A few announcements

• This Friday: Practical deep learning tutorial
  – Familiarity with Python and numpy is useful.
  – Focus is on using Theano: http://deeplearning.net/software/theano/

• Next week: I am away Mon-Wed
  – My office hours will be covered by Ryan (Tues 12-1pm, MC111).
  – For the Wednesday (Nov.11) class Ryan and Pierre-Luc will lead a hands-on problem solving session. Problem set will be posted on the discussion board.

• Project 3 due next Wednesday. CMT ready for submission.
The deep learning objective

very high level representation:

MAN

SITTING

... etc ...

slightly higher level representation

raw input level representation:

Learning an autoencoder function

- **Goal:** Learn a compressed representation of the input data.

- **We have two functions:**
  - Encoder: \( h = f_w(x) = s_f(Wx) \)
  - Decoder: \( x' = g_w(h) = s_g(W'h) \)
  where \( s() \) can be a sigmoid, linear, or other function and \( W, W' \) are weight matrices.

- **To train, minimize reconstruction error:**
  \[
  Err(W,W') = \sum_{i=1}^{n} L [ x_i , g_w(f_w(x_i)) ]
  \]
  using squared-error loss (continuous inputs) or cross-entropy (binary inputs).
### PCA vs autoencoders

In the case of a linear function:

\[ f_W(x) = Wx \quad g_{W'}(h) = W'h, \]

with squared-error loss:

\[ \text{Err}(W, W') = \sum_{i=1}^{n} || x_i - g_W(f_W(x_i)) ||^2 \]

we can show that the minimum error solution \( W \) yields the same subspace as PCA.

---

### Stacked autoencoders

**Key idea:** Apply greedy layerwise unsupervised pre-training.

http://www.dmi.usherbrooke.ca/~larocheh/projects_deep_learning.html
Regularization of autoencoders

- How can we generate **sparse** autoencoders? (And also, why?)
- **Weight tying** of the encoder and decoder weights ($W=W$) to explicitly constrain (regularize) the learned function.
- Directly **penalize the output of the hidden units** (e.g. with L1 penalty) to introduce sparsity in the weights.
- **Penalize the average output** (over a batch of data) to encourage it to approach a fixed target.

Denoising autoencoders

- **Idea**: To force the hidden layer to discover more **robust features**, train the autoencoder with a corrupted version of the input.
- **Corruption processes:**
  - Additive Gaussian noise
  - Randomly set some input features to zero.
  - More noise models in the literature.
- **Training criterion:**
  \[
  Err(W,W) = \sum_{i=1:n} E_{q(x|x)} L \{ x_i, g_W(f_W(x_i)) \}
  \]
  where $x$ is the original input, $x'$ is the corrupted input, and $q()$ is the corruption process.
Contractive autoencoders

- **Idea**: Here again, goal is to learn a representation that is robust to noise and perturbations of the input data.

- **Contractive autoencoder training criterion**:

  \[ Err(W,W') = \sum_{i=1}^{n} L \left[ x_i, g_W(f_W(x_i)) \right] + \lambda \|J(x)\|^2 \]

  where \( J(x) = \frac{\partial f_W(x)}{\partial x} \) is a Jacobian matrix of the encoder evaluated at \( x \), \( F \) is the Frobenius norm, and \( \lambda \) controls the strength of regularization.

  \( J(x) = f_W(x)\left(1-f_W(x)\right)w_j \) (assuming \( f() \) is a sigmoid)

  \[ \|J(x)\|^2 = \sum_j \left( f_W(x)\left(1-f_W(x)\right) \right)^2 \|w_j\|^2 \]
Supervised learning with deep models

Alternatively: Use the last representation layer (or concatenate all layers) as an input to a standard supervised learning predictor (e.g. SVM).

