Linear Factor Models

Deep Learning Decal

Hosted by Machine Learning at Berkeley

ML@B

Agenda

Background

Linear Factor Models

Probabilistic PCA

Independent Component Analysis

Slow Feature Analysis

Sparse Coding

Manifold Learning

Background

Where are we...?

ML@B

- We are now all proficient in understanding deep neural networks and how to optimize them
- But... many research frontiers in deep learning involve **probabilistic** models of the input $p_{model}(\mathbf{x})$
- We are often interested in using probabilistic inference to predict any of the variables in its environment, given any of the other variables

*** 99% of the material today is heavily borrowed from the Deep Learning textbook

Latent variables

ML@B

- Many probabilistic models have latent variables, $\mathbf{h}$, with

$$p_{model}(\mathbf{x}) = E_h p_{model}(\mathbf{x}|\mathbf{h})$$

- Latent variables are another way to represent the data
- Idea: distributed representations based on latent variables can obtain all of the advantages of learning which we have seen with deep networks

Latent variables: a review

ML@B

- **Latent variables**, as opposed to observable variables, are variables that are not observed but instead inferred from observed variables
- Latent variable models are used in: psychology, economics, engineering, medicine, physics, ML/AI, bioinformatics, NLP, management, and pretty much everywhere else

Example of a latent variable

ML@B

- In economics, we are often interested in measuring things such as quality of life, morale, happiness, and other things
- These things cannot be directly measured!
- The idea is to link these latent variables to observable variables
- For example, perhaps quality of life can be inferred from some linear combination of wealth, employment, environment, physical health, education, leisure time, etc...

Linear Factor Models

Back to deep learning...

ML@B

- As an introduction to probabilistic models with latent variables, we start with one of the simplest classes: **linear factor models**
- Warning: you may not be implementing any linear factor models to solve state-of-the-art problems, **but** they provide a nice building block for mixture models or deeper probabilistic models
- Many of the approaches we discuss today are necessary to build generative models that more advanced deep models (keep coming to class!) models will expand upon

Linear factor models

ML@B

- Defined by the use of a stochastic, linear decoder that generates $\mathbf{x}$ by adding noise to a linear transformation of $\mathbf{h}$
- Allow us to discover explanatory factors that have a simple joint distribution
- Simplicity of the linear decoder motivated these as some of the first latent variable models

Linear factor models (LFMs)

LFMs describe the data generation process as follows:

1. Sample the explanatory factors $\mathbf{h}$ from a distribution

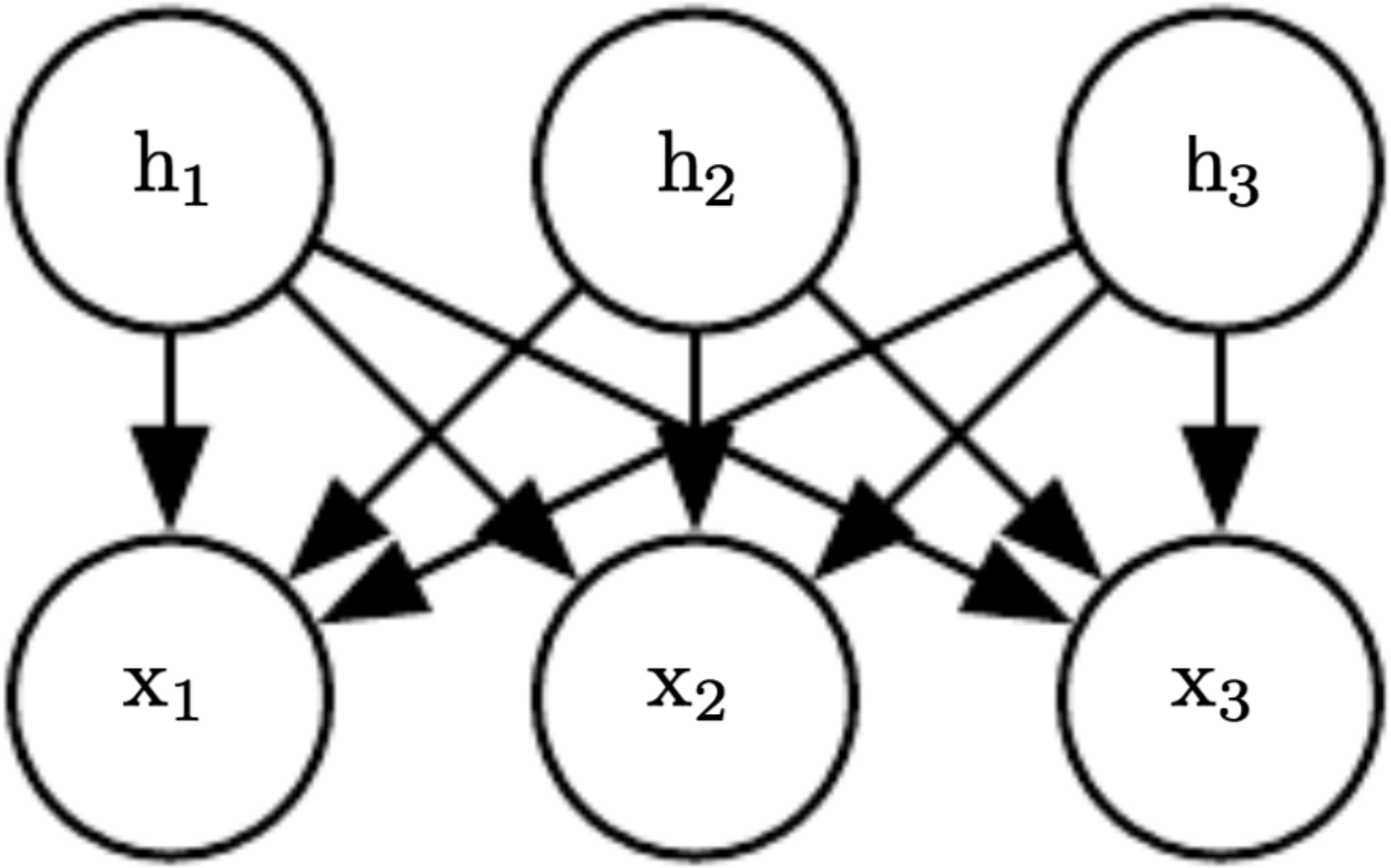
$$\mathbf{h} \sim p(\mathbf{h})$$

   where $p(\mathbf{h})$ is a factorial distribution (i.e. $p(\mathbf{h}) = \prod_i p(h_i)$
2. Sample the real-valued observable variables given the factors:

$$\mathbf{x} = \mathbf{W}\mathbf{h} + \mathbf{b} + \text{noise}$$

   where the noise is typically Gaussian and diagonal

LFMs illustrated



$$\mathbf{x} = \boldsymbol{W}\,\mathbf{h} + \boldsymbol{b} + \mathbf{noise}$$

Types of LFMs

ML@B

- The directed graphical model on the previous slide describes the LFM family, where we assume that observed $\mathbf{x}$ is obtained by a linear combination of independent latent factors $\mathbf{h}$, plus some noise
- Different types of LFMs make different choices about the form of the noise and of the prior $p(\mathbf{h})$
- We will touch upon:
  - Probabilistic PCA and factor analysis
  - Independent component analysis (ICA)
  - Slow feature analysis
  - Sparse coding

Factor analysis

- (Batholomew, 1987; Basilevsky, 1994)
- Here, the latent variable prior is just the unit variance Gaussian:

$$\mathbf{h} \sim N(\mathbf{h}; \mathbf{0}, \mathbf{I})$$

- Observed values x_i are assumed to be **conditionally independent** given $\mathbf{h}$
- That is, the noise is assumed to be drawn from a diagonal covariance Gaussian distribution, with covariance matrix $\psi = \text{diag}(\sigma_1^2, ..., \sigma_n^2)$
- The latent variables should **capture the dependencies** between the observed variables x_i
- Can show that $\mathbf{x}$ is a multivariate normal:

$$\mathbf{x} \sim N(\mathbf{x}; \mathbf{b}, \mathbf{W}\mathbf{W}^\mathbf{T} + \psi)$$

Probabilistic PCA

From factor analysis to probabilistic PCA

- A slight modification to the factor analysis model allows us to cast PCA in a probabilistic framework: make the conditional variances σ_i^2 equal to each other
- Now we have:

$$\mathbf{x} \sim N(\mathbf{x}; \mathbf{b}, \mathbf{W}\mathbf{W}^\mathbf{T} + \sigma^2\mathbf{I})$$

- Equivalently:

$$\mathbf{x} = \mathbf{W}\mathbf{h} + \mathbf{b} + \sigma\mathbf{z}$$

  where $\mathbf{z} \sim N(\mathbf{z}; \mathbf{0}, \mathbf{I})$ is Gaussian noise
- Can use an iterative EM algorithm to estimate $\mathbf{W}$ and σ^2 (Tipping and Bishop (1999))

Probabilistic PCA

ML@B

- Probabilistic PCA takes advantage of the observation that most variations in the data can be captured by the latent variables, **h**, up to some small residual **reconstruction error** σ^2
- Tipping and Bishop (1999) showed that probabilistic PCA becomes PCA as $\sigma \rightarrow 0$

Relationship between PCA and probabilistic PCA

- As $\sigma \to 0$, the conditional expectation of $\mathbf{h}$ given $\mathbf{x}$ becomes an orthogonal projection of $\mathbf{x} - \mathbf{b}$ onto the space spanned by the d columns of $\mathbf{W}$
- As $\sigma \to 0$, the density model defined by probabilistic PCA becomes very sharp around the d dimensions spanned by the columns of $\mathbf{W}$
- This may cause the model to assign low likelihood to data if the data does not actually cluster near a hyperplane

Motivation behind probabilistic PCA

ML@B

- In standard PCA, we assume linearity (bases of linear combinations of the measurement-basis), that large variances = import structure, and that principal components are orthogonal
- Linearity is not always justifiable!
- Calculating the covariance matrix can be very expensive in high-dimensional or big data settings
- De-correlation is not always the best approach (first and second order statistics are not always sufficient for revealing all dependencies in data, i.e. Gaussian data)

More motivation behind probabilistic PCA

ML@B

- May not have enough data for full-rank sample covariance
- Even if we do, computing it is $O(Np^2)$
  - EM-based PCA has $O(kNP)$ complexity
- Can start to apply Bayesian inference
- pPCA defines a proper covariance structure, with parameters estimable via the EM algorithm
- pPCA can be used as a constrained Gaussian density model (classification, novelty direction, ...)

