

Part 3

Practice, Issues, Questions

Deep Learning Tricks of the Trade

- Y. Bengio (2012), “Practical Recommendations for Gradient-Based Training of Deep Architectures”
 - Unsupervised pre-training
 - Stochastic gradient descent and setting learning rates
 - Main hyper-parameters
 - Learning rate schedule
 - Early stopping
 - Minibatches
 - Parameter initialization
 - Number of hidden units
 - L1 and L2 weight decay
 - Sparsity regularization
 - Debugging
 - How to efficiently search for hyper-parameter configurations

Stochastic Gradient Descent (SGD)

- Gradient descent uses total gradient over all examples per update, SGD updates after only 1 or few examples:

$$\theta^{(t)} \leftarrow \theta^{(t-1)} - \epsilon_t \frac{\partial L(z_t, \theta)}{\partial \theta}$$

- L = loss function, z_t = current example, θ = parameter vector, and ϵ_t = learning rate.
- Ordinary gradient descent is a batch method, very slow, **should never be used**. 2nd order batch methods are being explored as an alternative but SGD with selected learning schedule remains the method to beat.

Learning Rates

- Simplest recipe: keep it fixed and use the same for all parameters.
- Collobert scales them by the inverse of square root of the fan-in of each neuron
- Better results can generally be obtained by allowing learning rates to decrease, typically in $O(1/t)$ because of theoretical convergence guarantees, e.g.,

$$\epsilon_t = \frac{\epsilon_0 \tau}{\max(t, \tau)}$$

with hyper-parameters ϵ_0 and τ .

Early Stopping

- Beautiful **FREE LUNCH** (no need to launch many different training runs for each value of hyper-parameter for #iterations)
- Monitor validation error during training (after visiting # examples a multiple of validation set size)
- Keep track of parameters with best validation error and report them at the end
- If error does not improve enough (with some patience), stop.

Long-Term Dependencies

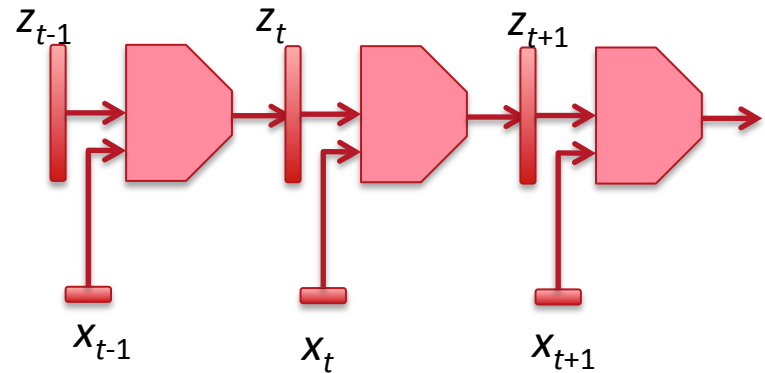


- In very deep networks such as **recurrent networks** (or possibly recursive ones), the gradient is a product of Jacobian matrices, each associated with a step in the forward computation. This can become very small or very large quickly [Bengio et al 1994], and the locality assumption of gradient descent breaks down.

$$L = L(s_T(s_{T-1}(\dots s_{t+1}(s_t, \dots))))$$
$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \dots \frac{\partial s_{t+1}}{\partial s_t}$$

- Two kinds of problems:
 - sing. values of Jacobians $> 1 \rightarrow$ gradients explode
 - or sing. values $< 1 \rightarrow$ gradients shrink & vanish

Long-Term Dependencies and Clipping Trick



Trick first introduced by Mikolov is to clip gradients to a maximum NORM value.

Makes a big difference in Recurrent Nets. Allows SGD to compete with HF optimization on difficult long-term dependencies tasks. Helped to beat SOTA in text compression, language modeling, speech recognition.

Normalized Initialization to Achieve Unity-Like Jacobian

Assuming $f'(act=0)=1$

To keep information flowing in both direction we would like to have the following properties.

