DEEP LEARNING TUTORIAL

Peter Sadowski

November 11, 2015

University of California Irvine

OVERVIEW

Introduction

Deep Learning Software Packages

Hyperparameters

Exercises

Common problems

Notes

INTRODUCTION

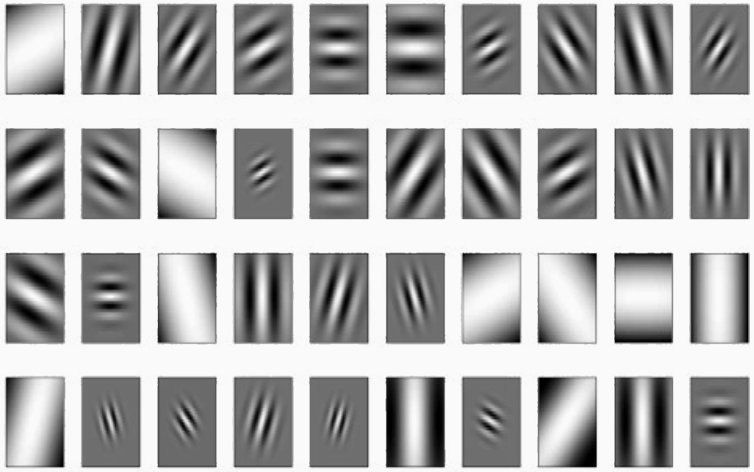
Why neural networks?

- Capacity to model complex functions.
- Adaptable to different types of machine learning problems.
- 'Efficient' training on large datasets.
- Empirical successes.

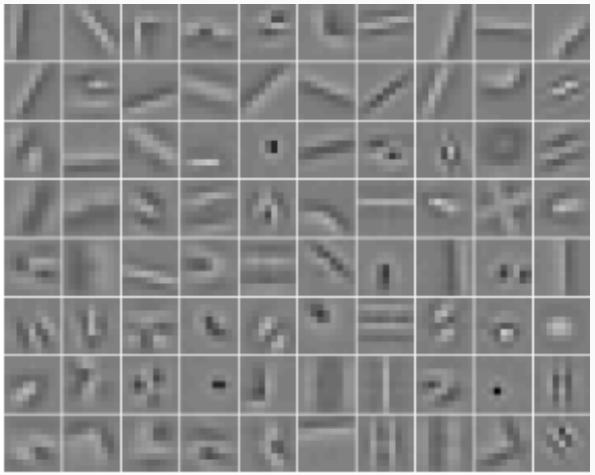
Why now?

- Improvements in computation (particularly GPUs).
- Better understanding of some of the tricky details.

Deep learning's promise:
Replacement of **engineered** features with **learned** features

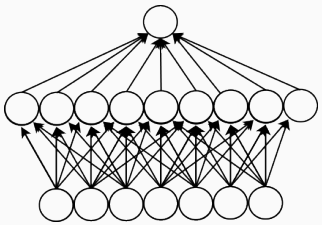


Engineered features

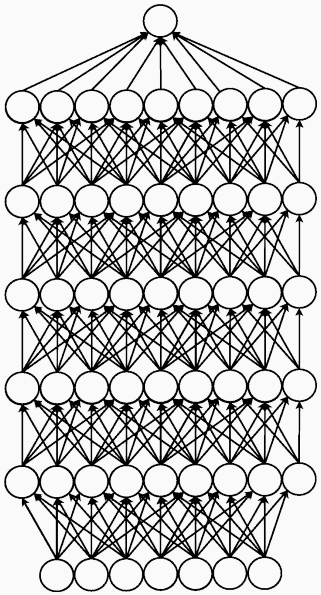


Learned features

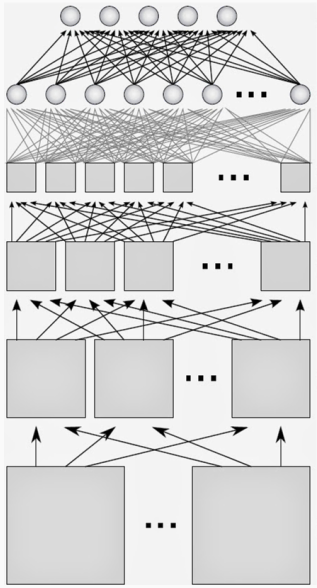
INTRODUCTION



Neural Network (NN)

