Deep Learning

Lecture 10

Fernando Perez-Cruz
based on Thomas Hofmann lectures

Swiss Data Science Center
ETH Zurich and EPFL – `datascience.ch`

November 26, 2018

Overview

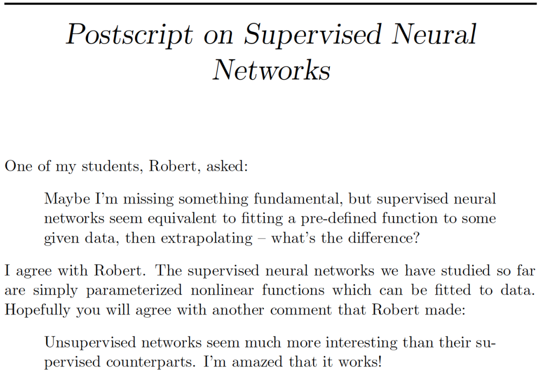
1. Density Estimation

2. Autoencoders

3. Factor Analysis

4. Latent Variable Models

From David MacKay's 2004 book:



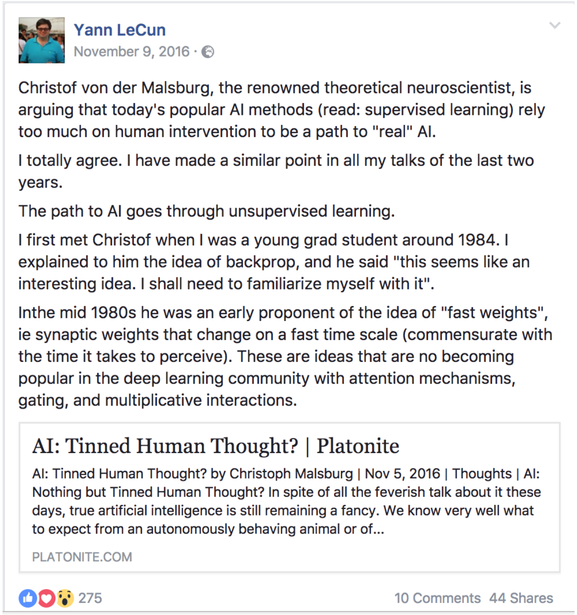
Postscript on Supervised Neural Networks

One of my students, Robert, asked:

> Maybe I'm missing something fundamental, but supervised neural networks seem equivalent to fitting a pre-defined function to some given data, then extrapolating – what's the difference?

I agree with Robert. The supervised neural networks we have studied so far are simply parameterized nonlinear functions which can be fitted to data. Hopefully you will agree with another comment that Robert made:

> Unsupervised networks seem much more interesting than their supervised counterparts. I'm amazed that it works!



Yann LeCun
November 9, 2016 ·

Christof von der Malsburg, the renowned theoretical neuroscientist, is arguing that today's popular AI methods (read: supervised learning) rely too much on human intervention to be a path to "real" AI.

I totally agree. I have made a similar point in all my talks of the last two years.

The path to AI goes through unsupervised learning.

I first met Christof when I was a young grad student around 1984. I explained to him the idea of backprop, and he said "this seems like an interesting idea. I shall need to familiarize myself with it".

Inthe mid 1980s he was an early proponent of the idea of "fast weights", ie synaptic weights that change on a fast time scale (commensurate with the time it takes to perceive). These are ideas that are no becoming popular in the deep learning community with attention mechanisms, gating, and multiplicative interactions.

AI: Tinned Human Thought? | Platonite

AI: Tinned Human Thought? by Christoph Malsburg | Nov 5, 2016 | Thoughts | AI: Nothing but Tinned Human Thought? In spite of all the feverish talk about it these days, true artificial intelligence is still remaining a fancy. We know very well what to expect from an autonomously behaving animal or of...

