Deep Learning Basics
April 15, 2016

Sebastian Stober <sstober@uni-potsdam.de>
• Deep Learning textbook:
  – http://www.deeplearningbook.org/

• standard ML textbooks:
Additional Resources

• online lectures:
  – [https://www.coursera.org/course/neuralnets](https://www.coursera.org/course/neuralnets)  

• generally useful:
Machine Learning Basics
“lazy engineer” approach:

How to program a machine \textit{to learn} to solve task XY?

- generalizes to new / similar problems
• **Learning:**
  – Improving with experience at some task

• **Formal definition:**
  – Improve over task $T$,
  – with respect to performance measure $P$,
  – based on experience $E$. 

Defining Learning

[MITCHELL 1997]
Machine Learning Tasks: Supervised vs. Unsupervised

Supervised Learning
Unsupervised Learning
Defining Learning

• Learning:
  – Improving with experience at some task

• Formal definition:
  – Improve over task T,
  – with respect to performance measure P,
  – based on experience E.

generalization on unseen (test) data
training examples
Bias-Variance Trade-Off

• conflict: choose model that
  – accurately captures the training data regularities
  – generalizes well to unseen data
• bias
  – error from (implicit or explicit) assumption of the learning algorithm => underfitting
• variance
  – error from fitting to small fluctuations => overfitting
Bias-Variance Trade-Off

- high bias, low variance:

- low bias, high variance:

- good trade-off:
How to improve generalization performance?
• underfitting: more training or increase model capacity
• overfitting: less training or decrease model capacity
Model Capacity

- decide when to stop based on validation set
Model Capacity

• intuitive definition: model’s ability to fit
• statistical learning theory: VC dimension

• optimal performance when model capacity appropriate for task complexity
  – higher-capacity models can solve harder tasks but are more prone to overfitting
• effective capacity may be reduced by limits of the learning algorithm => in deep learning
Hyper-Parameters

• set **before** training starts
• typical roles:
  – determine (representational) model capacity, including regularization params
  – control training algorithm (effective capacity)
• important: Never use the test set to tune hyper-parameters!
From Machine Learning to Deep Learning
Typical Machine Learning Workflow (for Classification)

Data → (hand-crafted) feature extraction → "simple" trainable classifier → predicted labels

- make use of domain knowledge from experts
- train with labeled data (supervised)
Typical Deep Learning Workflow (for Classification)

- Data
- Trainable feature transform(s)
- Trainable classifier
- Predicted labels

“pre-train” with unlabeled data (unsupervised)

Train with labeled data (supervised)

Make use of abundant data and (GPU) compute power
Deep Learning Basics

- Deep learning
  - Example: MLPs
- Representation learning
- Machine learning
- AI

Example:
- Shallow autoencoders
- Logistic regression
- Knowledge bases

[deeplearningbook.org]
The Promise of Deep Learning

- learn suitable feature representations along with the actual learning task

- using a general-purpose learning procedure
An Example Deep Net

[Zeiler & Fergus 2014]
Neuron (Sigmoid Unit)

\[
\text{bias}
\]

\[
\text{non-linearity (sigmoid function)}
\]

\[
\text{inputs} \quad \text{connection weights} \quad \text{net activation} \quad \text{(output) activation}
\]

\[
\text{Deep Learning Basics}
\]
Common Activation Functions

- sigmoid
- tanh
- ReLU
- softplus
Single Neuron Capacity

range depends on act()

position controlled through bias b

$\Rightarrow$ linearly separable problems

[slide from Hugo Larochelle]
**Multi-Layer Perceptron**
forward propagation (activations)

\[
\begin{align*}
    h &= \text{act}_1(W_1^T x + b_1) \\
o &= \text{act}_2(W_2^T h + b_2) \\
    &= \arg\max(o)
\end{align*}
\]

- **Input layer**
- **Hidden layer**
- **Output layer**
- **Prediction**
- **Error**

=> loss function
• with enough hidden units
  => arbitrarily complex but smooth functions
• but: may become infeasibly large!  => go deeper!

