final trained state?

- Does the network “prefer” certain activation/weight settings? Or do the activations/weights change with more training? Does this depend on initialization? Why?

# 5. Putting it all together (10 points)

Starting with the example code from lecture 3, integrate all your improvements from part 3 (Tanh, Cross entropy, ReLU and others that you can think of) together to attain the best possible training conditions. Comment your code thoroughly, and show plots of how your code improves upon the example. Explain thoroughly what you did and why it works. Submit your final code, but comment out the lines that you aren’t using, e.g. tanh.

**Extra credit:** Recall the discussion of random labels in class and how neural networks may have enough capacity to remember the whole training set in the weights. Uncomment the lines for random labels. Explain if your model may have enough capacity for overfitting. Suggest ideas for fixing this problem.