PLATONITE.COM

275 10 Comments 44 Shares

Section 1

Density Estimation

Density Estimation

- Density estimation:
  - standard problem in statistics and unsupervised learning
  - learn the distribution of the data
- Classically: parametric family of densities

$$p_\theta, \quad \theta \in \Theta$$

- MLE (maximum likelihood estimation)

$$\theta^* \xrightarrow{\max} \mathbf{E}[\ln p_\theta(\mathbf{x})], \quad \mathbf{x} \sim p(\mathbf{x}).$$

  - in practice: expectation w.r.t. empirical distribution

Prescribed Models

- Prescribed model definition
  - ensure that p_θ defines a proper density $\int p_\theta(\mathbf{x})\,d\mathbf{x} = 1$
- Ability to evaluate density p_θ at sample points $\mathbf{x}$
  - trivial for models such as exponential families (simple formulas)
  - impractical for complex models (Markov networks, DNNs)
- What are strategies for more complex models?

Unnormalized Models

- Another direction: unnormalized models – represent improper density functions

$$\underbrace{\bar{p}_\theta(\mathbf{x})}_{\text{represented}} = \underbrace{c_\theta}_{\text{unknown}} \cdot \underbrace{p_\theta(\mathbf{x})}_{\text{normalized}}$$

- We cannot use log-liklihood, because: scaling up $\bar{p}_\theta$ $\Longrightarrow$ unbounded likelihood

- Alternative estimation method for unnormalized models?

Score Matching

- Is there an operator that we can apply to $\bar{p}_\theta$ that does not depend on normalization? Yes!
- Score Matching (Hyvarinen 2005).

$$\psi_\theta := \nabla \log \bar{p}_\theta, \quad \psi = \nabla \log p$$

  - minimize criterion

$$J(\theta) = \mathbf{E}\|\psi_\theta - \psi\|^2$$

  - equivalently (eliminate ψ by 'integration by parts')

$$J(\theta) \overset{\pm c}{=} \mathbf{E}\left[\sum_i \partial_i \psi_{\theta,i} - \tfrac{1}{2}\psi_{\theta,i}^2\right]$$

  - expectation approximated by sampling

Section 2

Autoencoders

Linear Autoencoder

Given: data points $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, k$.

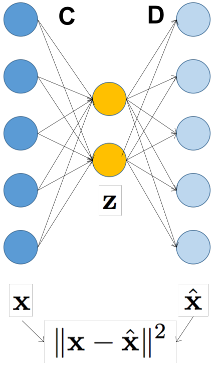
Goal: compress data into m-dimensional ($m \le n$) representation.

Linear auto-encoding
(with hidden layer $\mathbf{z} \in \mathbb{R}^m$)

$$\mathbf{x} \overset{\mathbf{C}}{\longmapsto} \mathbf{z} \overset{\mathbf{D}}{\longmapsto} \hat{\mathbf{x}} \overset{\mathcal{R}}{\longmapsto} \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Optimal choices of $\mathbf{C}$ and $\mathbf{D}$ s.t.

$$\frac{1}{2k}\sum_{i=1}^{k} \|\mathbf{x}_i - \mathbf{D}\mathbf{C}\mathbf{x}_i\|^2 \leftarrow \min$$



Singular Value Decomposition

SVD of data matrix $\mathbf{X} = \left[\mathbf{x}_1 \cdots \mathbf{x}_k\right] \in \mathbb{R}^{n \times k}$

$$\mathbf{X} = \mathbf{U} \cdot \underbrace{\mathsf{diag}^{\dagger}(\sigma_1, \ldots, \sigma_{\min(n,k)})}_{=:\boldsymbol{\Sigma} \in \mathbb{R}^{n \times k}} \cdot \mathbf{V}^{\top}, \mathbf{U} \in \mathbb{R}^{n \times n},\ \mathbf{V} \in \mathbb{R}^{k \times k}$$

Eckart-Young theorem: for $m \leq \min(n, k)$

$$\underset{\hat{\mathbf{X}}:\mathsf{rank}(\hat{\mathbf{X}})=m}{\operatorname{argmin}} \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 = \mathbf{U}_m \cdot \underbrace{\mathsf{diag}(\sigma_1, \ldots, \sigma_m)}_{=:\boldsymbol{\Sigma}_m \in \mathbb{R}^{m \times m}} \cdot \mathbf{V}_m^{\top}$$

- subscript m refers to a matrix pruned to m columns.