Empirical results: Rifai et al. (ICML 2011)

Table 1. Performance comparison of the considered models on MNIST (top half) and CIFAR-10 (bottom half). Results are sorted in ascending order of classification error on the test set. Best performer and models whose difference with the best performer was not statistically significant are in bold. Notice how the average Jacobian norm (before fine-tuning) appears correlated with the final test error. SAT is the average fraction of saturated units per example. Not surprisingly, the CAE yields a higher proportion of saturated units.

<table>
<thead>
<tr>
<th>Model</th>
<th>Test error</th>
<th>Average/|F(x)|_F</th>
<th>SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAE</td>
<td>1.14</td>
<td>0.73 x 10^-8</td>
<td>96.36%</td>
</tr>
<tr>
<td>DAE-g</td>
<td>1.18</td>
<td>0.86 x 10^-8</td>
<td>77.77%</td>
</tr>
<tr>
<td>RBM-binary</td>
<td>1.30</td>
<td>2.50 x 10^-8</td>
<td>78.59%</td>
</tr>
<tr>
<td>DAE-b</td>
<td>1.37</td>
<td>7.87 x 10^-8</td>
<td>68.19%</td>
</tr>
<tr>
<td>AE+wd</td>
<td>1.68</td>
<td>5.00 x 10^-8</td>
<td>12.99%</td>
</tr>
<tr>
<td>AE</td>
<td>1.78</td>
<td>17.5 x 10^-5</td>
<td>49.00%</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAE</td>
<td>47.86</td>
<td>2.40 x 10^-4</td>
<td>86.65%</td>
</tr>
<tr>
<td>DAE-b</td>
<td>40.08</td>
<td>4.85 x 10^-4</td>
<td>90.66%</td>
</tr>
<tr>
<td>DAE-g</td>
<td>58.81</td>
<td>4.94 x 10^-4</td>
<td>19.00%</td>
</tr>
<tr>
<td>AE+wd</td>
<td>45.03</td>
<td>34.19 x 10^-5</td>
<td>23.04%</td>
</tr>
<tr>
<td>AE</td>
<td>55.57</td>
<td>44.9 x 10^-5</td>
<td>22.57%</td>
</tr>
</tbody>
</table>
Variety of training protocols

• **Purely supervised:**
  – Initialize parameters randomly.
  – Train in supervised mode (gradient descent w/backprop.)
  – Used in most practical systems for speech and language.

• **Unsupervised, layerwise + supervised classifier on top:**
  – Train each layer unsupervised, one after the other.
  – Train a supervised classifier on top, keeping other layers fixed.
  – Good when very few labeled examples are available.

• **Unsupervised, layerwise + global supervised fine-tuning:**
  – Train each layer unsupervised, one after the other.
  – Add a classifier layer, and retrain the whole thing supervised.
  – Good when label set is poor.

• **Unsupervised pretraining often uses regularized autoencoders.**

From: [http://www.slideshare.net/philipzh/a-tutorial-on-deep-learning-at-icml-2013](http://www.slideshare.net/philipzh/a-tutorial-on-deep-learning-at-icml-2013)

---

Major paradigms for deep learning

• **Deep neural networks:** The model should be interpreted as a computation graph.
  – Unsupervised pre-training: Stacked autoencoders.

• Special architectures for different problem domains.
  – Computer vision => Convolutional neural nets.
  – Text and speech => Recurrent neural nets.
Neural networks for computer vision

- Design neural networks that are specifically adapted
  - Can deal with very high-dimensional inputs
    - E.g. 150x150 pixels = 22,500 inputs, or 3x22,500 if RGB
  - Can exploit the 2D topology of pixels (or 3D for video)
  - Can build in invariance to certain variations we can expect
    - Translations, illumination, etc.