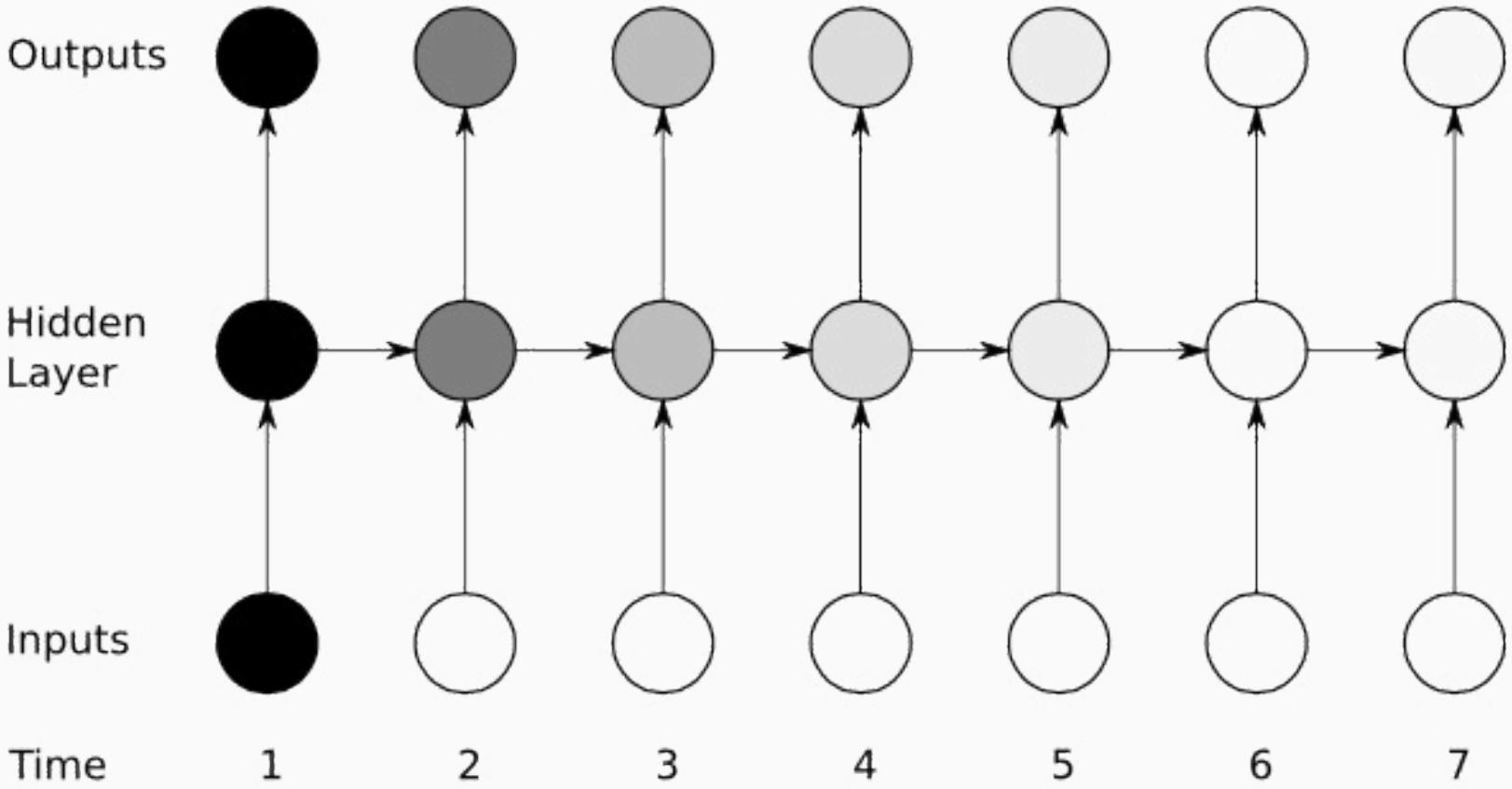


Deep NN



Convolutional NN

Recurrent neural network



DEEP LEARNING SOFTWARE PACKAGES

DEEP LEARNING SOFTWARE PACKAGES

What good deep learning software can provide:

- All the standard DL models and algorithms.
  - Recursive or convolutional architectures.
  - Weight decay, dropout, denoising.
- Symbolic representation and optimization.
- Architecture agnosticism (GPUs, parallelism, etc.)

It cannot provide (yet):

- Reasonable model architecture.
- Optimal learning algorithm hyperparameters.
- Out-of-the-box success.

Getting good results may take some work!

DEEP LEARNING SOFTWARE PACKAGES

Theano (Python)

- Underlies many popular deep learning tools:
  - Pylearn2
  - Blocks
  - Keras
  - Lasagne
  - nolearn

Torch (LuaJIT)

Caffe (C++)

Neon

Brainstorm

TensorFlow

All open source with permissive licenses, under active development.

Advantages of Theano:

- Computer algebra system with symbolic differentiation.
- Compiler will automatically optimize your computations for speed, numerical stability.
- Dynamic C code generation for speed.
- Support for latest NVIDIA CuDNN libraries.
- Integration with NumPy.
- Well-tested and well-documented.
- Large user community.
- Relatively easy installation.