No linear auto-encoder with m hidden units can improve on SVD as rank$(\mathbf{CD}) \leq m$. But can it achieve it? Yes!

Optimal Linear Compression

Proposition: Given data $\mathbf{X} = \mathbf{U} \cdot \text{diag}(\sigma_1, \ldots, \sigma_n) \cdot \mathbf{V}^\top$. The choice $\mathbf{C} = \mathbf{U}_m^\top$ and $\mathbf{D} = \mathbf{U}_m$ minimizes the squared reconstruction error of a two layer linear auto-encoder with m hidden units.

Proof:

$$\begin{aligned} \mathbf{DCX} &= \underbrace{\mathbf{U}_m}_{\text{1st}} \underbrace{\mathbf{U}_m^\top}_{\text{2nd}} \underbrace{\mathbf{U\Sigma V}^\top}_{\text{data}} \\ &= \mathbf{U}_m \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{\Sigma} \, \mathbf{V}^\top = \mathbf{U}_m \begin{bmatrix} \mathbf{\Sigma}_m \, \mathbf{0} \end{bmatrix} \mathbf{V}^\top = \mathbf{U}_m \, \mathbf{\Sigma}_m \, \mathbf{V}_m^\top \end{aligned}$$

Comment: Note that we can do weight sharing between decoder and encoder network: $\mathbf{D} = \mathbf{C}^\top$.

Optimal Linear Compression (cont'd)

Is this choice unique? No!

For any invertible matrix $\mathbf{A} \in \mathsf{GL}(m)$:

$$\underbrace{(\mathbf{U}_m \mathbf{A}^{-1})}_{=:\tilde{\mathbf{D}}} \cdot \underbrace{(\mathbf{A}\mathbf{U}_m^\top)}_{=:\tilde{\mathbf{C}}} = \mathbf{U}_m \mathbf{U}_m^\top$$

Solutions restricted to $\mathbf{D} = \mathbf{C}^\top$ (weight sharing)

$\Longrightarrow \mathbf{A}^{-1} = \mathbf{A}^\top$, i.e. $\mathbf{A} \in \mathsf{O}(m)$ (orthogonal group)

$\Longrightarrow$ mapping $\mathbf{x} \mapsto \mathbf{z}$ only determined up to rotations

Principal Component Analysis

Assume that we have centered the data (pre-processing)

$$\mathbf{x}_i \mapsto \mathbf{x}_i - \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i$$

$\implies \mathbf{S} := \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{n \times n}$ is the sample covariance matrix

$\implies \mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$

$\implies$ columns of $\mathbf{U}$ are eigenvectors of covariance matrix

$\implies \mathbf{U}_m\mathbf{U}_m^\top$ is projection to m principal components of $\mathbf{S}$

This is equivalent to Principal Component Analysis (PCA).

Non-linear Autoencoder

Autoencoder: any network that aims to learn the identity map.

Just a special case of a feedforward network $\Longrightarrow$ backpropagation.

Typically: break down network into two parts: G and H such that

- $F = H \circ G \approx (\mathbf{x} \mapsto \mathbf{x})$
- Encoder $G = F_l \circ \cdots \circ F_1 : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{x} \mapsto \mathbf{z} := \mathbf{x}^l$
- Decoder $H = F_L \circ \cdots \circ F_{l+1} : \mathbb{R}^m \to \mathbb{R}^n$, $\mathbf{z} \mapsto \mathbf{y} = \hat{\mathbf{x}}$
- layer l is usually a "bottleneck" layer

Can learn powerful non-linear generalization of PCA.