- **Forward-propagation:**

$$\forall(i, i'), Var[z^i] = Var[z^{i'}] \Leftrightarrow \forall i, n_i Var[W^i] = 1$$

- **Back-propagation:**

$$\forall(i, i'), Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^{i'}}\right] \Leftrightarrow \forall i, n_{i+1} Var[W^i] = 1$$

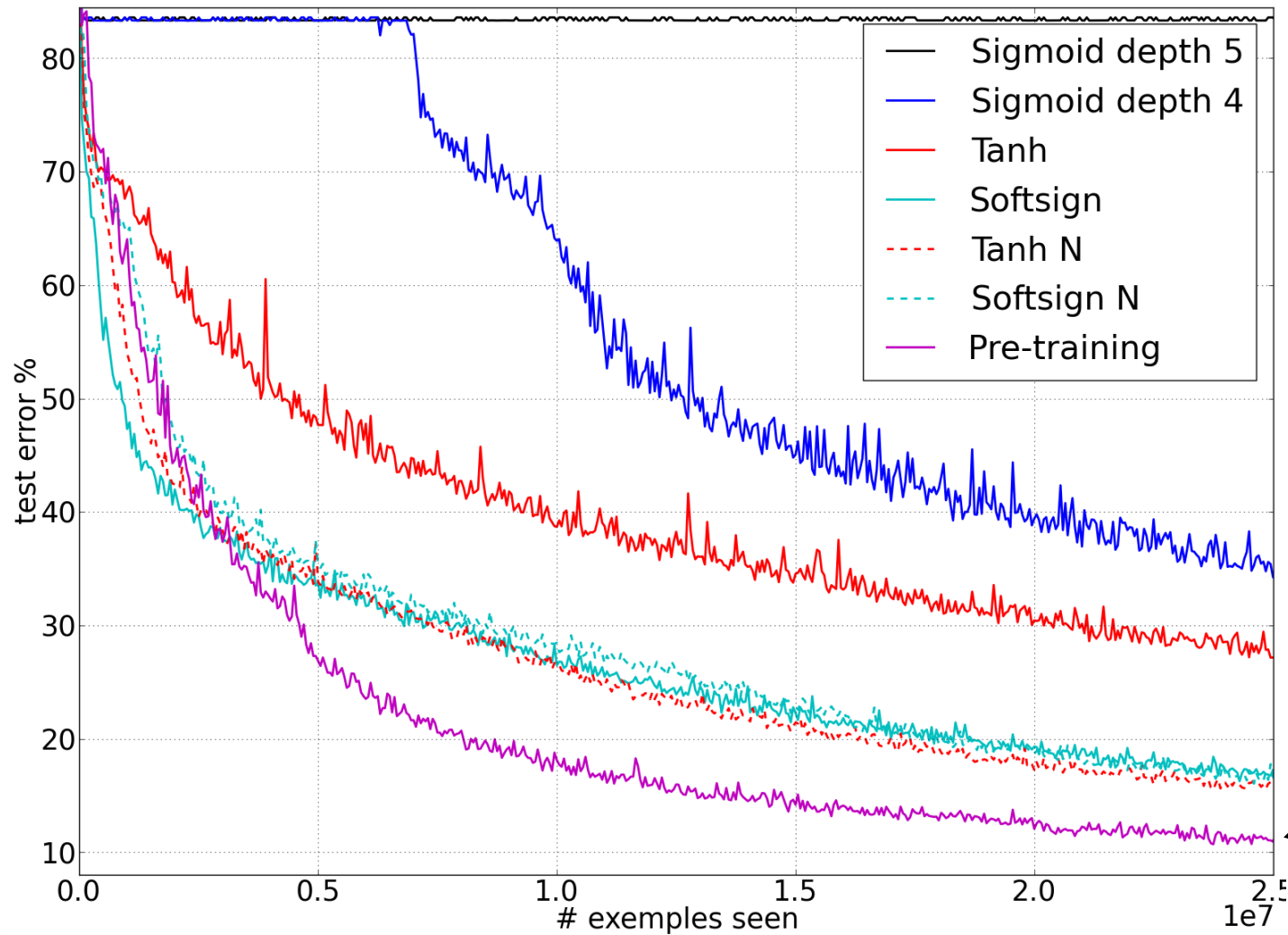
Possible compromise:

$$\forall i, Var[W^i] = \frac{2}{n_i + n_{i+1}} \quad (4)$$

This gives rise to proposed **normalized initialization** procedure:

$$W^i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}\right] \quad (5)$$

Normalized Initialization with Variance-Preserving Jacobians



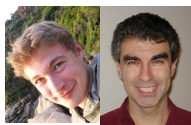
Parameter Initialization

- Initialize hidden layer biases to 0 and output (or reconstruction) biases to optimal value if weights were 0 (e.g. mean target or inverse sigmoid of mean target).
- Initialize weights $\sim \text{Uniform}(-r, r)$, r inversely proportional to fan-in (previous layer size) and fan-out (next layer size):

