Linear Optimization Assignment #1

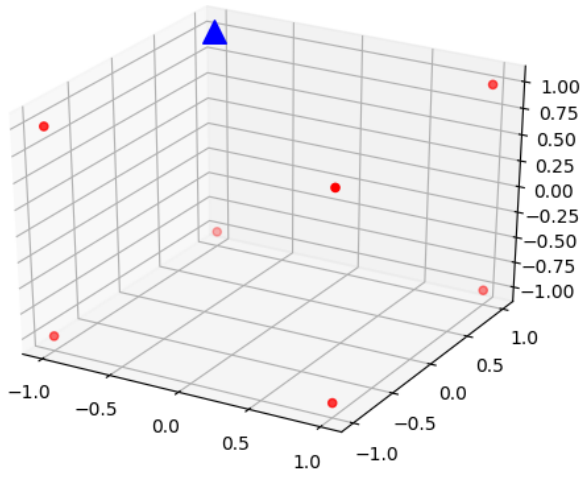
Instruction: Write a report and complete code. Download the code from ftp(10.13.71.168). Upload them to ftp(10.13.72.84).

- Upload:
  - Address: 10.13.72.84
  - Username: opt; Passwd: opt18; Port: 21
- Download:
  - Address: 10.13.71.168
  - Username: opt; Passwd: opt18; Port: 21

Problem 1: MP Neuron

A McCulloch-Pitts (M-P) neuron accepts bipolar input $x_i \in \{-1, 1\}$ and gives $y = \text{sgn}(\sum_{i=1}^{m} w_i x_i + b)$. Give weights and bias for a M-P neuron with inputs $x, y, z \in \{-1, 1\}$ and whose output is z if $x = -1$ and $y = 1$, and is -1 otherwise.

Solution 1.1: MP Neuron



All separating hyperplanes are correct, *e.g.*, $-x + y + z - 2 = 0$, $f(x, y, z) = \text{sgn}(-x + y + z - 2)$.
共 2 分：过程 1 分，答案 1 分

Problem 2: Gradient Descent

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima. Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations as can be seen in fig 1.

$$v_t = \mu v_{t-1} + \varepsilon \nabla_u f(u) \quad (1)$$
$$u_{t+1} = u_t - v_t \quad (2)$$

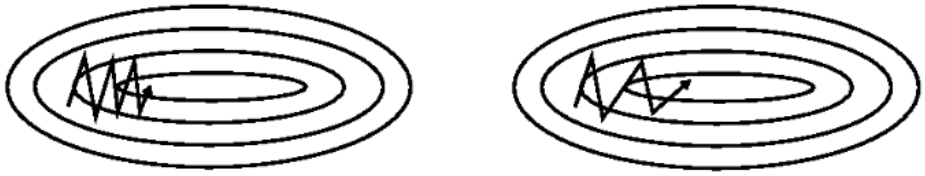


Figure 1: Left: SGD without momentum; Right: SGD with momentum

a). Execute four iterations of gradient descent with momentum to find the minimum of the function $f(u) = \frac{u^3}{3} + 50u^2 - 100u - 30$. Start with $u_0 = 20$ and $v_0 = 0$, use a learning rate that is set to $\varepsilon = 0.01$, and parameter μ set to 0.1.

b). Evaluate the benefit of using a momentum for the task in a). by comparing your findings to the vanilla gradient descent method.

Solution 2.1: Gradient Descent

Strictly follow formula 1 and 2. 公式中上面是 + 下面是-，尽管下面-上面 + 也是带动量的 SGD，但是希望能按照题目的公式。

With momentum:

- $u_1 = 20, v_1 = 23$
- $u_2 = -3, v_1 = -1.61$
- $u_3 = -1.39, v_3 \approx -2.53$
- $u_4 = 1.14, v_4 = -0.099$

最终 $f(x) \approx -76.48, u_t \approx 1.24$. (1 pnt)

Vanilla gradient descent:

- $u_1 = 20, v_1 = 23$
- $u_2 = -3, v_1 = -3.91$
- $u_3 = 0.91, v_3 \approx -0.08$

- $u_4 = 0.99, v_4 = -0.0015$

Finally, $f(x) \approx -79.67, u_t \approx 0.9902$. (1 pnt)

图 1中显示的病态 Hessian 矩阵下，动量的理论好处：

Momentum aims primarily to solve two problems: poor conditioning of the Hessian matrix and variance in the stochastic gradient. In fig 1, we illustrate how momentum overcomes the first of these two problems. The contour lines depict a quadratic loss function with a poorly conditioned Hessian matrix. At each step along the way, we draw an arrow indicating the step that gradient descent would take at that point. We can see that a poorly conditioned quadratic objective looks like a long, narrow valley or canyon with steep sides. Momentum correctly traverses the canyon lengthwise, while gradient steps waste time moving back and forth across the narrow axis of the canyon.