Independent Component Analysis

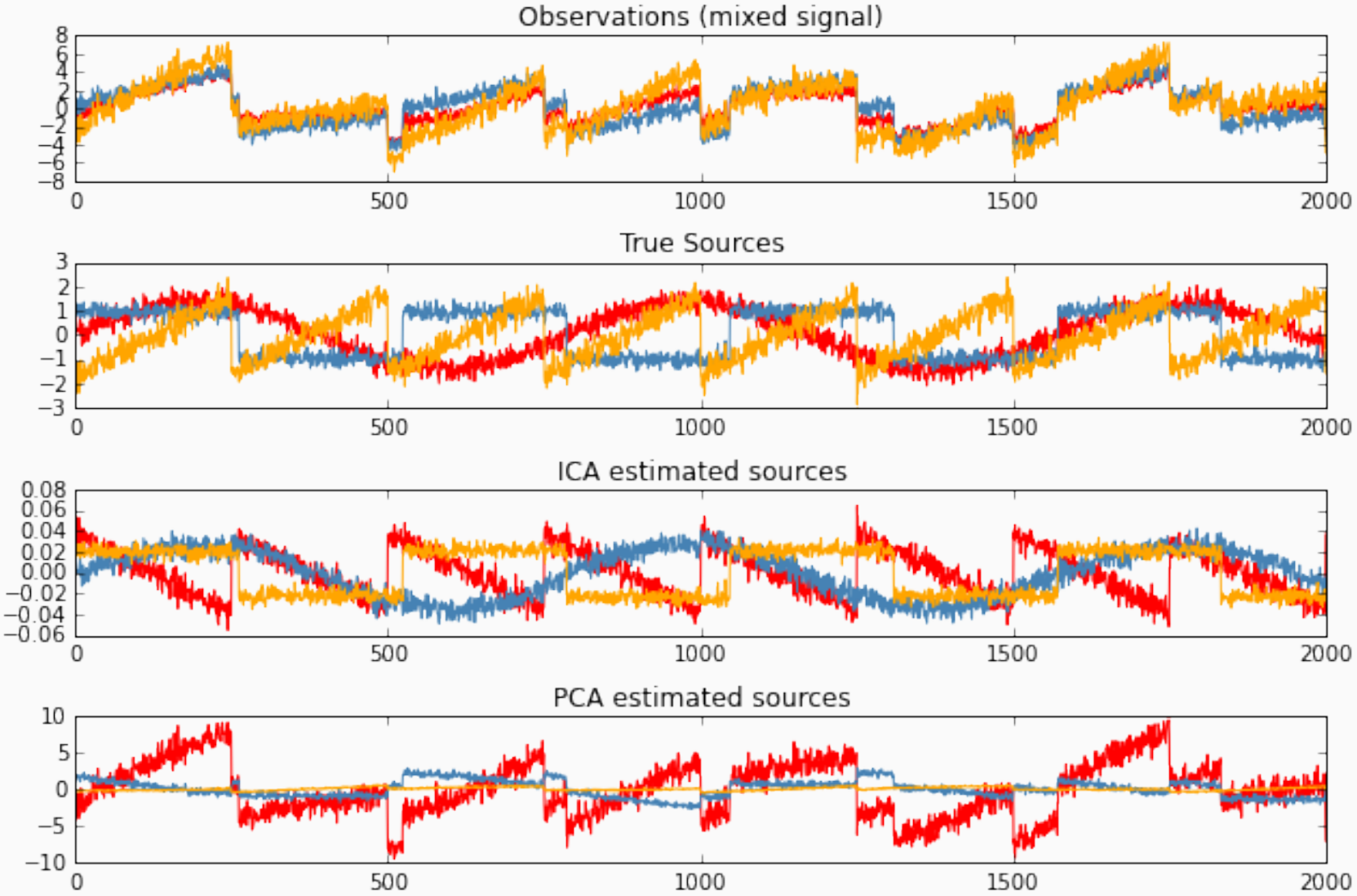
Independent component analysis (ICA)

ML@B

- One of the oldest representation learning algorithms
- Models linear factors by seeking to separate an observed signal into underlying signals that are scaled and added together
- The underlying signals are intended to be fully independent
- $\exists$ many variants

ICA

- A variant from Pham et al trains parametric generative model
- The prior $p(\mathbf{h})$ is fixed
- The model deterministically generates $\mathbf{x} = \mathbf{W}\mathbf{h}$
- A nonlinear change of variables allows us to determine $p(\mathbf{x})$
- Learning the model proceeds by using maximum likelihood


Observations (mixed signal)
True Sources
ICA estimated sources
PCA estimated sources

Motivation behind ICA

ML@B

- By choosing $p(\mathbf{h})$ to be independent, can recover factors that are as close as possible to independent
- Used to recover low-level signals that have been mixed
- Here, each data point x_i is one sensor's observation of the mixed signals, and each h_i is one estimate of the original signals
- Example: we have n people speaking simultaneously in n different microphones in different locations, ICA can detect changes in the volume between each speaker as heard by each microphone and separate the signals so that each h_i contains only one person speaking clearly

Applications of ICA

ML@B

- Optical imaging of neurons
- Neuronal spike sorting
- Facial recognition
- Removing artifacts (i.e. eye blinks) from EEG (electroencephalography) data
- Predicting stock market prices
- Mobile phone communications

More on ICA...