Representation Learning

Autoencoder provides a canonical way of representation learning.

Data compression as a "proxy", not as a goal.

Separate signal from noise.

Unsupervised learning, no labels required.

Nice `js` demo for digit images:
`http://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html`

Regularized Autoencoder

One can regularize the code $\mathbf{z}$ via regularizer $\Omega(\mathbf{z})$
$\Longrightarrow$ regularized autoencoder

Flavors of regularization:

- standard L_2 penalty: ability to learn "overcomplete" codes
- code sparseness. e.g. via $\Omega(\mathbf{z}) = \lambda \|\mathbf{z}\|_1$
- contractive autoencoders, $\Omega(\mathbf{z}) = \lambda \left\| \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right\|_F^2$
  - penalizes Jacobian, generalizes weight decay
    (cf. Rifai et al., 2011)

Denoising Autoencoder

Regular autoencoders:

$$\min \to \mathbf{E}_{\mathbf{x}}[\ell(\mathbf{x}, (H \circ G)(\mathbf{x}))], \quad \text{e.g. } \ell(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Denoising Autoencoder (Vincent et al, 2010):

Perturb inputs $\mathbf{x} \mapsto \mathbf{x}_{\boldsymbol{\eta}}$, where $\boldsymbol{\eta}$ is a random noise vector, e.g., additive (white) noise

$$\mathbf{x}_\eta = \mathbf{x} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

$$\min \to \mathbf{E}_{\mathbf{x}} \mathbf{E}_{\boldsymbol{\eta}}[\ell(\mathbf{x}, (H \circ G)(\mathbf{x}_{\boldsymbol{\eta}}))]$$

Idea: learn features that are robust under noise.

Hope: $\|\mathbf{x} - \mathbf{x}_{\boldsymbol{\eta}}\|^2 > \|\mathbf{x} - H(G(\mathbf{x}_{\boldsymbol{\eta}}))\|^2 =$ de-noising

Section 3

Factor Analysis

Latent Variable Models

Generic way of defining probabilistic, i.e. generative models: latent variable models

1. Latent variable $\mathbf{z}$, with distribution $p(\mathbf{z})$
2. Conditional models for observables $\mathbf{x}$, $p(\mathbf{x}|\mathbf{z})$
3. Observed data model: integrating/summing out latent variables:

$$p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})\,\mu(d\mathbf{z}) = \begin{cases} \mu = \text{Lebesgue} & \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})\,d\mathbf{z} \\ \mu = \text{counting} & \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) \end{cases}$$

Simple discrete model: mixture model

- $\mathbf{z} \in \{1, \ldots, K\}$, $p(\mathbf{z})$ = mixing proportions
- $p(\mathbf{x}|\mathbf{z})$: conditional densities (e.g. Gaussians)

Linear Factor Analysis

Latent variable prior $\mathbf{z} \in \mathbb{R}^m$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Linear observation model, $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x} = \boldsymbol{\mu} + \mathbf{W}\mathbf{z} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad \boldsymbol{\Sigma} := \text{diag}(\sigma_1^2, \dots \sigma_n^2)$$

- $\boldsymbol{\eta}$ and $\mathbf{z}$ are independent
- typically: $m < n$, fewer factors than features
- few factors account for dependencies b/w many observables
- MLE for $\boldsymbol{\mu}$ on training set: $\hat{\boldsymbol{\mu}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{x}_i$

  (can assume data is centered)

Linear Factor Analysis

Proposition: In the factor analysis model

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \boldsymbol{\Sigma})$$

Proof:

1. moment generating functions (MGF) and their properties
2. multivariate normal distributions and their MGFs
3. actual proof of the proposition

Moment Generating Functions

Moment generating function of random vector $\mathbf{x}$

$$M_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}, \quad M_{\mathbf{x}}(\mathbf{t}) := \mathbf{E}_{\mathbf{x}} \exp[\mathbf{t}^\top \mathbf{x}]$$

Uniqueness theorem (e.g. Feller 1968, Vol. 2): If $M_{\mathbf{x}}$, $M_{\mathbf{y}}$ exist for RVs $\mathbf{x}$, $\mathbf{y}$ and $M_{\mathbf{x}} = M_y$ then (essentially) $p(\mathbf{x}) = p(\mathbf{y})$.