$$\sqrt{6 / (\text{fan-in} + \text{fan-out})}$$

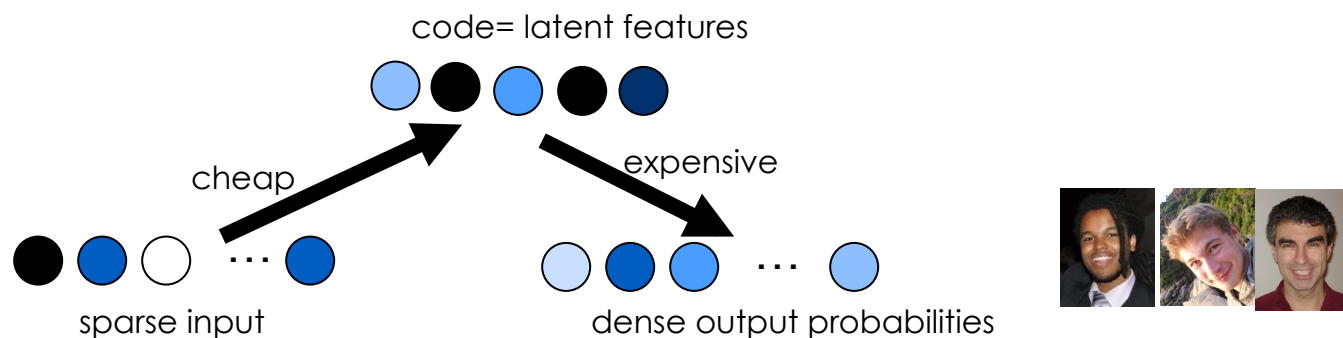
for tanh units (and 4x bigger for sigmoid units)

(Glorot & Bengio AISTATS 2010)



Handling Large Output Spaces

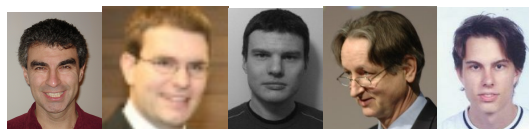
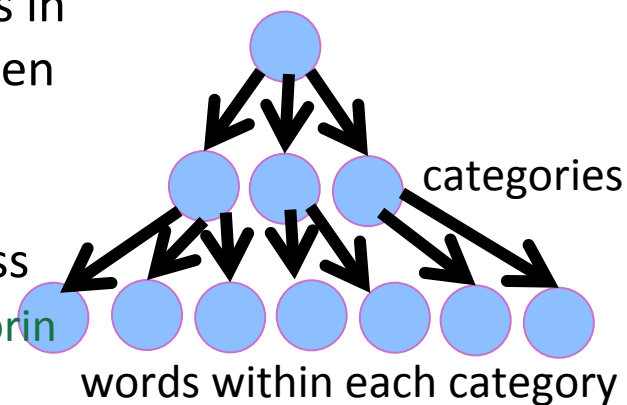
- Auto-encoders and RBMs reconstruct the input, which is sparse and high-dimensional; Language models have huge output space.



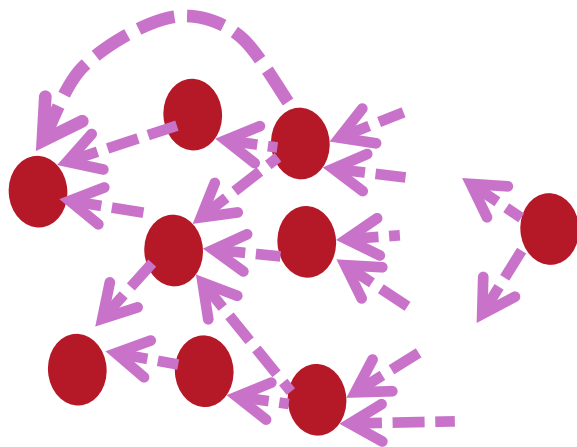
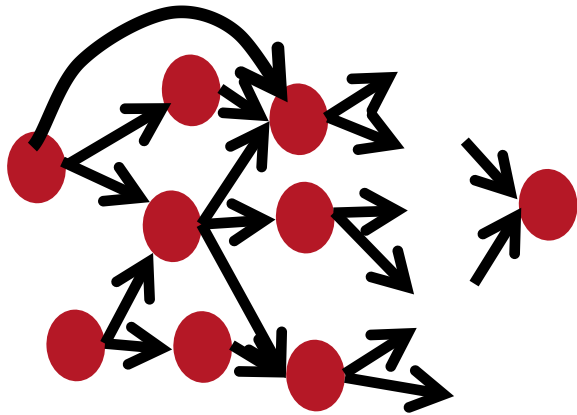
- (Dauphin et al, ICML 2011) Reconstruct the non-zeros in the input, and reconstruct as many randomly chosen zeros, + importance weights



- (Collobert & Weston, ICML 2008) sample a ranking loss
- Decompose output probabilities hierarchically (Morin & Bengio 2005; Blitzer et al 2005; Mnih & Hinton 2007,2009; Mikolov et al 2011)



Automatic Differentiation



- The gradient computation can be automatically inferred from the symbolic expression of the fprop.
- Makes it easier to quickly and safely try new models.
- Each node type needs to know how to compute its output and how to compute the gradient wrt its inputs given the gradient wrt its output.
- Theano Library (python) does it symbolically. Other neural network packages (Torch, Lush) can compute gradients for any given run-time value.

