CSCI 497P / 597P Winter 2019

HW2

Due: Thursday, March 14, 2019 at 9:59pm

This homework is optional; if you submit it, your grade (once again, graded on a faithful-effort basis) will be averaged with your HW1 grade. Otherwise, your homework grade will depend only on your HW1 grade. You are free to collaborate however you see fit, but if you are submitting the homework you must write up your solutions yourself.

1 Projective and Two-View Geometry

1. For each of the following components of a camera projection matrix, describe its entries and how they relate to the different camera parameters. Also specify the coordinate system of the input and output of the transformation.

   (a) Extrinsics matrix

   (b) Projection matrix

   (c) Intrinsics matrix

2. The funamental matrix is not full rank.

   (a) Give its rank, explain the geometric significance of the rank.

   (b) Explain the geometric interpretation of its null space. As a reminder, the null space of a matrix T is the set of vectors x such that $Tx = 0$.

3. In this problem, we will unsuccessfully attempt to break math by finding the intersection point of parallel lines.

   (a) Given the homogeneous coordinates of a line $\ell = [a, b, c]^T$, derive an expression for the slope of the line. You may assume the slope is not undefined.

   (b) Give a criterion for determining whether two homoegeneous lines $\ell_1 = [a_1, b_1, c_1]^T$ and $\ell_2 = [a_2, b_2, c_2]^T$ are parallel. Again, assume their slopes as derived in part (a) are not undefined.

   (c) It turns out that in projective space, the intersection point of two parallel lines is well-defined. Calculate the intersection of ℓ_1 and ℓ_2 above.

(d) Using the intersection point's coordinates and the parallel criterion from part (b), what can you say about the homogeneous coordinates of the intersection of two parallel lines?

(e) A homoegeneous point with the above property is called a ″point at infinity″. Describe the meaning of a ″point at infinity″ using the more intuitive 3D interpretation of 2D projective space.

4. You are given an uncalibrated stereo pair. Is it possible to generate a 3D model of the scene? If so, explain how, and any limitations of the approach. If not, explain why it's impossible.

2 Photometric Stereo

5. For calibrated photometric stereo to work, how many independent light directions are needed at a minimum? Why?

6. Suppose we're solving a photometric stereo system $I = L^T N$. If n is the number of images, p, is the number of pixels in an image, I is $n \times p$, L is $3 \times n$, and N is $3 \times p$. If light sources are not directional (e.g., lights are point sources close to the scene being imaged), why does the matrix equation no longer hold?

3 Recognition and Machine Learning

7. As we saw in Project 5, there is no ReLU applied after the final layer in the AlexNet architecture, meaning that the raw scores that get fed into softmax/cross-entropy can take negative values. Given the probabilistic interpretation of the softmax-normalized score vectors, why might applying ReLU be a bad idea?

8. You are training a binary linear classifier on a 3-dimensional feature space. The loss function to be optimized is an SVM loss with L2 regularization:

$$L_i(x_i; w, b) = \max(0, 1 - y_i(w^T x + b)) + \frac{\lambda}{2}||w||_2^2$$

where x is a 3-dimensional data point, w is a 3-vector of parameters, b is a scalar bias parameter, and λ is a hyperparameter (i.e., it is fixed during training). Your classifier is a Python class that has members to store its parameters (w, b), the intermediate values of the computation, and the gradients of the loss wrt each of the intermediate values and parameters. The forward pass of the classifier is written as follows:

```
def forward(x, y):
    """ Compute the loss for datapoint x with true label
    y (-1 or 1) """
    self.reg = 0.5 * self.lambda * np.dot(self.w, self.w)
    self.p = np.dot(self.w, x) + b
    self.r = 1 - y * self.p
    self.s = max(0, self.r)
    self.loss = self.s + self.reg
```

(a) Write the backward pass to compute of the loss with respect to each intermediate value and parameter. Notice that $\frac{\partial L}{\partial w}$ is a vector of partial derivatives, one for each element of the w parameter vector.

- Hint 1: you may find it helpful to write the dot products in the forward pass in scalar notation.
- Hint 2: the gradient of the loss with respect to w depends on both the SVM loss term and the regularization term; because these two contributions summed in the loss function, the gradient wrt w is also a sum of the gradients from each of these terms. In math terms, because

  $$L = s + reg$$

  the gradient of L wrt w is

  $$\frac{\partial L}{\partial w} = \frac{\partial s}{\partial w} + \frac{\partial reg}{\partial w}$$

```
backward(x):
    """ Compute the backwards pass. Each variable’s
    name refers to the gradient of self.loss with
    respect to the named variable. For example, dp
    is dLoss/dp. Assume that forward(x) has been run,
    so the forward pass has populated self.reg, self.p,
    self.r, self.s, and self.loss. """

    self.ds =


    self.dr =


    self.dp =


    self.dreg =


    self.db =


    self.dw =
```

(b) Implement a training loop in the following method; assume this is a method of your classifier, so you can access the above variables and methods using `self`. Your training loop should iterate over the entire dataset once and update its parameters using SGD (i.e., batch size 1). You can assume X has been shuffled so there's no need to sample points randomly.

```
def train(X, Y, lr):
    """ Train this classifier using vanilla SGD.
    X is n-by-3; each row is a data point
    Y is n-by-1; each entry is -1 or 1, the correct
        class label for the corresponding point in X.
    lr is hte gradient descent step size (learning rate)
    """
```

9. What characteristic must an activation function satisfy in order to serve its purpose?

10. Why is a sigmoid a bad choice for activation function?

11. Consider the following architecture, which is very similar to AlexNet you used in Project 5. For brevity, we've excluded the ReLU operations, because they don't change the dimensions or require any parameters.

```
nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2), # conv1
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(64, 192, kernel_size=5, padding=2), # conv2
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(192, 384, kernel_size=3, padding=1), # conv3
nn.Conv2d(384, 256, kernel_size=3, padding=1), # conv4
nn.ReLU(inplace=INPLACE),
fnn.Conv2d(256, 256, kernel_size=3, padding=1), # conv5
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Linear(256 * 6 * 6, 4096),       # fc1
nn.Linear(4096, 4096),              # fc2
nn.Linear(4096, 1000)               # fc3
```

(a) The network's input dimenksions are 224x224. Calculate the size of the feature map after each layer in the network above.

i. conv1

ii. conv2

iii. conv3

iv. conv4

v. conv5

vi. fc1

vii. fc2

viii. fc3

(b) Now, calculate the number of parameters are required for each layer.

i. conv1

ii. conv2

iii. conv3

iv. conv4

v. conv5

vi. fc1

vii. fc2

viii. fc3

(c) To train the network on a GPU, we need to store arrays containing each feature map and each set of parameters in GPU RAM. Further, it has to store the gradient of the loss wrt each set of parameters. Assuming feature maps and parameters all have type float32 and thus require 4 bytes per element to store, how much GPU RAM would be necessary to train AlexNet? Note that we're ignoring any overhead for each array, space required to store the input image, and a variety of other details.

(d) In practice, we train by pushing batches of images through the network at once. If my GPU has 6GB of RAM, what's the largest batch size I could use?