- **Convolutional neural networks** leverage several ideas.
  1. Local connectivity.
  2. Parameter sharing.
  3. Pooling / subsampling hidden units.
Neural networks for computer vision

- A few key ideas:
  1. Features have **local receptive fields**.
     - Each hidden unit is connected only to a patch of the input image.
     - Units are connected to all colour channels.
  2. **Share matrix of parameters** across units.
     - Hidden units organized in “feature maps”.
     - Extract the same feature at several different positions in the image.
     - Compute feature map via discrete convolution with a kernel matrix.
Neural networks for computer vision

- A few key ideas:
  1. Features have local receptive fields.
  2. Share matrix of parameters across units.
  3. Pooling/subsampling of hidden units in same neighbourhood.

Convolutional neural nets (CNNs)

- Alternate between the convolutional and pooling layers.
- Fully connected layer at the end.
- Train full network using backpropagation.
Example: ImageNet

- SuperVision (a.k.a. AlexNet, 2012):
  - **Deep**: 7 hidden "weight" layers
  - **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
  - Entirely supervised
  - More data = good

Training results: ImageNet

- 96 learned low-level filters

Empirical results (2012)

ImageNet 1K competition, fall 2012

Empirical results

- Results for image retrieval (query items in leftmost column):


Empirical results (2013)

Test error (top-5)
Empirical results (2015)

ILSVRC top-5 error on ImageNet

Achieving super-human performance?

- Estimated 3% error in the labels.
- Differences between labeling process and human assessment:
  - Labels acquired as binary task. *Is there a dog in this picture?*
  - Human performance measured on 1K classes (>120 species of dogs in the dataset).
  - Labels acquired from experts (dog experts label the dogs, etc.).
- Machines and humans make different kinds of mistakes.
  - Both have trouble with multiple objects in an image.
  - Machines struggle with small/thin objects, image filters.
  - Humans struggle with fine-grained recognition.
**Tasks for which CNNs are the best (machine) learner**

- **Handwriting recognition** MNIST (many), Arabic HWX (IDSIA)
- **OCR in the Wild** [2011]: StreetView House Numbers (NYU and others)
- **Traffic sign recognition** [2011] GTSRB competition (IDSIA, NYU)
- **Pedestrian Detection** [2013]: INRIA datasets and others (NYU)
- **Volumetric brain image segmentation** [2009] connectomics (IDSIA, MIT)
- **Human Action Recognition** [2011] Hollywood II dataset (Stanford)
- **Object Recognition** [2012] ImageNet competition
- **Scene Parsing** [2012] Stanford bgd, SiftFlow, Barcelona (NYU)
- **Scene parsing from depth images** [2013] NYU RGB-D dataset (NYU)
- **Speech Recognition** [2012] Acoustic modeling (IBM and Google)
- **Breast cancer cell mitosis detection** [2011] MITOS (IDSIA)

The list of perceptual tasks for which ConvNets hold the record is growing. Most of these tasks (but not all) use purely supervised convnets.

*From: [http://www.slideshare.net/philipzh/a-tutorial-on-deep-learning-at-icml-2013](http://www.slideshare.net/philipzh/a-tutorial-on-deep-learning-at-icml-2013)*

---

**Do we really need deep architectures?**

- We can approximate any function to arbitrary levels of precision with shallow (2-level) architectures.

- Deep learning is more efficient for representing certain classes of functions, where there is certain types of structure.
  - Natural signals (images, speech) typically have such structure.

- Deep learning architectures can represent more complex functions with fewer parameters.
  - Trade-off (less) space for (more) time.

- So far, very little theoretical analysis of deep learning.
Quick recap + more resources

• A good survey paper:

• Other notes and images in today’s slides are from:
  – Lecture slides from Geoff Hinton, Hugo Larochelle, Aaron Courville
    • http://www.cs.toronto.edu/~hinton/csc2535
  – http://www.slideshare.net/philipzh/a-tutorial-on-deep-learning-at-icml-2013

What you should know

• Types of deep learning architectures:
  – Stacked autoencoders
  – Convolutional neural networks

• Typical training approaches (unsupervised / supervised).
• Examples of successful applications.