(Bergstra et al SciPy'2010)



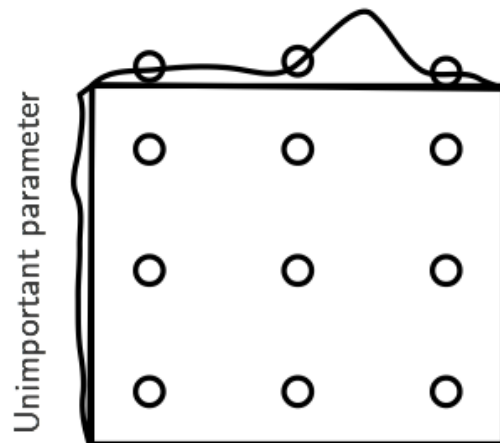
Random Sampling of Hyperparameters

(Bergstra & Bengio 2012)



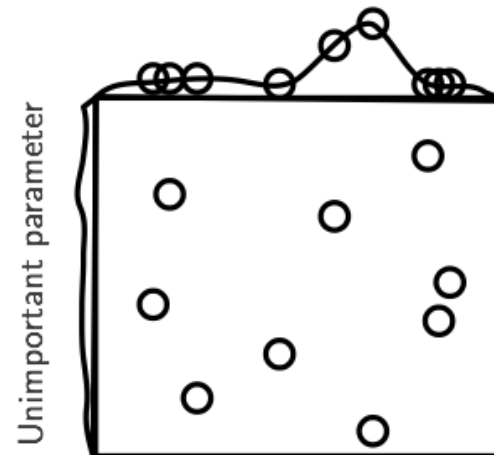
- Common approach: manual + grid search
- Grid search over hyperparameters: simple & wasteful
- Random search: simple & efficient
 - Independently sample each HP, e.g. $\text{l.rate} \sim \exp(\text{U}[\log(.1), \log(.0001)])$
 - Each training trial is iid
 - If a HP is irrelevant grid search is wasteful
 - More convenient: ok to early-stop, continue further, etc.

Grid Layout



Important parameter

Random Layout



Important parameter

Issues and Questions

Why is Unsupervised Pre-Training Working So Well?

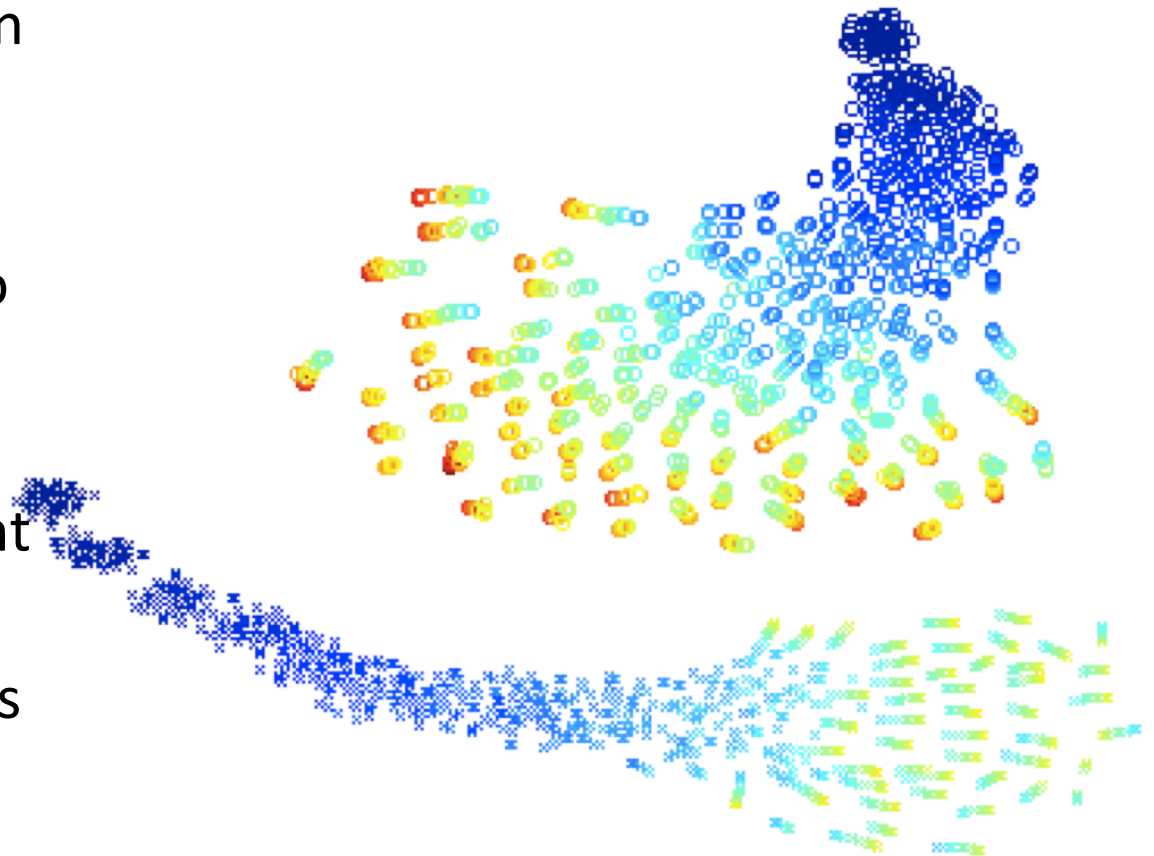
- Regularization hypothesis:
 - Unsupervised component forces model close to $P(x)$
 - Representations good for $P(x)$ are good for $P(y|x)$
- Optimization hypothesis:
 - Unsupervised initialization near better local minimum of $P(y|x)$
 - Can reach lower local minimum otherwise not achievable by random initialization
 - Easier to train each layer using a layer-local criterion