[Mitchell 1997]
• used for multi-class classification
  – 1 output neuron per class
  – (optional) linear transform: $z = W^T x + b$
  – interpret $z$ as unnormalized conditional log probabilities given input $x$, i.e. $z_i = \log P'(y=i \mid x)$

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad \text{normalization (sum} = 1)$$

– predict class with highest probability (winner-take-all principle)
• How could a real softmax be obtained? (I.e. a soft version of the max function)
Cost Function

- goal: maximize log likelihood of target class \( i \)
  
i.e. maximize \( \log(P(y=i \mid z)) = \text{softmax}(z)_i \)

\[
\log \text{softmax}(z)_i = z_i - \log \sum_j \exp(z_j)
\]

- convert into a loss function:
  minimize negative log likelihood (NLL)
  
(also referred to as "Cross Entropy")
Error Backpropagation

for each training example \((x, y)\),
compute partial derivatives of loss function w.r.t. every single network parameter \((W, b)\)

\[
\begin{align*}
h &= \text{act}_1(W_1^T x + b_1) \\
o &= \text{act}_2(W_2^T h + b_2) \\
&= \arg\max(o)
\end{align*}
\]

\[
\text{for each training example } (x, y), \text{ compute partial derivatives of loss function w.r.t. every single network parameter } (W, b)
\]

\[
\text{prediction } \Rightarrow \text{loss function}
\]
Computation Graph

symbolic representation of the loss function
=> allows automatic differentiation

Deep Learning Basics
Gradient Descent

gradient at one point (current parameters) of error function => descent (step size?)

[Mitchell 1997]
Gradient Descent

a more realistic error function (for 2 weights)

=> can get stuck in local minima!

[slide from Andrew Ng]
Gradient Descent

- for all training examples (batch training)
  - can take very long
  - possible aggregation effects
- for each training example (online training)
  - less chance of getting stuck in local minima
  - can be very “jumpy”
- good trade-off: minibatch training
Deep Net Training
Building Blocks
Datasets

• features (network inputs)
  – topology (flat, structured, …)
  – variable length?

• targets (desired network outputs)
  – encoding (one-hot, plain labels, structured, …)

• optional meta-data
Datasets (2)

• partitioning
  – training set: train model
  – validation set: choose hyper-parameters
  – test set: estimate generalization performance

• normalization

• storage
  – in memory (on GPU?), DB
  – generated on-the-fly
Model Structure

• hidden layer types and dimensions
  – e.g. fully connected, convolutional, recurrent
• biases?
• activation functions
• output layer (e.g. softmax, linear, …)
Model Initialization

• general rule of thumb:
  – start with linear model behavior
• constant, identity
• random (uniform, Gaussian, …)
• sparse
• orthogonal
• pre-trained
• model domain knowledge

Why is simply setting everything to 0 a bad idea?
Cost Function
also: objective or loss function

• measures model quality
  – How much has it improved / learned?
  – How well does it generalize?

• very common choices:
  – cross entropy (supervised training)
  – reconstruction error (unsupervised training)
Regularization

• early stopping
• weight regularization / “weight decay”
  – L1 = penalize non-zero values
  – L2 = penalize extreme weights
• sparse activation
• Dropout = randomly set activations to zero
  – separate effects from correlated features
• DropConnect = randomly set weights to zero
Training Algorithm

• (full) batch gradient descent
  – update after seeing all examples (= 1 epoch)
• stochastic gradient descent (SGD)
  – update after each example
• minibatch SGD
  – update after each batch of examples
• second-order methods
  – Hessian-free optimization
Learning Rule

- scale
- Momentum
- AdaDelta
- AdaGrad
- RMSProp
- Adam
Termination Criterion

- fixed number of epochs
- fixed maximum runtime
- monitoring-based
  - weights change below threshold
  - training error
  - early stopping (validation error)

- combinations
Training Extensions

• monitoring
  – errors
  – gradients
  – weights
• plotting
• parameter adjustments
• save model
  – keep track of best model so far