Available on PyPI: pip install Theano

Advantages of Pylearn2:

- Built on Theano, but requires minimal familiarity.
- Well-tested and well-documented.

Put the following project in your python path:
git clone git://github.com/lisa-lab/pylearn2.git

DEEP LEARNING SOFTWARE PACKAGES

Summary of requirements for this tutorial:

- Theano (link)
- Download Pylearn2 (link) and include it in your PYTHONPATH.
- For visualizations: IPython notebooks (Jupyter), matplotlib
- For HDF5 files: H5PY

HYPERPARAMETERS

HYPERPARAMETERS

Specified by user:

- Training data
  - Representation (vector vs. image, etc).
  - Normalization
- Model
  - Layers: type, shape, number.
  - Activation/transfer functions.
  - Output type and loss function.
- Learning algorithm
  - Initialization.
  - Update rule.
  - Learning rate.
  - Learning rate decay.
  - Momentum.
  - Regularization (weight decay, dropout).
  - Stopping criteria.

Training data

- Vector vs. images vs. sequences.
- Pre-processing:
  - Standardization
  - Take log?
  - Whitening
- Precision
- Sparsity

HYPERPARAMETERS

Model

- Prediction tasks
- Architecture
- Activation function

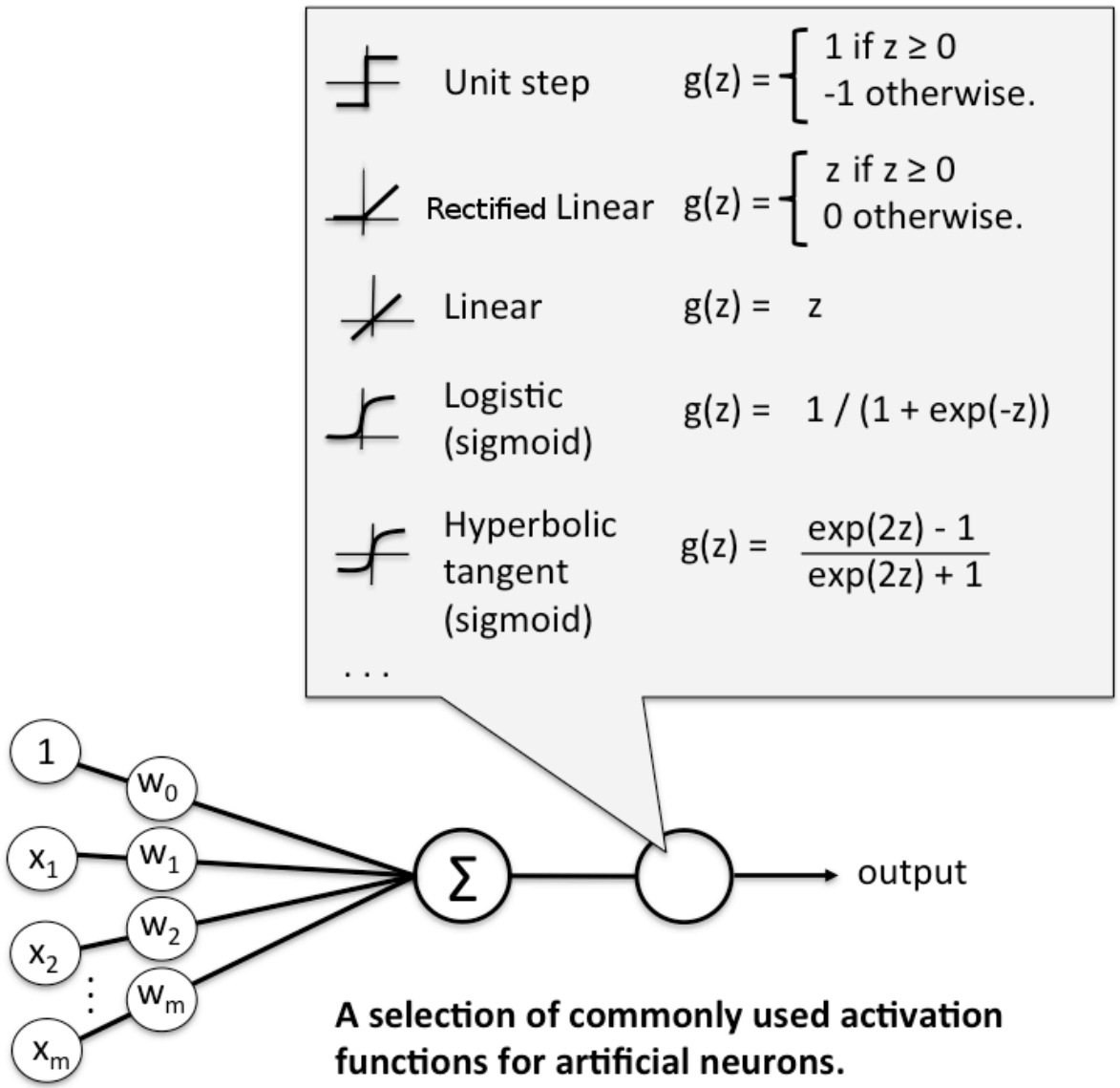
Prediction tasks

- **Classification:** For 2 classes, logistic neuron is used for the output of the network. For $n = 3+$ classes, output has n neurons with softmax activation function. In both cases, the loss function is cross-entropy, so this has a nice probabilistic interpretation.