(Erhan et al JMLR 2010)

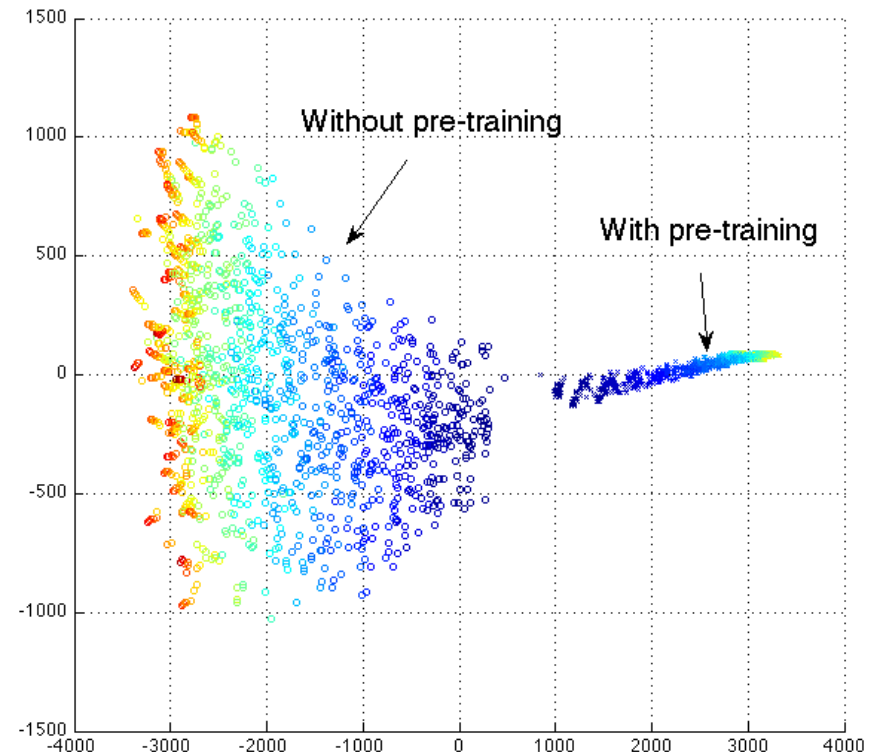
Learning Trajectories in Function Space

- Each point a model in function space
- Color = epoch
- Top: trajectories w/o pre-training
- Each trajectory converges in different local min.
- No overlap of regions with and w/o pre-training



Learning Trajectories in Function Space

- Each trajectory converges in different local min.
- With ISOMAP, try to preserve geometry: pretrained nets converge near each other (less variance)
- Good answers = worse than a needle in a haystack (learning dynamics)



Inference Challenges

- Many latent variables involved in understanding language (sense ambiguity, parsing, semantic role)
- Almost any inference mechanism can be combined with deep learning
- See [Bottou, Bengio, LeCun 97], [Graves 2012]