(1 分)

在本题目中 $f(u)$ 是具有极小值的三次函数，梯度不会有震荡，所以反而是普通梯度下降好（1 分）

Problem 3: Forward-Backward-Pass

Examine the multi-layer perceptron given in fig 2.

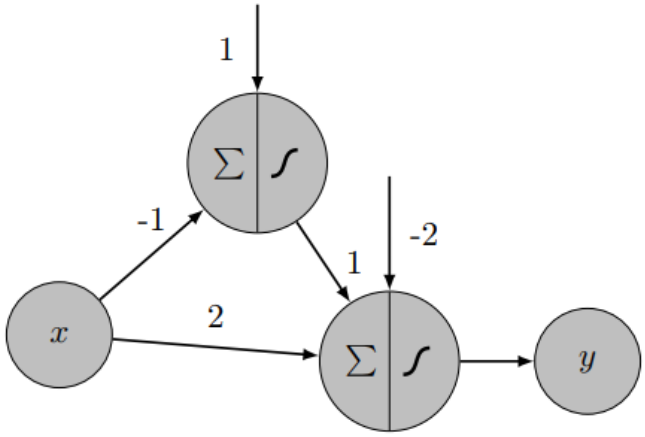
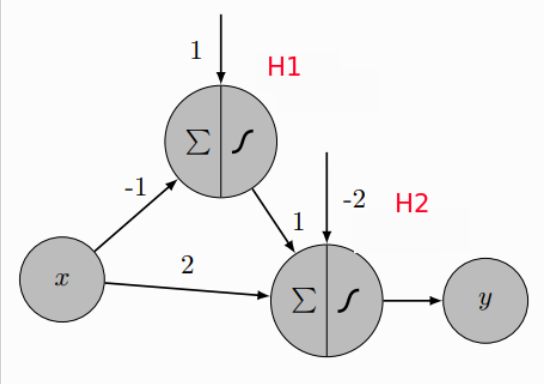


Figure 2: MLP with two logistic units

a). Both neurons use the logistic activation function $f(u) = \frac{1}{1+e^{-u}}$ The network has a single input variable x and one output variable y. Calculate the output of both neurons and the error made by the MLP when applying a pattern with $x = 0$ and target value 0.5.

b). Calculate the partial derivatives of the square error $\frac{1}{2}(\hat{y} - y)^2$ with respect to the weights for the pattern used in task a).

Solution 3.1: *Forward-Backward-Pass*



Assume activation of H1 is $h_1 = f(w_1 x + b_1)$, activation of H2 is $h2 = f(w_2 x + w_3 h_1 + b_2)$

output of H1 = 1; activation of H1 = 0.73; output of H2 = -1.27; activation of H2 = 0.22; error=0.0393 (共 1 分，部分对 0.5)

$\partial e/\partial w_3 = -0.035, \partial e/\partial w_2 = 0, \partial e/\partial w_1 = 0, \partial e/\partial b_2 = -0.048, \partial e/\partial b_1 = -0.0094$ （1 分）