- **Regression:** Output is real valued. Loss function is typically the squared error.
- **Autoencoders:** The target is the input itself. Network should have some kind of information bottleneck.

Structure and weight-sharing

- **Feed-forward:** Computation can be represented as a Directed Acyclic Graph.
- **Recursive:** Applying the same set of weights repeatedly in a tree structure.
- **Recurrent:** Network contains cycles.
- **Convolutional:** 'Filter' function applied repeatedly over one or more dimensions.

HYPERPARAMETERS



A selection of commonly used activation functions for artificial neurons.

Additional activation functions:

- Softmax
- Maxout
- Locally-competitive units
- Leaky ReLU
- Learned, parameterized activations

Weight Initialization

- Saturated neurons prevent information from flowing.
- Gradient diffusion or explosion.
- Good heuristics exist:
  - Glorot et al, 2010
  - Saxe et al, 2014

HYPERPARAMETERS

Learning algorithms / optimization:

- Stochastic gradient descent (SGD)
- Mini-batches
- Momentum
- Second-order methods
- Conjugate gradients
- Adagrad, RMSProp, ADAM, etc.

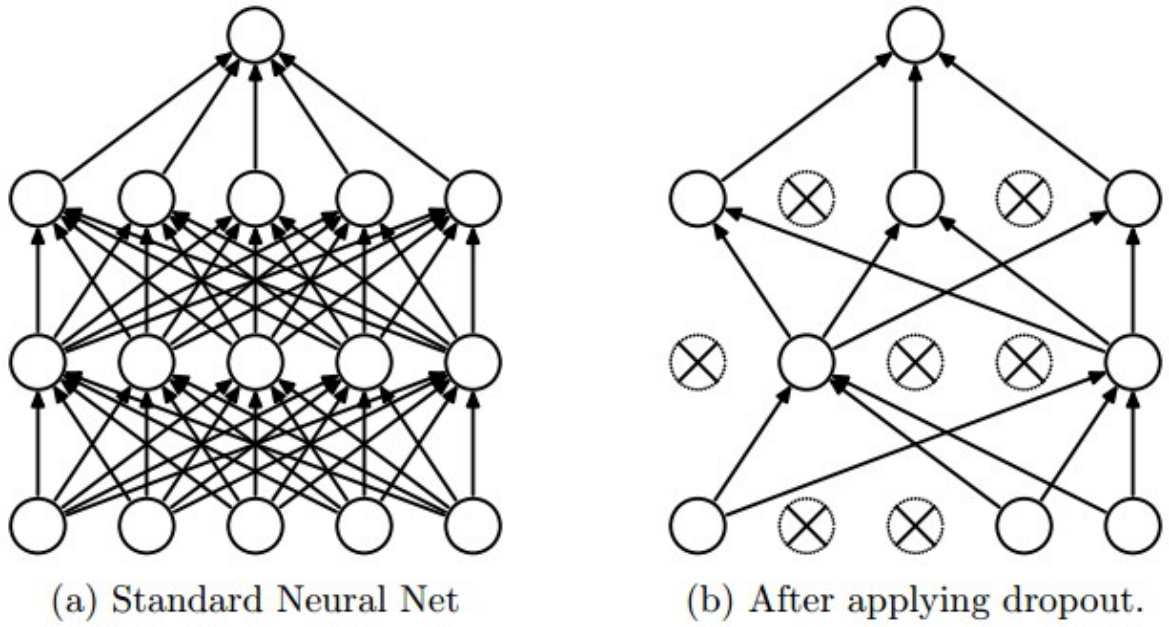
HYPERPARAMETERS

Typically: We use SGD on mini-batches, setting a modest initial value for the learning rate, then decreasing it over time by a fixed factor (or manually).

Momentum Use initial burn in with small amount of momentum. Often increase to 0.99-0.999.