$M_{\mathbf{x}}$ represents moments of $\mathbf{x}$ in the following way: $k_1, \ldots, k_n \in \mathbb{N}$,

$$\mathbf{E}\left[x_1^{k_1} \cdots x_n^{k_n}\right] = \frac{\partial^k}{\partial t_1^{k_1} \ldots \partial t_n^{k_n}} M_{\mathbf{x}}\Big|_{\mathbf{t}=\mathbf{0}}$$

Moment generating functions can be used to deal with sums of i.i.d. random variables $M_{\mathbf{x}+\mathbf{y}} = M_{\mathbf{x}} \cdot M_{\mathbf{y}}$

Multivariate Normal Distribution

$\mathbb{R}^n \ni \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

- $\boldsymbol{\mu}$: mean, $\mathbf{E}\mathbf{x} = \boldsymbol{\mu}$
- $\boldsymbol{\Sigma}$: variance-covariance matrix, $\mathbf{E}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^\top = \Sigma$

Probability density function

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]}{\sqrt{(2\pi)^n \cdot \det(\boldsymbol{\Sigma})}}$$

Moment generating function

$$M_{\mathbf{x}}(\mathbf{t}) = \exp\left[\mathbf{t}^\top \boldsymbol{\mu} + \frac{1}{2}\mathbf{t}^\top \boldsymbol{\Sigma}\mathbf{t}\right]$$

Linear Factor Analysis

Proof (continued):

- define $\tilde{\mathbf{x}} := \mathbf{W}\mathbf{z}$ such that $\mathbf{x} = \tilde{\mathbf{x}} + \boldsymbol{\mu} + \boldsymbol{\eta}$, then

$$M_{\tilde{\mathbf{x}}}(\mathbf{t}) = \mathbf{E}_{\tilde{\mathbf{x}}} \exp\left[\mathbf{t}^\top \tilde{\mathbf{x}}\right] = \mathbf{E}_{\mathbf{z}} \exp\left[\mathbf{t}^\top \mathbf{W}\mathbf{z}\right] = M_{\mathbf{z}}(\mathbf{W}^\top \mathbf{t})$$

- with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$M_{\mathbf{z}}(\mathbf{W}^\top \mathbf{t}) = \exp\left[\frac{1}{2}(\mathbf{W}^\top \mathbf{t})^\top \mathbf{I} (\mathbf{W}^\top \mathbf{t})\right] = \exp\left[\frac{1}{2}\mathbf{t}^\top (\mathbf{W}\mathbf{W}^\top)\mathbf{t}\right]$$

- $\mathbf{x} = \tilde{\mathbf{x}} + \boldsymbol{\eta} + \boldsymbol{\mu}$

$$M_{\mathbf{x}} = M_{\hat{\mathbf{x}}} \cdot M_{\boldsymbol{\eta}} \cdot \exp[\mathbf{t}^\top \boldsymbol{\mu}] = \exp\left[\mathbf{t}^\top \boldsymbol{\mu} + \frac{1}{2}\mathbf{t}^\top (\mathbf{W}\mathbf{W}^\top + \boldsymbol{\Sigma})\mathbf{t}\right]$$

$\square$

(Non-)Identifiability

Note that for $\mathbf{W}$ and any $m \times m$ orthogonal matrix $\mathbf{Q}$

$$(\mathbf{WQ})(\mathbf{WQ})^\top = \mathbf{W}\underbrace{\mathbf{QQ}^\top}_{=\mathbf{I}}\mathbf{W}^\top = \mathbf{WW}^\top$$

Consequence:
factors are only identifiable up to rotations/reflections in $\mathbb{R}^m$

$\Longrightarrow$ factor rotations for "interpretability"

Data Compression View

How is factor analysis related to data compression?