Problem 4: Multi-Layer Perceptron Architecture

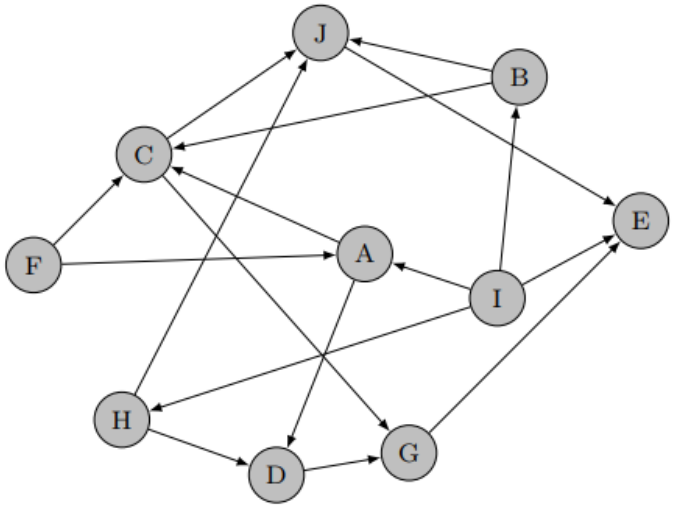


Figure 3: MLP with ten neurons

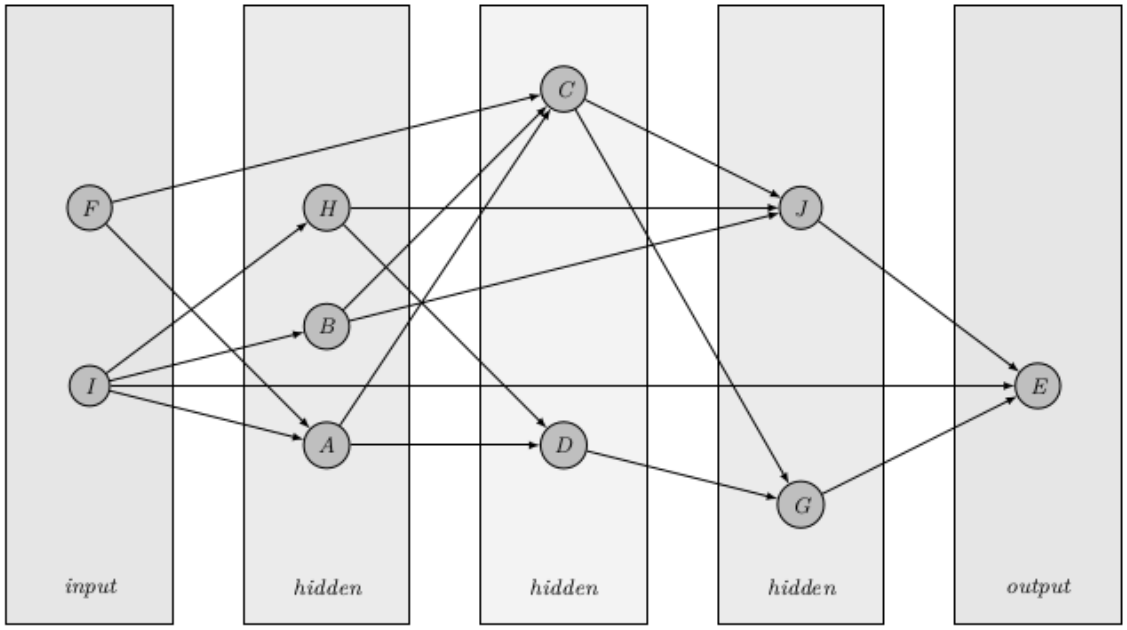
Now, consider the network structure of a multi-layer perceptron with 10 neurons given in fig 3. Each circle denotes a neuron, the arrows denote connections between neurons.

a). Which of the neurons are input neurons, which ones are output neurons?

b). How many layers does this MLP have? Which neurons belong to which layer?

c). Assume we are applying a pattern to the MLP. Give an order in which the neuron activations a_i can be calculated.

Solution 4.1: *Multi-Layer Perceptron Architecture*



- *input neurons: F, I*
- *output neurons: E*
- *number of layers: 5*
- *possible order: I,F,H,B,A,D,C,J,G,E*

input and output neuron (1 pnt), the number of layers (1 pnt), optional order (1 pnt)

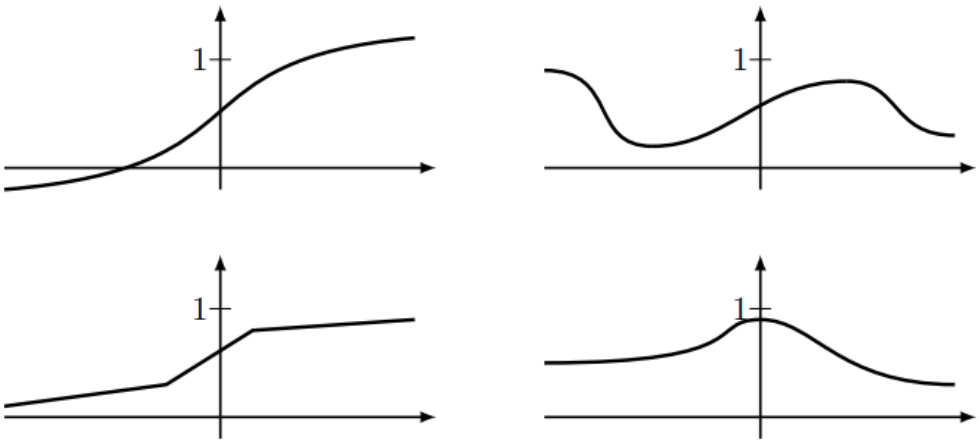


Figure 4: Functions to be realized by MLPs

Problem 5: Multi-Layer Perceptrons

Which of the functions given by the plots in fig 4 can be implemented by multi-layer perceptrons? The MLP should only contain neurons with logistic activation functions. (Note: The weights of the networks must be finite numbers.)