Regularization

- Weight-decay (L2 penalty on large weights)
- Max column norms
- Dropout

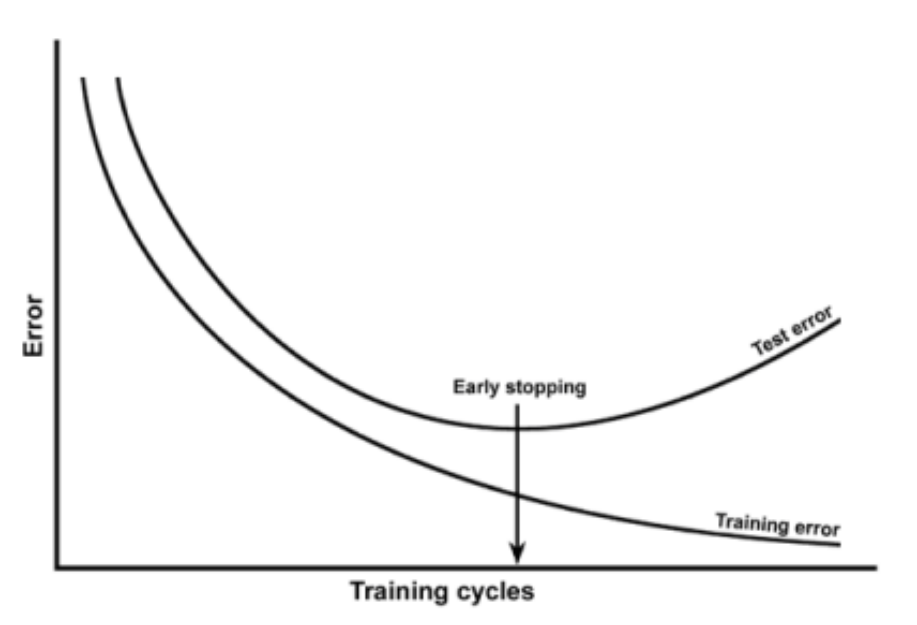


(a) Standard Neural Net

(b) After applying dropout.

Stopping criteria considerations:

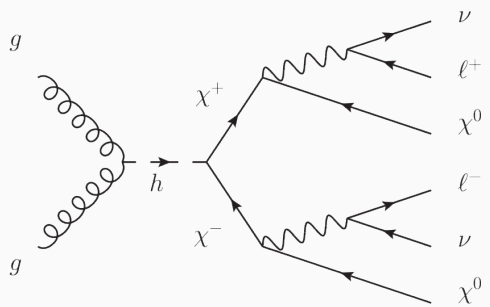
- Overfitting.
- Training error convergence.
- Learning rate and momentum schedules.



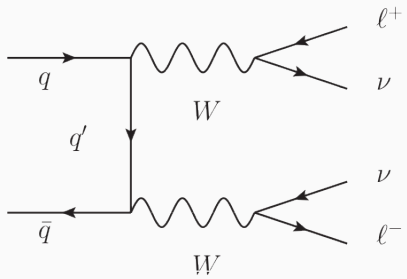
EXERCISES

EXERCISE 1

Exotic particle search from supersymmetry:



Process with hypothetical supersymmetric particles



Background with W bosons

Classification problem with 6 million simulated examples.
Full data set at: https://archive.ics.uci.edu/ml/datasets/SUSY

From Baldi et al 2014

EXERCISE 2

Classification of hand-written digits (0-9) from 28x28 pixel greyscale images (MNIST data set).
Full data set of 70k examples: http://yann.lecun.com/exdb/mnist/

From LeCun et al

COMMON PROBLEMS

COMMON PROBLEMS

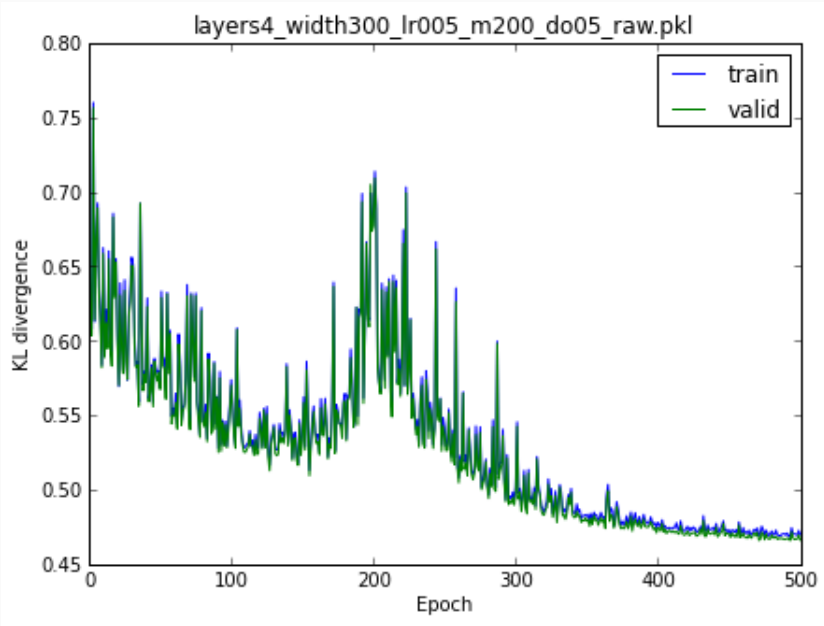
Obvious problems:

- Overfitting.
- Underfitting.
- Wrong objective function.
- Weights explode.