Encoder step: implicitly defined by posterior distribution:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \quad \text{(Bayes rule)}$$

posterior $=$ normal distribution (cf. `http://cs229.stanford.edu/`)

$$\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} = \mathbf{W}^\top \left(\mathbf{W}\mathbf{W}^\top + \boldsymbol{\Sigma}\right)^{-1} (\mathbf{x} - \boldsymbol{\mu})$$
$$\boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}} = \mathbf{I} - \mathbf{W}^\top \left(\mathbf{W}\mathbf{W}^\top + \boldsymbol{\Sigma}\right)^{-1} \mathbf{W}$$

Data Compression View (cont'd)

Assume $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$: encoding via pseudoinverse in the limit

$$\mathbf{W}^\top \left(\mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I} \right)^{-1} \overset{\sigma^2 \to 0}{\longrightarrow} =: \mathbf{W}^\dagger \in \mathbb{R}^{m \times n}$$

- right pseudoinverse $\mathbf{W}\mathbf{W}^\dagger = \mathbf{I} \in \mathbb{R}^{n \times n}$
- $\mathbf{W}$ orthogonal columns $\Longrightarrow \mathbf{W}^\dagger = \mathbf{W}^\top$

Consequently:

$$\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} \to \mathbf{W}^\dagger (\mathbf{x} - \boldsymbol{\mu}), \quad \mathbf{\Sigma}_{\mathbf{z}|\mathbf{x}} \to \mathbf{0}$$

- knowing $\boldsymbol{W}$ (assuming $\mathbf{\Sigma}$ is isotropic) $\Longrightarrow$ easy to compute $\mathbf{z}$

Maximum Likelihood Estimation

What can we say about $\mathbf{W}$? Analyze optimality conditions. (Assume data is centered beforehand)

Log-probability for $\mathbf{x}_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{A})$, $\mathbf{X} = \left[\mathbf{x}_1 \cdots \mathbf{x}_k\right]$ with empirical co-variance matrix $\mathbf{S} := \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \mathbf{x}_i^\top$.

$$\log p(\mathbf{X}; \mathbf{A}) = \text{const} - \frac{k}{2}\left[\text{trace}\left(\mathbf{S}\mathbf{A}^{-1}\right) - \log \det(\mathbf{A})\right]$$

Matrix gradients

$$\nabla_{\mathbf{A}} \text{trace}\left(\mathbf{S}\mathbf{A}^{-1}\right) = -\mathbf{A}^{-1}\mathbf{S}\mathbf{A}^{-1}, \quad \nabla_{\mathbf{A}} \log \det \mathbf{A} = \mathbf{A}^{-1}$$

$$\nabla_{\mathbf{A}} \log p(\mathbf{X}; \mathbf{A}) \overset{!}{=} 0 \iff \mathbf{S}\mathbf{A}^{-1} = \mathbf{I}$$

Maximum Likelihood Estimation (continued)

Constrained covariance matrix

$$\mathbf{A} = \mathbf{W}\mathbf{W}^\top + \mathbf{\Sigma}$$

Chain rule

$$\nabla_{\mathbf{W}}\mathbf{A} = 2\mathbf{W}, \quad \nabla_{\mathbf{\Sigma}}\mathbf{A} = \mathbf{I},$$

Stationary condition for $\mathbf{W}$ given $\mathbf{\Sigma}$

$$\mathbf{S}\left(\mathbf{\Sigma} + \mathbf{W}\mathbf{W}^\top\right)^{-1}\mathbf{W} = \mathbf{W}$$

Generally, no analytic solution known. Special case: $\mathbf{\Sigma} = \sigma^2\mathbf{I}$.