- Some variants add noise in the generation of $\mathbf{x}$ instead of using a deterministic decoder
- Most aim to make elements of $\mathbf{h} = \mathbf{W}^{-1}\mathbf{x}$ independent from each other
- Restricting $\mathbf{W}$ to be orthogonal may reduce computational cost
- ALL variants of ICA require $p(\mathbf{h})$ to be non-Gaussian
- Typical choice: $p(h_i) = \frac{d}{dh_i}\sigma(h_i)$
  - Larger peaks near 0 may cause ICA to learn sparse features
- Not all variants are generative models (ICA often used as an analysis tool for separating signals rather then generating data or estimating its density)

Generalizations of ICA: nonlinearity

ML@B

- ICA can be generalized to a nonlinear generative model (Hyvarinen and Pajunen (1999))
- **Nonlinear independent components estimation (NICE) (Dinh et al, 2014)**
  - Stacks a series of invertible transformations (encoder stages) with the property of having the determinant of the Jacobian of each transformation being easily computable
  - Can compute the likelihood exactly
  - Attempts to transform the data into a space where it has a factorized marginal distribution, BUT is more likely to succeed because of the nonlinear encoder
  - Each encoder has a decoder that is its perfect inverse!

Generalizations of ICA: relaxation of independence within sub-groups

ML@B

- Suppose we want to learn groups of features instead of individuals features
- We may allow statistical dependence within groups but have it discouraged between groups
- When the groups of related units are non-overlapping, this is called **independent subspace analysis**
- Can assign spatial coordinates to hidden unit and form overlapping groups of spatially neighboring units, encouraging nearby units to learn similar features
- Applied to natural images, **topographic ICA** learns Gabor filters (neighboring filters have similar orientation, location, or frequency)

Slow Feature Analysis

Slow feature analysis (SFA)

ML@B

- **SFA** uses information from time signals to learn invariant features (Wiskott and Sejnowski 2002)
- Motivated by the **slowness principle**
  - Important characteristics of scenes change slowly compared to individual measurements that make up the description of a scene
  - Example: a zebra moves from left to right across an image; individual pixels change rapidly as the zebra's stripes pass over, but the feature that indicates whether a zebra is in the image doesn't change, and the image indicating the zebra's position changes very slowly
- Want to regularize our model to learn features that change slowly over time

The slowness principle

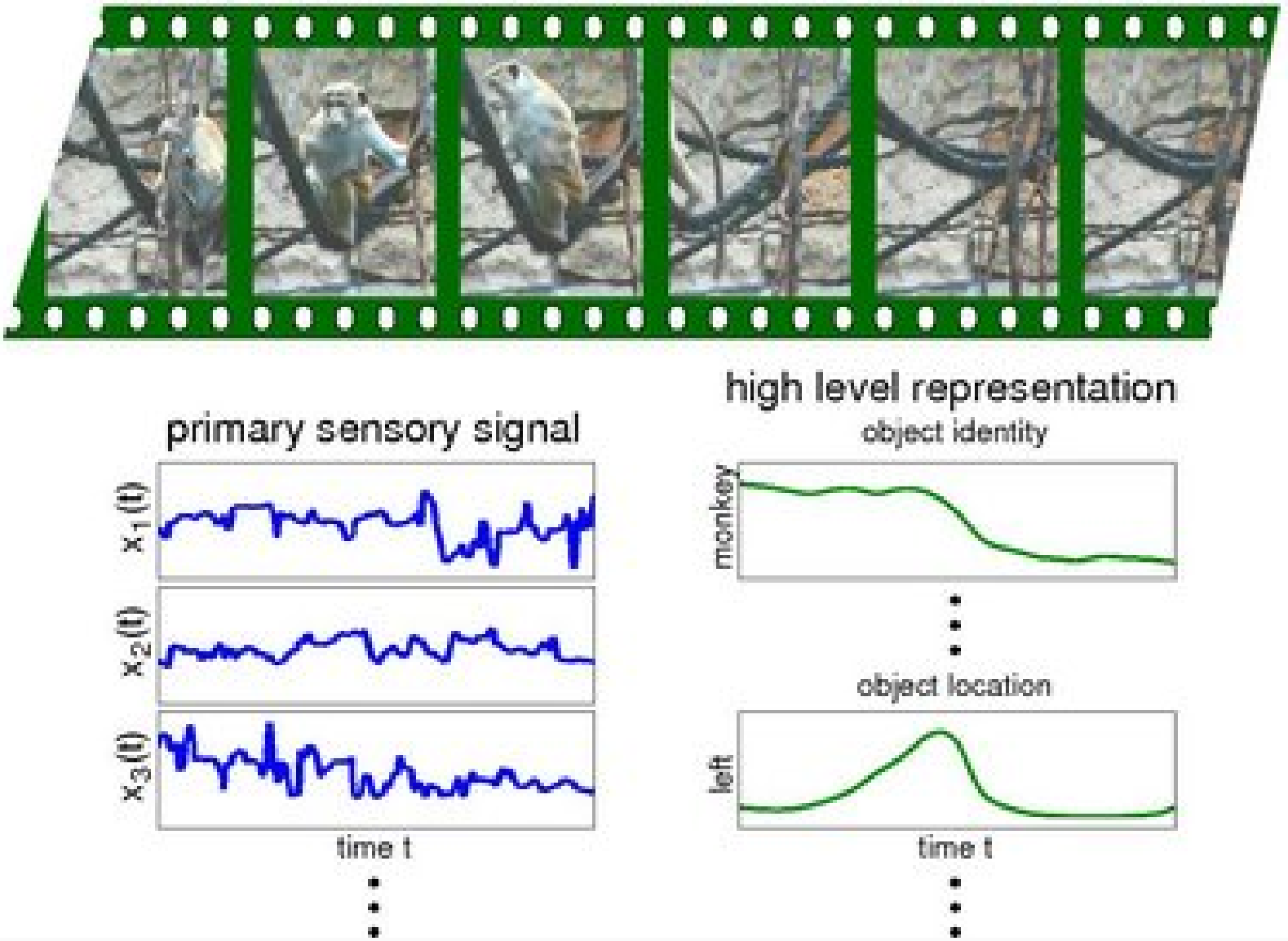
ML@B

- Can apply the slowness principle to any differentiable model trained with gradient descent
- To do so, add a cost function of the form

$$\lambda \sum_t L(f(x^{t+1}), f(x^t))$$

where λ is a hyperparameter that determines the strength of the slowness regularization, t is the time index, f is the feature extractor, and L is a loss function