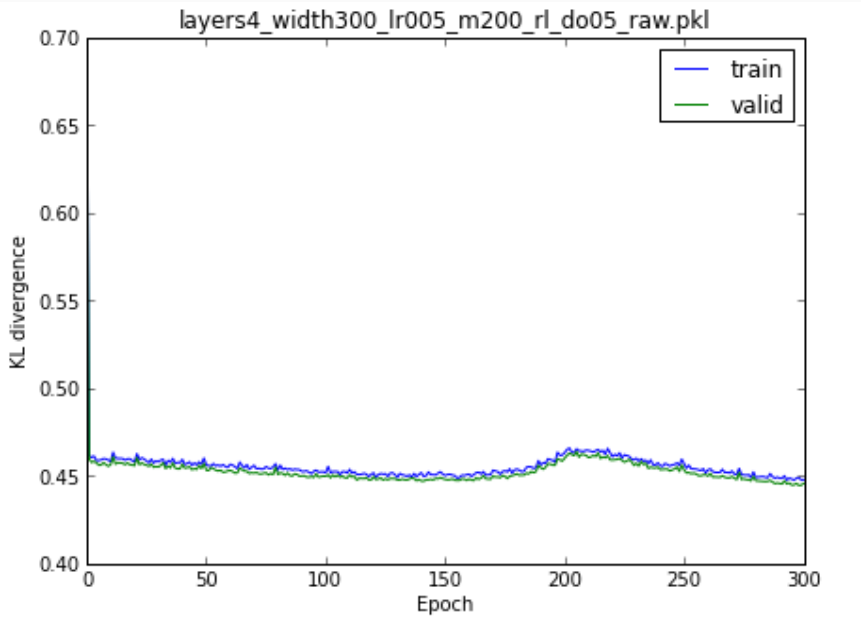
Non-obvious problems:

- Bad weight initialization.
- Gradient diffusion.
- Learning rate or momentum too large.