Maximum Likelihood Estimation (continued)

Assume $\mathbf{W}^\top \mathbf{W} = \text{diag}(\rho_i^2)$, then by Woodbury's formula:

$$\left(\sigma^2 \mathbf{I}_n + \mathbf{W}\mathbf{W}^\top\right)^{-1} \mathbf{W} = \mathbf{W}\text{diag}\left(\frac{1}{\sigma^2 + \rho_i^2}\right)$$

Thus we get for each column $\mathbf{w}_i$ of $\mathbf{W}$ an eigenvector equation

$$\mathbf{S}\mathbf{w}_i = (\sigma^2 + \rho_i^2)\mathbf{w}_i$$

If $\mathbf{u}_i$ is the i-th eigenvector of $\mathbf{S}$, then

$$\mathbf{w}_i = \rho_i \mathbf{u}_i, \quad \rho_i^2 = \max\{0, \lambda_i - \sigma^2\}$$

Probabilistic PCA = PCA as $\sigma^2 \to 0$ (Tipping & Bishop, 1999)

Section 4

Latent Variable Models

DeFinetti's Theorem

- For exchangeable data, we can decompose them by a latent variable model

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \int \prod_{i=1}^{N} p_\theta(\mathbf{x}_i|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

- We expect that those hidden variables are:
  Interpretable and actionable and even show causal relations.

Interpretability

F. Dohsi-Velez et al. (NIPS 2015)

Objectives such as data exploration present unique challenges and opportunities for problems in unsupervised learning. While in more typical scenarios, the discovered latent structures are simply required for some downstream task – such as features for a supervised prediction problem – in data exploration, the model must provide information to a domain expert in a form that they can readily interpret. It is not sufficient to simply list what observations are part of which cluster; one must also be able to explain why the data partition in that particular way. These explanations must necessarily be succinct, as people are limited in the number of cognitive entities that they can process at one time.

Latent Variable Models

- Classically: Define complex models via marginalization of a latent variable model

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \quad \text{or} \quad p_\theta(\mathbf{x}) = \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z})$$

- EM algorithm (ELBO: evidence lower bound)

$$\begin{aligned} \log p_\theta(\mathbf{x}) &\geq \mathbf{E}_q \left[\log p_\theta(\mathbf{x}, \mathbf{z})\right] + \mathsf{H}(q(\mathbf{z})) \qquad \xleftarrow{\max \mathbf{E}} \theta, q \\ &= \mathbf{E}_q \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathsf{KL}(q(\mathbf{z})||p_\theta(\mathbf{z})) \end{aligned}$$

  - optimal $q(\mathbf{z}; \mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$ (posterior), not always tractable

- Variational inference: further approximation
  - restrict to simpler families of distribution (weakening of ELBO)
  - amortized inference $\Longrightarrow$ variational auto-encoders

Dimensionality Reduction

General model:

$$\mathbf{X} = f(\mathbf{Z}\mathbf{B})$$

where $\mathbf{X}$ is $N \times D$, $\mathbf{Z}$ is $N \times K$ and $\mathbf{B}$ is $K \times D$ (and $K \ll D$).

Depending on $f(\cdot)$, $\mathbf{Z}$ and $\mathbf{B}$, we arrive at different models:

- Principal Component Analysis/Factor Analysis.
- Nonnegative Matrix Factorization.
- LLE/Isomap/GPLVM.
- Restricted Boltzmann Machine.
- Dirichlet Processes (aka Chinese Restaurant Process).
- Beta Processes (aka Indian Buffet Process).
- Implicit Models (e.g. Generative Adversarial Networks).

Implicit Models

- Statistical models via: generating stochastic mechanism or simulation process
  - aka: ‘implicit’ models
- Deep implicit models
  - latent code $\mathbb{R}^d \ni \mathbf{z} \sim \pi(\mathbf{z})$, e.g. $\pi(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - parameterized mechanism $F_\theta : \mathbb{R}^d \to \mathbb{R}^m$
  - induced distribution $\mathbb{R}^m \ni \mathbf{x} \sim p_\theta(\mathbf{x})$
  - sampling is easy: random vector + forward propagation