The slowness principle visualized



Slow feature analysis

ML@B

- SFA applies a linear feature extractor and can be trained in closed form
- SFA is not quite a generative model - it defines a linear map between input space and feature space
    - Does not impose a prior over the feature space and thus doesn't impose a distribution $p(\mathbf{x})$ on the input space

The SFA algorithm

- Define $f(\mathbf{x};\theta)$ to be a linear transformation and solve the optimization problem:

$$min_\theta E_t(f(\mathbf{x}^{(t+1)})_i - f(\mathbf{x}^{(t)})_i)^2$$

subject to the constraints

$$E_t f(\mathbf{x}^{(t)})_i = 0$$

and

$$E_t[f(\mathbf{x}^{(t)})_i^2] = 1$$

- The mean 0 constraint of the learned feature is necessary for the solution to be unique
- The unit variance constraint is necessary to prevent the solution where all features collapse to 0

More SFA

- Learned features are ordered, with the first being the slowest (similar to PCA)
- To learn multiple features, add the third constraint:

$$\forall i < j, E_t[f(\mathbf{x}^{(t)})_i f((\mathbf{t}))_j] = 0$$

- The above constraint specifies that the features must be linearly decorrelated from each other
- Without it, all of the learned features would capture the slowest feature
- The SFA problem can be solved in closed form by a linear algebra package

Even more SFA...

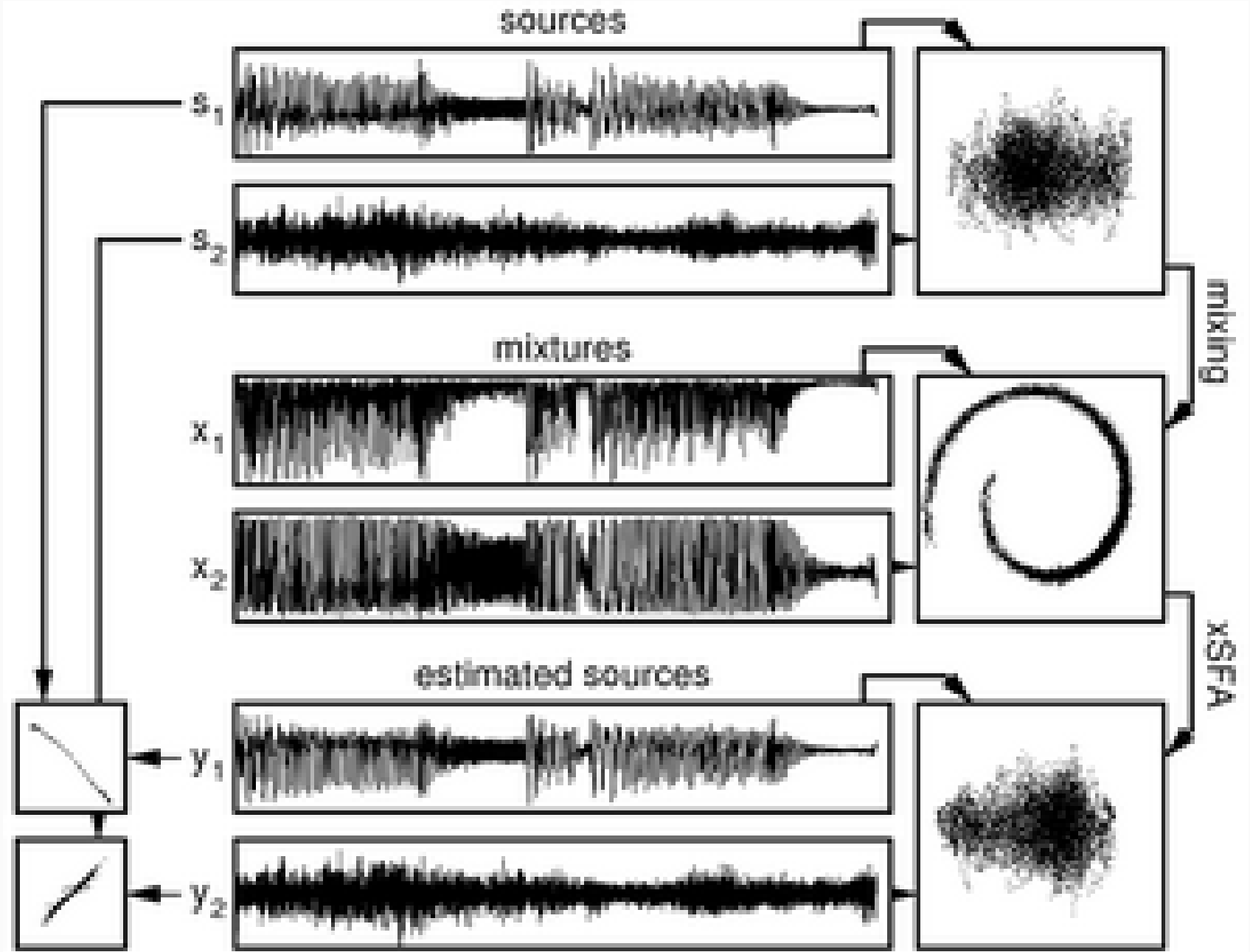
- SFA is typically used to learn nonlinear features by applying a nonlinear basis expansion to $\mathbf{x}$ before running SFA
- Linear SFA models can be composed to learn deep nonlinear slow feature extractors
- For example, first learn a linear SFA extractor, apply a nonlinear basis expansion to the output, and then learn another linear SFA feature extractor on that expansion