- Complex inference can be hard (exponentially) and needs to be approximate → learn to perform inference

Dealing with Inference

- $P(h|x)$ in general intractable (e.g. non-RBM Boltzmann machine)
- But explaining away is nice
- Approximations
 - Variational approximations, e.g. see Goodfellow et al ICML 2012 (assume a unimodal posterior)
 - MCMC, but certainly not to convergence
- We would like a model where approximate inference is going to be a good approximation
 - Predictive Sparse Decomposition does that
 - Learning approx. sparse decoding (Gregor & LeCun ICML'2010)
 - Estimating $E[h|x]$ in a Boltzmann with a separate network (Salakhutdinov & Larochelle AISTATS 2010)

Dealing with a Partition Function

- $Z = \sum_{x,h} e^{-\text{energy}(x,h)}$
- Intractable for most interesting models
- MCMC estimators of its gradient
- Noisy gradient, can't reliably cover (spurious) modes
- Alternatives:
 - Score matching (Hyvarinen 2005)
 - Noise-contrastive estimation (Gutmann & Hyvarinen 2010)
 - Pseudo-likelihood
 - Ranking criteria (wsabie) to sample negative examples (Weston et al. 2010)
 - Auto-encoders?

Score Matching

(Hyvarinen 2005)

- Score of model p : $d \log p(\mathbf{x})/d\mathbf{x}$ **does not contain partition fn Z**

- Matching score of p to target score: **?**

$$\mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \left\| \frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} - \frac{\partial \log q(\mathbf{x})}{\partial \mathbf{x}} \right\|^2 \right]$$

- Hyvarinen shows it equals

$$\mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{2} \left\| \frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} \right\|^2 + \sum_i \frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{x}_i^2} \right] + \text{const}$$

- and proposes to minimize corresponding empirical mean
- Shown to be asymptotically unbiased to estimate parameters
- Note: for GRBM, 1st term is squared reconstruction error and 2nd term looks like contractive penalty $\sum_{ij} h_i(1-h_i) W_{ij}^2$



