Problem Proposal: Machine learning mainly facing challenges of unsupervised learning.

Linear factor Models

Here we propose term linear factor model describing the mapping between latent variables and observed values.

- Probabilistic PCA and Factor Analysis (Discussed in PRML)
- Independent Component Analysis (ICA)
  - Aimed to embed a big signal into several small underlying signals.
  - Different variant of ICA aims on different tasks: $p(x)$? $p(h)$? or W.
- Slow feature analysis

  SFA aims to find the most stable feature along the time, by using linear transformation. It can be done by adding a term on the cost function:

  $$\lambda \sum_t L(f(x^{(t+1)}), f(x^{(t)}))$$ → Hope they resemble to each other

  To ensure the feature learned to be independent (thus meaningful), we often adding constrain:

  $$\forall i<j \quad \mathbb{E}_t[f(x^{(t)})_i f(x^{(t)})_j] = 0$$ → Satisfied when i, j are independent.

  - An advantage of SFA: When we know about the environment dynamics, we can theoritically make predictions, because SFA can explain time slow feature.
  - Maybe the slow prior is too strong, SFA is currently not working.
- Sparse Coding

  $p(x|h) = \mathcal{N}(x \mid Wh+b, \beta^{-1} \cdot I)$, given two kind of prior to make the distribution of h more sparse:

  $$p(h_i) = \text{Laplace}(h_i \mid 0, \tfrac{2}{\lambda}) = \frac{\lambda}{4} e^{-\frac{1}{2}\lambda |h_i|}$$ — we often see absolut value for sparsity

  - sparse coding can take advantage of non-parametric method to make less generalization error.

- The base (k) of sparse coding is always a larger value (greater than n), not like PCA.
- Manifold Interpretation of PCA

  Concept of learning a manifold, learning a shape indicating the data distribution.

Autoencoders

Undercomplete Autoencoders: Only used for dimensionality reduction, failed when the decoder is too strong (remember all the training data)

Regularized Autoencoders:

Challenge: How we design learning algorithms and model structure to let it learn valuable information.

- Sparse Autoencoders

  Adding sparse regularizer to h, the calculate the joint distribution or the likelihood (note the term (likelihood used in this book) $P(h|x)$.

  $$\log P_{model}(h,x) = \underline{\log P_{model}(h)} + \log P_{model}(x|h)$$

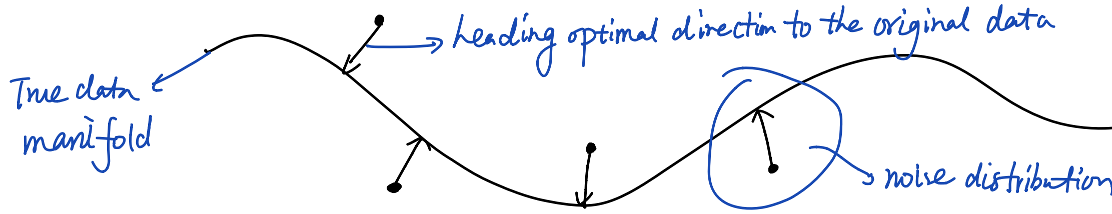
  $P_{model}(h_i) = \frac{\lambda}{2} e^{-\lambda|h_i|}$ (sometimes acted as a regularizer)

- Denoising Autoencoder

  Minimizing $L(x, g(f(\tilde{x})))$ $\tilde{x}$ is a noisy version of x

  Adding noisy input and we want to rebuild noise-free input.

  - normally the noise form will be $C(\tilde{x}|x) \sim N(\tilde{x}|x)$, then sampling from that distribution to get $\tilde{x}$.
  - Indeed, any cost function / noise add-on can be related to an optimal solution of maximizing the matching score would be used here.



- Regularizing by Penalizing Derivatives

$$\Omega(h, x) = \lambda \sum_i \|\nabla_x h_i\|^2$$

Make it not so sensitive to small changes of x.

Representational Power, Layer Size and depth: In practice, we prefer deeper NN with smaller size per layer, also we can train a neural network auto encoder from shallow to depth.

Stochastic Encoders and Decoders: We can optimize autoencoder as we optimize neural networks

Learning manifold with Autoencoders:

Two force that ensure the success of autoencoder:

① Reconstruction of input / training data.

② Regularization of not being so sensitive to input data. (Invariant)

- Different from non-parametric method, auto encoders learns manifold of data, or equivalently, tagent line of the manifold. It is invariant of small locally pertubation but sensitive of between class difference.

Contractive Autoencoders

Adding penalty for large encoding derivatives:

$$\Omega(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

Can be trained layer by layer and bounding $\underline{f}$ and g

"Contractive" termed from linear operation, where Jacobain is limited to be small.