Solution 5.1: *Multi-Layer Perceptron*

- TL: no (since logistic function yields values between 0 and 1) (.5 pnt)
- TR: yes (.5 pnt)
- BL: no (function must be differentiable) (.5 pnt)
- BR: yes (.5 pnt)

Problem 6: MLP and BackPropogation

- Download *hw2.zip* from ftp(10.13.71.168), The code pass test under Python 2.7.14 and Python 3.6.3. Complete code surrounded by "TODO" in *net.py* and *opt.py*. For example,

```
################################################################################
# TODO: Implement the affine forward pass. Store the result in out. You   #
# will need to reshape the input into rows.                                #
################################################################################


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

- Run *python main_check.py* and paste the settings (*e.g.* random seed, loss function name) and the results to your report. Make sure your code pass gradient check, *e.g.*, relative error between Numerical gradient and analytic gradient is small (*e.g.* smaller than 1e-7). We use centered formula $\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{f(x+h)-f(x)}{h}$ to compute numerical gradient since it has an error on order of $O(h)$ and use relative error as metric.
- Run *python main_train.py* to train a model on generated fake data.
- Bonus(answer one of them is enough): Do something extra surrounding the topics in this assignment, using the code you developed. For example, is there some other interesting question we could have asked? Is there any insightful visualization you can plot? For example, you can compare the convergence speed and performance of linear regression and/or logistic regression and/or two-class softmax classification. You can select **some** samples (*e.g.* 50 or 500) in cifar10 or use the dataset in Problem 7: to do real case analysis. Will MLP overfit on small subset on cifar10? If the generated fake data is imbalanced, what would happen and how to deal with it? *et. al.*
- You do not necessarily strictly use the framework provided by teacher assistant. You can use any language and framework and write from scratch. Even tensorflow/pytorch with autograd is permitted, if you have time. Just make sure implement MLP by yourself and then do similar experiments, *i.e.*, do not use any function like MLP or LinearRegression.
- Finally, upload to ftp(10.13.72.84) before the deadline(Sunday, April 1, 23:59:59).

Solution 6.1: MLP and BackPropogation

After filling the code and running the *main_check.py*, I get the results as following. There are other parameters settings and results, so we just need to make sure relative errors smaller than 1e-7.

```
loss use softmax
```

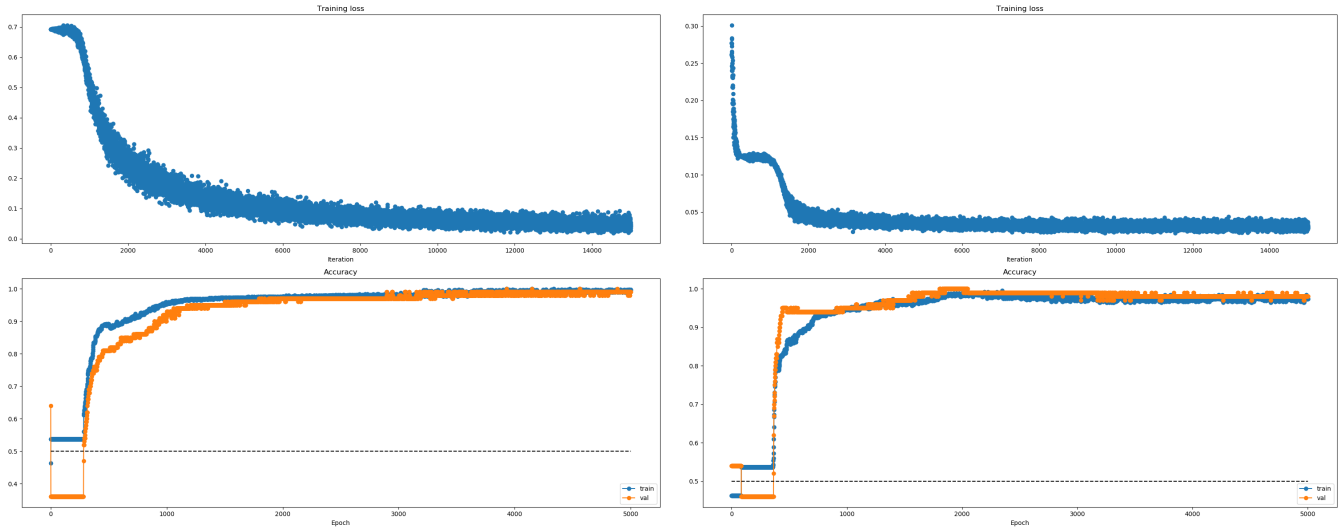
```
----------------
Testing test-time forward pass
scores are [[-1.372  0.972]
 [-0.876  1.132]
 [-0.38   1.292]]
----------------
Testing training loss
loss is 1.4685851981586016
W1 relative error: 4.65e-11
W2 relative error: 1.27e-11
b1 relative error: 5.55e-12
b2 relative error: 1.19e-11
```

相对误差小于 1e-7, 说明 net.py 中前向后向的 5 个代码空缺都正确（2.5 分）

I set the hyper-parameters as