SFA is biologically plausible

- When trained on small patches of videos of natural scenes, SFA with quadratic basis expansions learns features that are similar to those of complex cells in the V1 cortex! (Berkes and Wiskott, 2005)
- When trained on videos of random motion in 3-D computer rendered environments, deep SFA learns features represented by neurons in rat brains that are used for navigation (Franzius et al, 2007)

s

SFA Audio example



SFA: closing thoughts

ML@B

- So far, the slowness principle has not become the basis for any state of the art applications
- Unclear what causes the limited performance
- The slowness prior might be too strong?
- Perhaps it is better to impose a prior that features should be easy to predict from one time step to the next
- Object position is a useful feature regardless of its speed, but the slowness principle encourages the model to ignore positions of objects with high speeds

Sparse Coding

Sparse coding (Olshausen and Field, 1996)

ML@B

- A linear factor model that is used for unsupervised feature learning and feature extraction
- "Sparse coding" refers to the process of inferring the value of $\mathbf{h}$ in this model
- Like most linear factor models, it uses a linear decoder plus noise to obtain a reconstruction of $\mathbf{x}$

Sparse coding: model

- Sparse coding models assume that the linear factors have Gaussian noise with isotropic precision β:

$$p(\mathbf{x}|\mathbf{h}) = N(\mathbf{x}; \mathbf{W}\mathbf{h} + \mathbf{b}, \frac{1}{\beta}\mathbf{I})$$

- $p(\mathbf{h})$ is chosen to have sharp peaks near 0
- For example, the Laplace distribution (with sparsity coefficient λ:

$$p(h_i) = Laplace(h_i; 0, \frac{2}{\lambda}) = \frac{\lambda}{4} e^{-\frac{1}{2}\lambda|h_i|}$$

and the Student-t prior:

$$p(h_i) \propto \frac{1}{(1 + \frac{h_i^2}{\nu})^{\frac{\nu+1}{2}}}$$

Sparse coding: training

- Cannot train sparse coding with maximum likelihood
- Training alternates between encoding the data and training the decoder to better reconstruct the data given the encoder
- Unlike PCA, the encoder with sparse coding is non-parametric
- The encoder solves the optimization problem in which we seek the single most likely code value:

$$\mathbf{h}^* = f(\mathbf{x}) = argmax_h p(\mathbf{h}|x)$$

- Adding the assumption of Gaussian noise and a Laplacian prior, this becomes:

$$argmax_h p(\mathbf{h}|\mathbf{x})$$

$$= argmax_h log(p(\mathbf{h}|\mathbf{x}))$$

$$= argmin_h \lambda||\mathbf{h}||_1 + \beta||\mathbf{x} - \mathbf{W}\mathbf{h}||_2^2$$