COMMON PROBLEMS

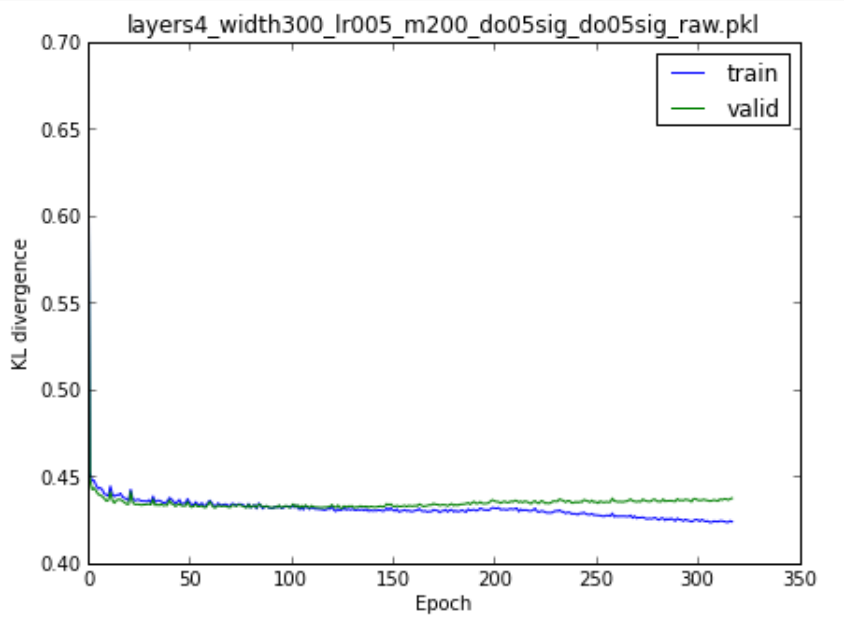


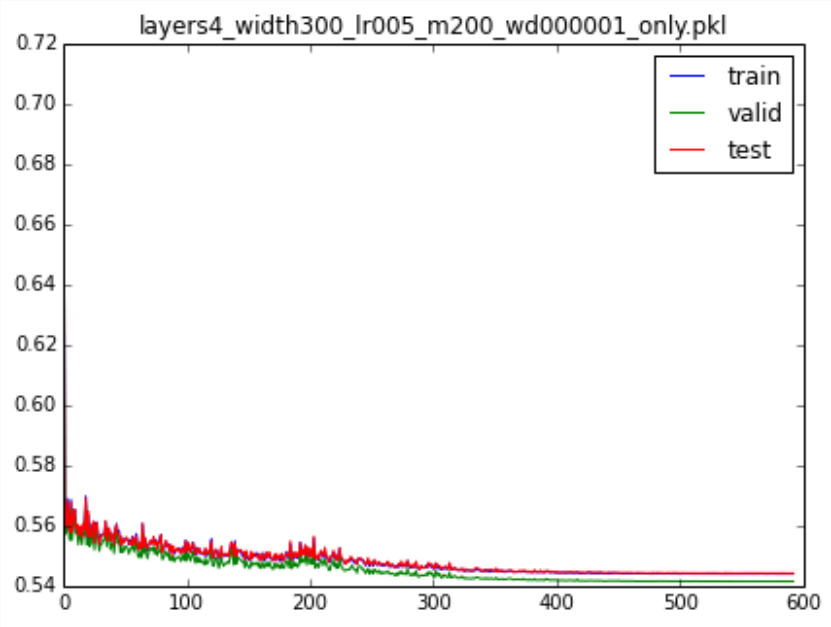
Learning rate too large.



Too much momentum.

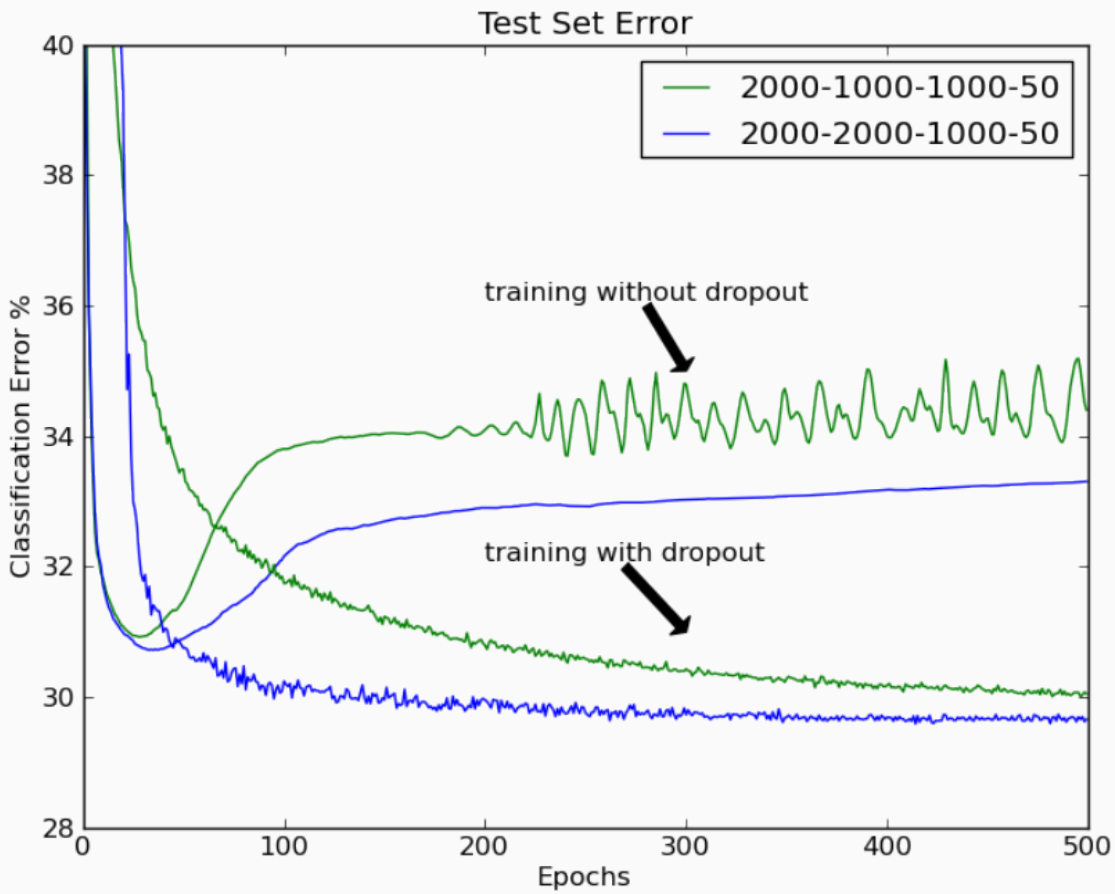
COMMON PROBLEMS





Good learning rate, good decay.

COMMON PROBLEMS



Fast training is not necessarily better.