```
Left fig:
N, D, H, C = small_data['X_train'].shape[0], 2, 10, 2
loss = 'softmax'

Right fig:
N, D, H, C = small_data['X_train'].shape[0], 2, 10, 1
loss = 'mse'
```



learning curve 收敛，说明 net.py 和 opt.py 中 3 个代码空缺正确（1.5 分）

Bonus:

- If the generated fake data is imbalanced, what would happen and how to deal with it?

  If the generated fake data is imbalanced, the classification accuracy is not that reliable. For example, if there are two classes and the label of 95 percents of input is 1 and else are -1, then

a rather simple strategy is to predict 1 no matter what the input is. Upsample or Downsample to make sure that within a batch the positive and negative is balanced. (1 pnt)

- Compare the convergence speed and performance of linear regression and/or logistic regression and/or two-class softmax classification.

  Linear/logistic regression maybe more stable than two-layer neural network. (1 pnt)

- You can select **some** samples (*e.g.* 50 or 500) in cifar10 or use the dataset in Problem 7: to do real case analysis. Will MLP overfit on small subset on cifar10?

  MLP will overfit. (1 pnt)

- Other reasonable point. (1 pnt)

Problem 7: Predict House Prices

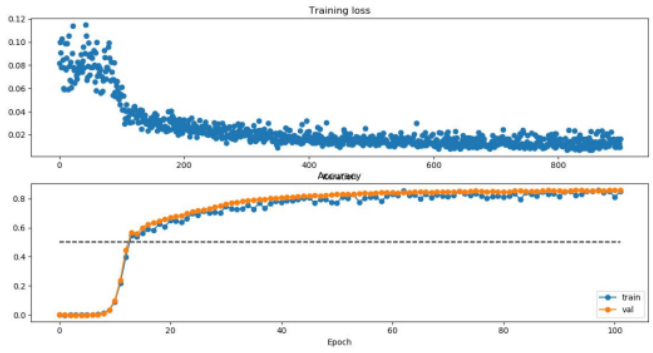
Given 79 explanatory variables describing (almost) every aspect of residential homes, such as the size and the location of the house, please write down the r2 score of your model in the report.

- Download *hw2.zip* from ftp(10.13.71.168). The dataset is cleaned to become simple train/val format. You can try to modify the process of data clean.
- Make sure implement the linear regression model by yourself and do not use 'LinearRegression' in 'sklearn'. Linear regression optimized by sgd is similar to MLP, which have been implemented in Problem Problem 6:. You can also use closed form solution for linear regression with/without l_2 regularization.

  hint: l_2 regularization is equivalent to pertubation, makes sure the inverse of matrix exits.
- Bonus(answer one of them is enough): Do something extra surrounding the topics in this assignment, using the code you developed. For example, try lasso regression and/or huber loss and/or learning rate decay and/or select best model by cross validation. What would happen if the x and y do not subtract mean? If you implement linear regression by closed-form formulation and sgd, you can compare them on speed and performance. *et. al.*
- You do not necessarily strictly use the framework provided by teacher assistant. You can use any language and write from scratch. Just make sure implement regression model by yourself and then do similar experiments.
- Finally, upload to ftp(10.13.72.84) before the deadline(Sunday, April 1, 24:00:00).

Solution 7.1: Predict House Prices

Learning curve for linear regression should be similar, r2_socre: 0.860.88 (1 pnt)



Bonus:

- lasso regression and/or huber loss and/or learning rate decay and/or select best model by cross validation. (1 pnt)
- If you implement linear regression by closed-form formulation and sgd, you can compare them on speed and performance. (1 pnt)

Pay attention to tuning lr for different model and make sure it converge. On this small dataset, there is slight difference between different method

- What would happen if the x and y do not subtract mean? May not converge. (1 pnt)
- Other reasonable point. (1 pnt)