Denoising Auto-Encoders doing Score Matching on Gaussian RBMs

- clean input - corrupted input = direction of increasing log-likelihood

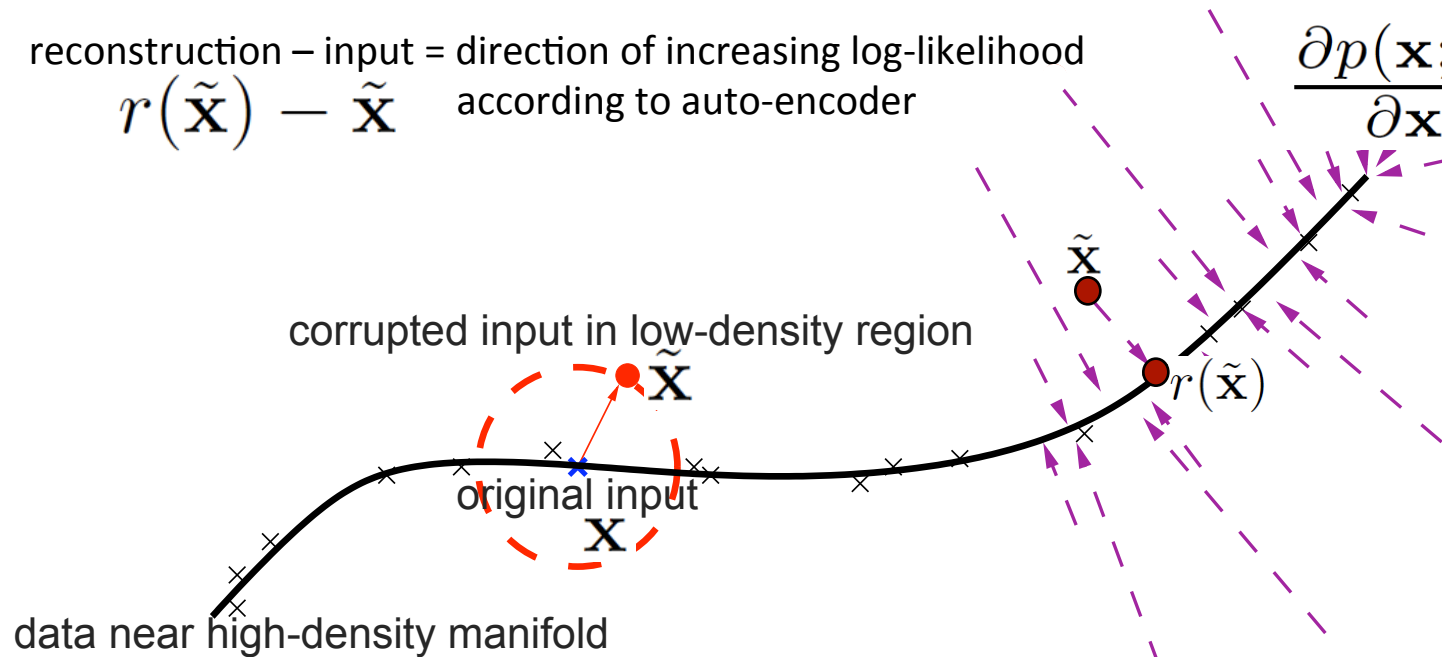
$$\mathbf{x} - \tilde{\mathbf{x}}$$

$$\frac{\partial \log q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}$$

- reconstruction - input = direction of increasing log-likelihood according to auto-encoder

$$r(\tilde{\mathbf{x}}) - \tilde{\mathbf{x}}$$

$$\frac{\partial p(\mathbf{x};\theta)}{\partial \mathbf{x}}$$



- Denoising error = $\| (r(\tilde{\mathbf{x}}) - \tilde{\mathbf{x}}) - (\mathbf{x} - \tilde{\mathbf{x}}) \|^2 = \| r(\tilde{\mathbf{x}}) - \mathbf{x} \|^2$

Denoising Auto-Encoders doing Score Matching on Gaussian RBMs

- Gaussian DAE reconstruction:

$$r(\tilde{\mathbf{x}}) = \mathbf{W}^T h(\tilde{\mathbf{x}}) + \mathbf{c} = \mathbf{W}^T \text{sigm}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{c}$$

- Corresponds to gradient of free energy of Gaussian RBM
- Any (free) energy function with x^2 term gives rise to score function that can be written as proportional to $r(x)-x$ (=residual). **Recent research shows this is true for any energy fn**
- Not all DAEs have reconstruction residual = a derivative (most previous DAEs with binomial KL divergence reconstruction error)
- See also ([Swersky 2010](#)), thesis on link between ordinary auto-encoder reconstruction error & Score Matching

Contrastive Sampling of Negative Examples

- (Collobert & Weston ICML 2008)
- (Bordes et al AAI 2011, AISTATS 2012)
- Similar to wsabie (Weston et al, MLJ 2010)

$$\sum_{x \in \mathcal{D}} \sum_{\tilde{x} \sim Q(\tilde{x}|x)} \max(\mathcal{E}(x) - \mathcal{E}(\tilde{x}) + 1, 0)$$

Positive example Negative example

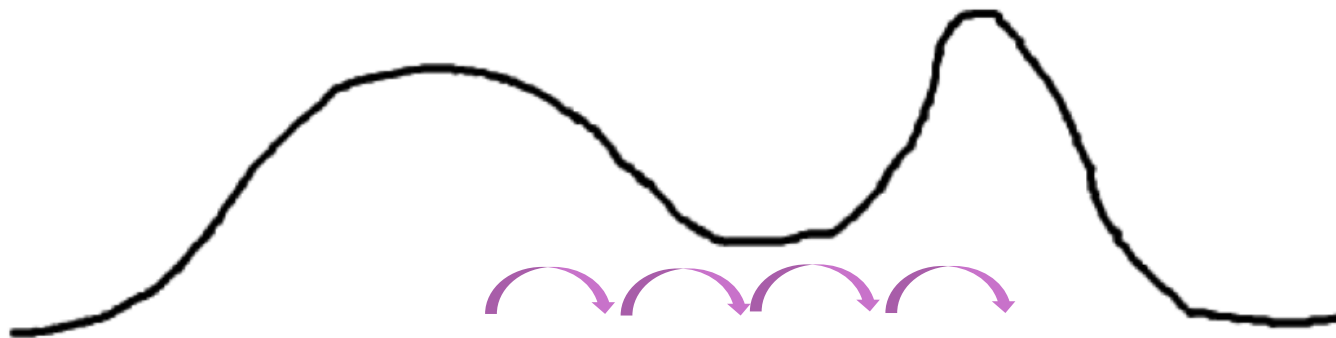
pull down push up

energy function

- In those cases, the negative example \tilde{x} is obtained by uniformly sampling one of the elements of \bar{x} and keeping the rest fixed.

For gradient & inference: More difficult to mix with better trained models

- Early during training, density smeared out, mode bumps overlap



- Later on, hard to cross empty voids between modes



Are we doomed if
we rely on MCMC
during training?
Will we be able to
train really large &
complex models?