From Hinton et al 2012

COMMON PROBLEMS

Training error increases: Debug by decreasing learning rate to a very small value.

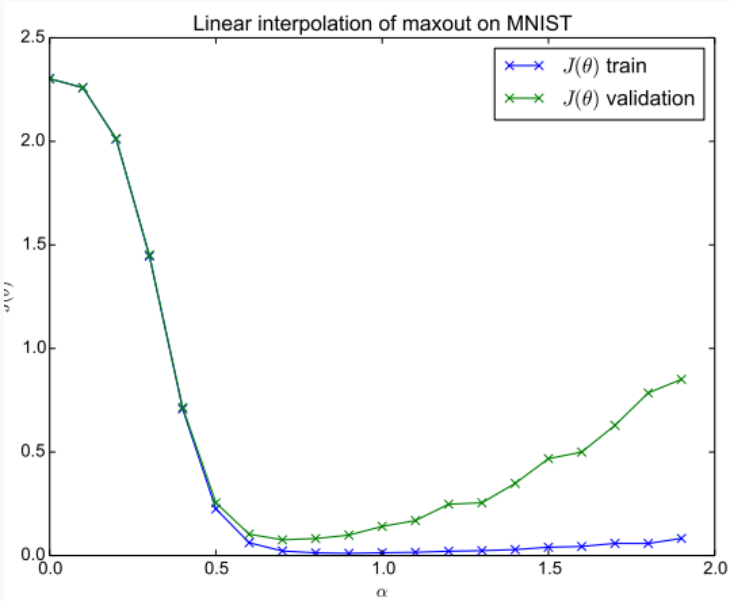
Exploding weights: If data set is small or neurons are linear, the weights may grow without bound. Use weight decay or a max column norm constraint.

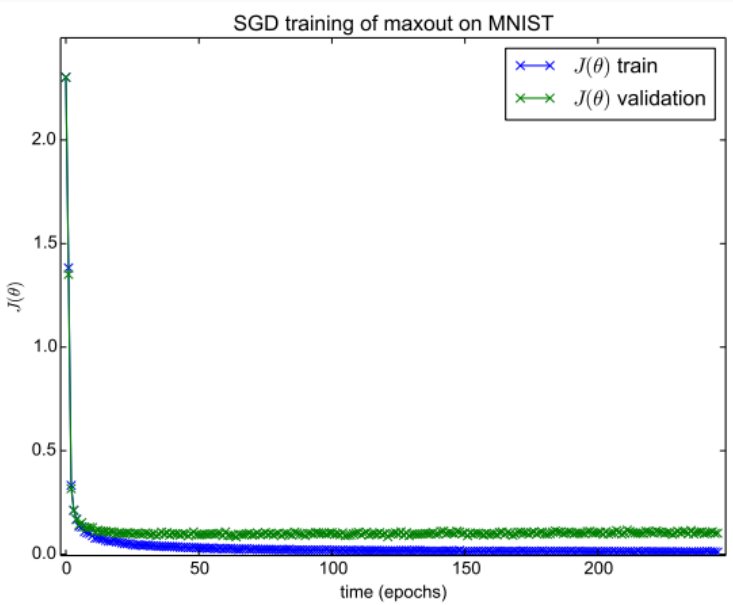
Adding training data reduces performance: Check learning rate and momentum schedule.

NOTES

A NOTE ON OPTIMIZATION

Local minima are not a primary concern. See Goodfellow et al 2015, Qualitatively Characterizing Nueral Network Optimization Problems





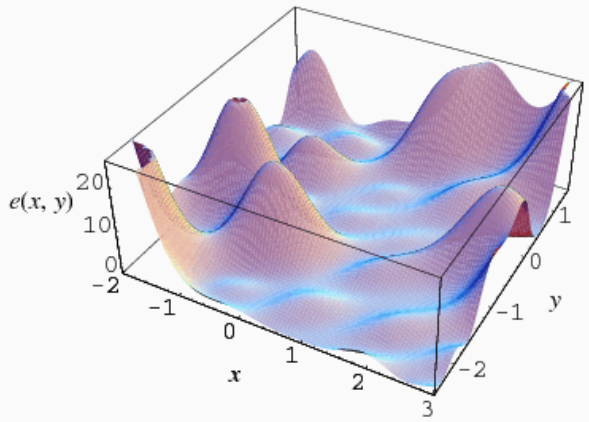
HYPERPARAMETER OPTIMIZATION

Traditionally:

- Grid search
- ”Graduate Student Descent”

Other options:

- Random search
- Bayesian optimization with Gaussian Processes (e.g. Spearmint) for continuous hyperparameters
- Active learning: http://arxiv.org/abs/1502.07943



INTRODUCTION