COGS 185, Spring 2016
Advanced Machine Learning Methods

Final Project

Due date: June 10 (Friday), 11:59pm, 2016
Late policy: (1) a grace period of 3 hours will be given, which means no penalty assigned if you turn it in no later than 12/11/2015, 2:59Am; (2) every 5% of the total points will be deducted for every extra day past due.
Policy for teamwork: Option 2 or 3 below is a single person project; if you only choose one of them and will be basically following the instructions, you have to work on your own and no teamwork is allowed. For option 1, if you form a two-person team, you will at least need to try out two different models (e.g. AlexNet and LeNet etc.) on two datasets (MNIST, CIFAR-10, MiniImageNet etc.).
For option 4, please talk to me about your basic idea and to make sure your idea and plan are doable and suitable for the final project. Try to avoid two potential pitfalls: (1) your idea might be too ambitious that cannot be finished within 2-3 weeks; or (2) your idea is too simple that is not justifiable as a valid course project.

Submission: submit your final report in pdf format to ted.ucsd.edu

Report format:

Write a report with >1,500 words including main sections: a) abstract, b) introduction, c) method, d) experiment, e) conclusion, and f) references. You can follow the paper format as e.g leading machine learning journals such as Journal of Machine Learning Research (http://www.jmlr.org/) or IEEE Trans. on Pattern Analysis and Machine Intelligence (http://www.computer.org/web/tpami), or leading conferences like NIPS (https://papers.nips.cc/) and ICML (http://icml.cc/2016/?page_id=151).

See below a link about writing a scientific paper:

http://abacus.bates.edu/~ganderso/biology/resources/writing/HTWtoc.html

The format of your references can be of any kind that is adopted in the above journals or conferences.

Grading:

The merit and grading of your project can be judged from aspects described below that are common when reviewing a paper:

1. Interestingness of the problem you are studying. (10 points)
2. How challenging and large is the dataset you are studying? (10 points)
3. Any aspects that are new in terms of algorithm development, uniqueness of the data, or new applications? (20 points)
4. Is your experimental design comprehensive? Have you done thoroughly experiments in tuning hyper parameters? (30 points)
5. Is your report written in a professional way with sections including abstract, introduction, data and problem description, method description, experiments, conclusion, and references? (30 points)
6. Bonus points will be assigned to projects that have adopted more than one evaluation metrics, tested on more algorithms, tried new methods, and worked on novel applications.

There will be three options for the final project (if you have your own idea, please come to talk to me):

Option (1): Convolutional neural networks

Train a convolutional neural networks method on Tiny ImageNet dataset (http://pages.ucsd.edu/~ztu/courses/tiny-imagenet-200.zip)

You can choose any deep learning platforms including MatConvNet (http://www.vlfeat.org/matconvnet/), Theano (http://deeplearning.net/software/theano/), Torch (http://torch.ch/), Caffe (http://caffe.berkeleyvision.org/), MxNet (https://github.com/dmlc/mxnet), and TensorFlow. See in the link other possible platforms you can use: http://deeplearning.net/software_links/

You can train a model by building your own network structure or by adopting/following standard networks like AlexNet, GoogLeNet, VGG, etc.

You are also welcome to use datasets other than ImageNet, e.g. CIFAR-10, and Kaggle datasets (https://www.kaggle.com/) such as the deep sea image competition: https://www.kaggle.com/c/datasciencebowl

Option (2): Deep belief networks

from G. Hinton (http://www.cs.toronto.edu/~hinton/csc2515/projects/deep1.html)
His course webpage: http://www.cs.toronto.edu/~hinton/csc2515/lectures.html

Details about the deep belief networks (DBN) can be found in their Science paper:
Hinton, G. E., Osindero, S., Welling, M. and Teh, Y.
Unsupervised Discovery of Non-linear Structure using Contrastive Backpropagation.
http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog0000_76/epdf

Links to some other packages that you might consider using:
http://deeplearning.net/software_links/

Try to use the MNIST dataset. Code and data are available at http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html

Task: Try to use varying number of training samples vs. testing samples to see the behavior of the DBN algorithm. Also apply SVM on the same setting and compare the performances of DBN with SVM.

Wen using DBN, you will need to determine how many hidden layers the DBN should have, how many units each layer has, and how long the pre-training should be.

Due to the big overhead in training deep learning, you can try to reduce your training data by trying e.g. 1,000 and 3,000 samples.

Other variations you can consider by fixing training size to e.g. 1,000 samples:

a. Number of hidden layers: 1, 2, and 3
b. Number of units in the first layer: 100, 500
c. Number of units in the second layer: 200, 500
d. Learning rate: 0.05, 0.1, 0.5, 1.0.

Option (3): Sparse coding

You will download the code from the following webpage:

http://redwood.berkeley.edu/bruno/sparsenet/

based on paper

[1] http://redwood.berkeley.edu/bruno/papers/nature-paper.pdf

and
[2] http://redwood.berkeley.edu/bruno/papers/VR.pdf

http://ufldl.stanford.edu/wiki/index.php/Sparse_Coding

(1) Change the step size or learning rate of the gradient descent, to observe how the S variance and L2 norm change over the number of trials w.r.t. different, e.g. learning rates.
(2) Change the parameter lambda for sparsity term and see what happens. lambda was computed as lambda = 2*sigma*sigma*beta shown in eqn. (14) in paper [2]. Here you can fix sigma and vary beta to change the values of lambda.
(3) Change the sparsity function (from Cauchy to Laplacian) and see what happens. Cauchy function is referred as S(x) = log(1+x*x) in pg. 4 under eqn. (8) in paper [2]. Laplacian is computed as S(x) = |x|.

You will need to change the function in "cgf.c"

Cauchy:
--
```
double sparse(x)
double x;
{
  return(log(1.0+x*x));
}

double sparse_prime(x)
double x;
{
  return(2*x/(1.0+x*x));
}
```

--

to Laplacian:

```
double sparse(x)
double x;
{
  return(fabs(x));
}

double sparse_prime(x)
double x;
```

```
{
  if (x>0)
   return +1;
else if (x<0)
  return -1;
else
  return 0.0;
}
```

(4) Change the number of basis elements from 2 fold overcomplete to 3 fold overcomplete and see what happens.

2-fold overcompleteness would have A matrix as:

```
A = rand(64, 128)-0.5;
A = A*diag(1./sqrt(sum(A.*A)));
```

3-fold overcompleteness would have A matrix as:

```
A = rand(64, 192)-0.5;
A = A*diag(1./sqrt(sum(A.*A)));
```

(5) Run the code on unwhitened and whitened data, and see the differences.
(6) Write a coherent report, consisting a) a brief description of sparse coding learning. Explain why we want to learn a sparse code, and describe the precise learning algorithm, using either the notation of the Nature paper or the notation of the lecture. b) report the findings in (1)-(5).

Option (4): A cool application of your own choice

If you have anything specific you want to, which is related to our course, you are welcome to discuss with me. Some interesting projects in which more advanced applications can be tried out using convolutional neural networks including, image captioning (http://cs.stanford.edu/people/karpathy/deepimagesent/), visual q&a (http://www.visualqa.org/), and facial analysis (https://www.microsoft.com/cognitive-services/).

Some sample projects of CS231n at Stanford can be found here: http://cs231n.stanford.edu/reports.html.