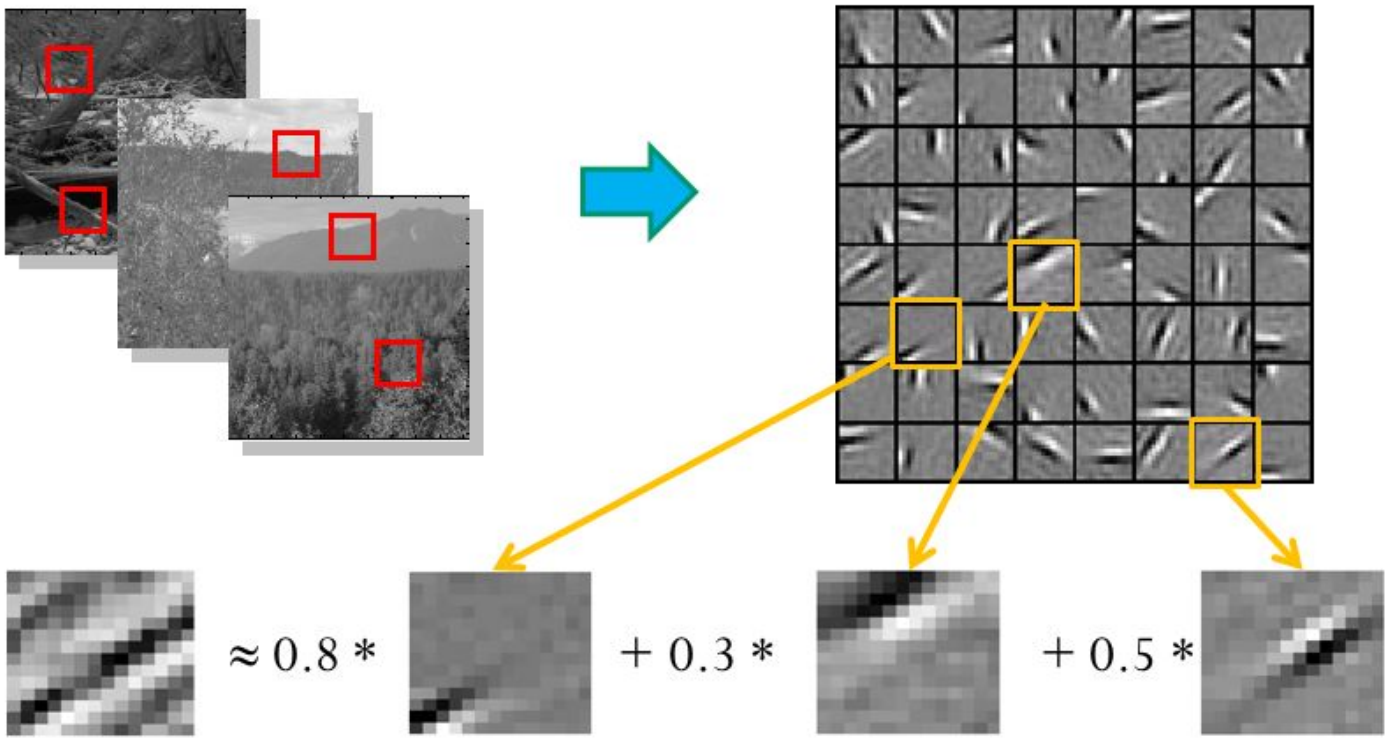
Sparse coding

ML@B

- In principle, the non-parametric encoder in sparse coding can minimize the combination of reconstruction error and log-prior better than any parametric encoder
- There is no generalization error to the encoder
- When the inference problem is convex, the optimization procedure will always find the optimal code
- Coates and Ng (2011) found that sparse coding features generalize better for object recognition tasks than features of a model based on a parametric encoder, such as a linear-sigmoid autoencoder
- Goodfellow et al (2013d) showed that a variant of sparse coding generalizes better than other feature extractors when very few labels are available (20 or fewer per class)

Sparse coding visualization

Sparse Coding Example

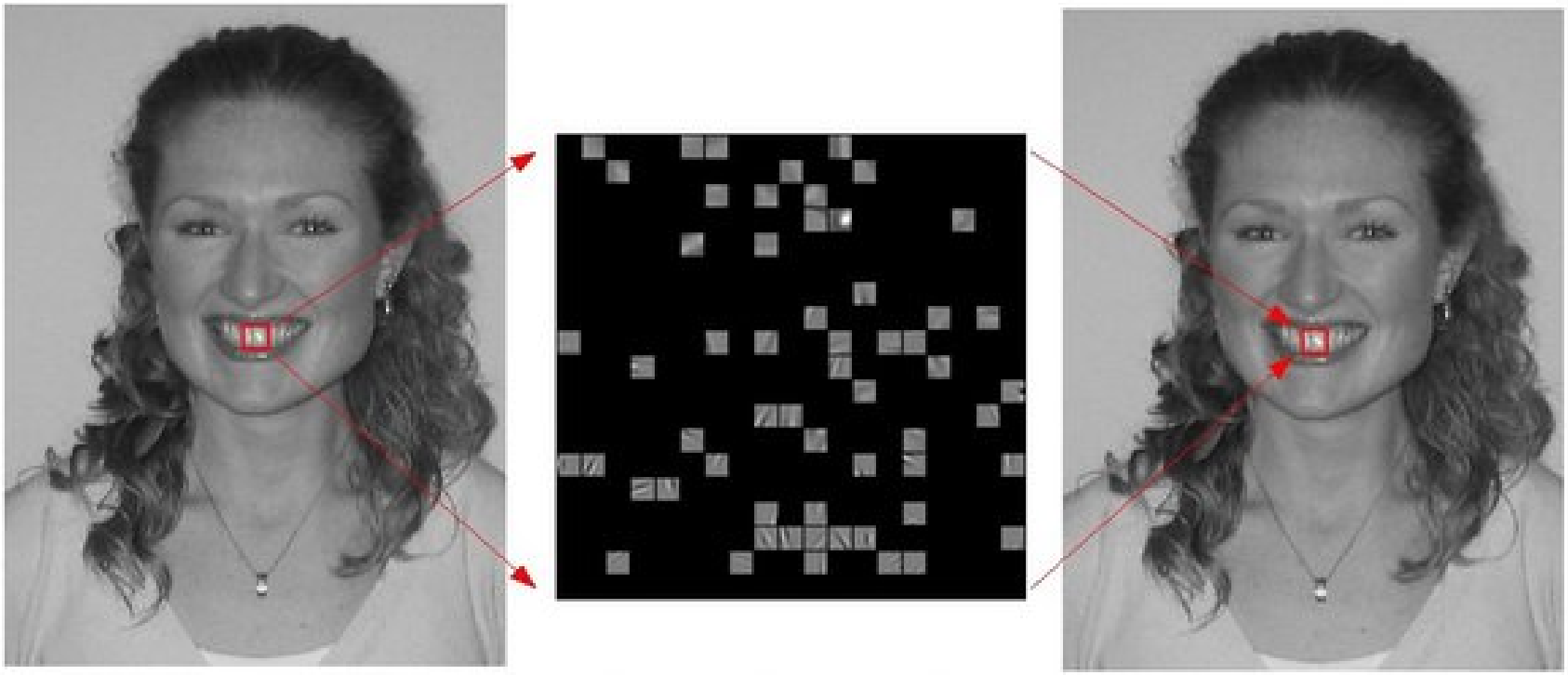


$[a_1, \ldots, a_{64}] = [0, 0, \ldots, 0, \mathbf{0.8}, 0, \ldots, 0, \mathbf{0.3}, 0, \ldots, 0, \mathbf{0.5}, 0]$
(feature representation)