Poor Mixing: Depth to the Rescue

- Deeper representations can yield some disentangling
- Hypotheses:
 - more abstract/disentangled representations unfold manifolds and fill more the space
 - can be exploited for better mixing between modes
 - E.g. reverse video bit, class bits in learned object representations: easy to Gibbs sample between modes at

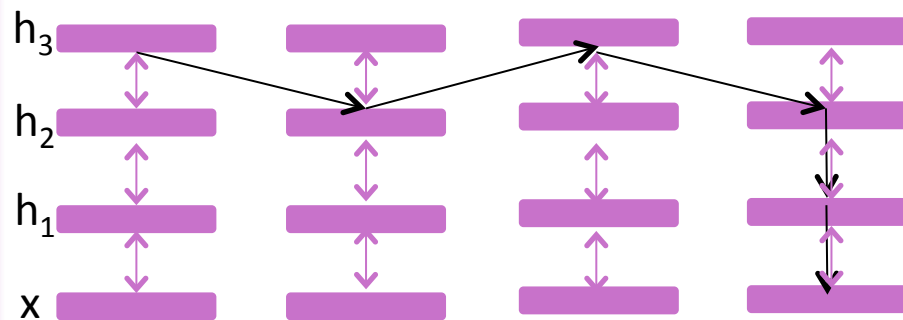
Layer abstract level



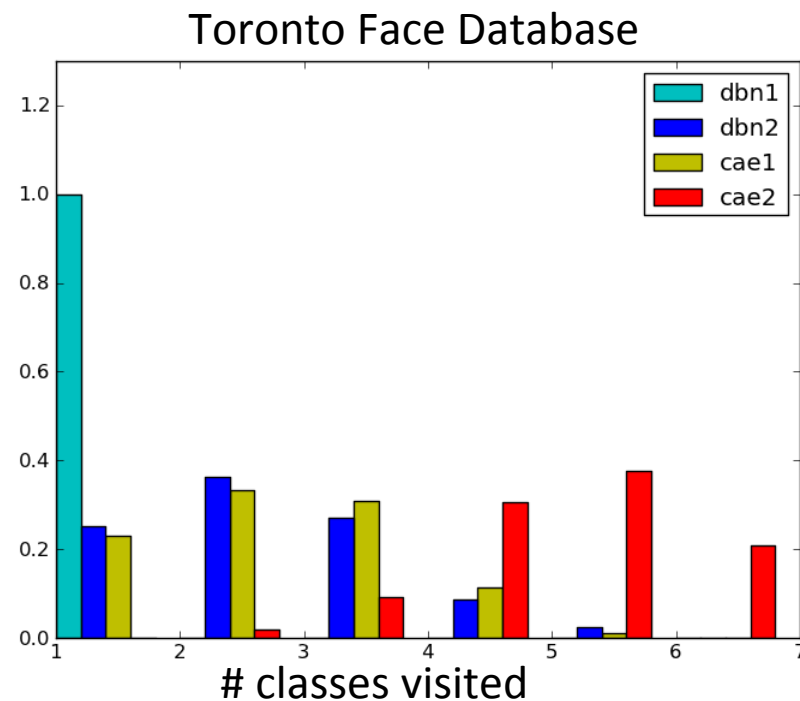
Points on the interpolating line between two classes, at different levels of representation

Poor Mixing: Depth to the Rescue

- Sampling from DBNs and stacked Contrastive Auto-Encoders:
 1. MCMC sample from top-level singler-layer model
 2. Propagate top-level representations to input-level repr.
- Visits modes (classes) faster



192



Regularized AE: MCMC Miracles?

- Virtually no burn-in waste with Denoising AE, trained to map random configurations to plausible ones in 1 step (very few necessary in practice)
- Tempering-like effect by controlling step size σ (in manifold directions) trades off mixing speed with accuracy (more math needed here!)
- Fast mode mixing if sampling at higher levels

Other reasons why regularized auto-encoders are interesting alternatives

- Easy “inference” (can have iterative inference with lateral connections, but not considered as an approximation to the right thing)
 - No partition function (and associated approximations)
 - No negative feedback loop between sampling and learning
- **do we actually need an explicit probabilistic model?**

More Open Questions

- What is a good representation? Disentangling factors? Can we design better training criteria / setups?
- Can we safely assume $P(h|x)$ to be unimodal or few-modal? If not, is there any alternative to explicit latent variables?
- Should we have explicit explaining away or just learn to produce good representations? (possibly iteratively)
- Should learned representations be low-dimensional or sparse/saturated and high-dimensional?
- Why is it more difficult to optimize deeper (or recurrent/recursive) architectures? Does it necessarily get more difficult as training progresses? Can we do better?