Slide credit: Andrew Ng

Compact & easily interpretable

Sparse coding reconstruction



Disadvantages of sparse coding

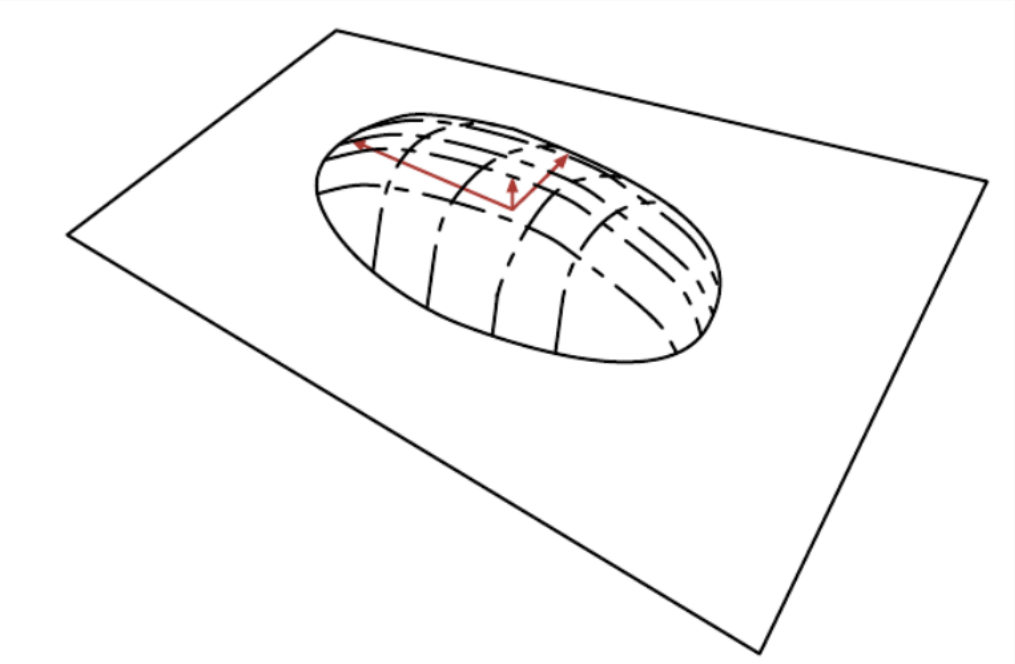
ML@B

- Non-parametric encoder requires greater time to compute $\mathbf{h}$ given $\mathbf{x}$ because it requires running an iterative algorithm
- It is not straight-forward to back-propogate through the non-parametric encoder
    - Makes it difficult to pre-train a sparse coding model with an unsupervised criterion and then fine-tune it using a supervised criterion
- Often produce poor samples, even when the model is able to reconstruct the data well and provide useful features for a classifier
    - Each feature may be learned well, but the factorial prior on the hidden code causes the model to include random subsets of all features
- Motivates the development of deeper model that impose a non-factorial distribution on the deepest layer code, or more sophisticated shallow models

Manifold Learning

Manifold interpretation of PCA

- Linear factor models, including PCA and factor analysis, can be interpreted as learning a manifold
- Probabilistic PCA can be viewed as defining a thin pancake-shaped region of high probability
    - A Gaussian distribution that is very narrow along some axes, and wide along others



Manifold interpretation of PCA

- PCA can be interpreted as aligning the pancake with a linear manifold in higher-dimensional space
- This interpretation applies to any linear autoencoder that learns matrices $\mathbf{W}$ and $\mathbf{V}$ with the goal of making the reconstruction of $\mathbf{x}$ lie as close to $\mathbf{x}$ as possible
- Let the encoder be:

$$\mathbf{h} = f(\mathbf{x}) = \mathbf{W}^T(x - \mu)$$

- The encoder computes a low-dimensional representation of h
- The decoder computes the reconstruction as:

$$\hat{\mathbf{x}} = g(\mathbf{h}) = \mathbf{b} + \mathbf{V}\mathbf{h}$$

Manifold interpretation of PCA

- The choices of linear encoder and decoder that minimize the reconstruction error:

$$E[||\mathbf{x} - \hat{\mathbf{x}}||^2]$$

  correspond to $\mathbf{V} = \mathbf{W}$ and $\mu = \mathbf{b} = E[\mathbf{x}]$
- The columns of $\mathbf{W}$ form an orthonormal basis that spans the same subspace as the principal eigenvectors of the covariance matrix, $\mathbf{C}$
- In PCA, the columns of $\mathbf{W}$ are the eigenvectors, ordered by the magnitude of their eigenvalues
- The eigenvalue λ_i of $\mathbf{C}$ corresponds to the variance of $\mathbf{x}$ in the direction of eigenvector $\mathbf{v}^{(i)}$

Other manifold learning methods

- Isomap
- Locally-linear embedding
- Spectral clustering
- t-SNE (t-distributed stochastic neighbor embedding)
  - https://artsexperiments.withgoogle.com/tsnemap/
  - https://distill.pub/2016/misread-tsne/
- Nonlinear PCA
- Diffusion maps
- Multidimensional scaling (MDS)
- Many more...
- http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Other factor analysis techniques

- Isomaps
- (Deep) canonical factor analysis
- Multiple factor analysis
- Common factor analysis
- Image factoring
- Alpha factoring
- Factor regression model

Next steps...

- LFMs are some of the simplest generative models and some of the simplest models that learn to represent data
- Akin to how linear regression models extend to deep feed-forward networks, LFMs may be extended to autoencoders and deep probabilistic models that perform the same task but are much more powerful and flexible

Questions?