Project discussion, 22 May: Mandatory but ungraded. We split into 6 sub-classes. The purpose is to make sure your project is on track, good progress and good goals. **The discussion following your presentation is the most important.**

Each group gives a ~10 min presentation by all members (each person talks for ~2 min, ~1 slide)

1) Motivation & background, which data?
2) small Example,
3) final outcome, (focused on method and data)
4) difficulties,

Timing: There are upto 8 Groups in each sub-class, thus we have **15 min in total/group, with 2 min/person 10min presentation time/group**. The discussion following a presentation might be the most important.

June 5, 5-8pm: Poster and Pizza

Generative Models

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Taxonomy of Generative Models

Today: discuss 3 most popular types of generative models today

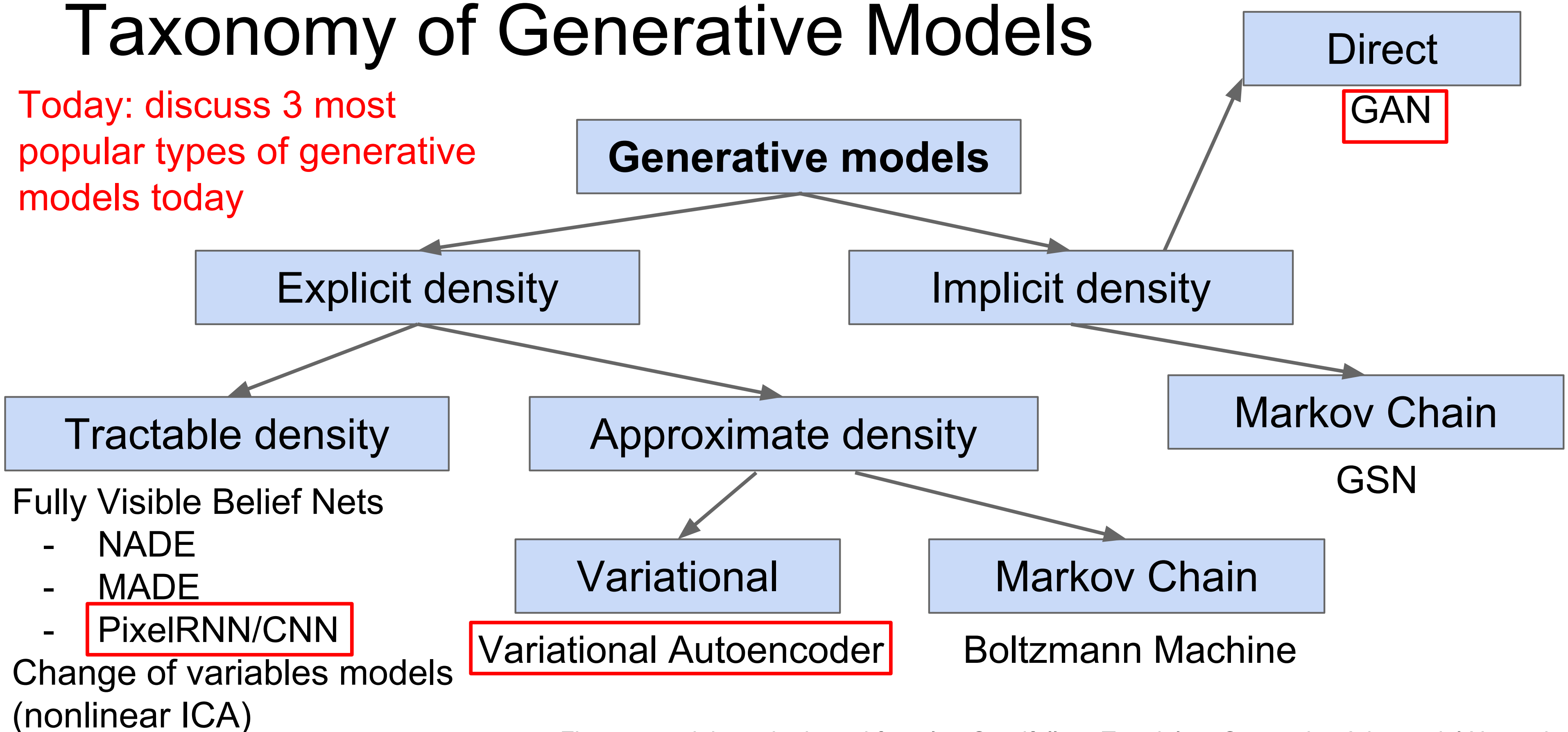


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Bayes summary

Bayes $\mathrm{p}(x|y) = \frac{\mathrm{p}(y|x)\mathrm{p}(y)}{\mathrm{p}(x)}$

Optimizing posterior $\mathrm{p}(x|y)$

You can also optimize the evidence (type II likelihood) $\mathrm{p}(\mathrm{x})$

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Then maximize likelihood of training data

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

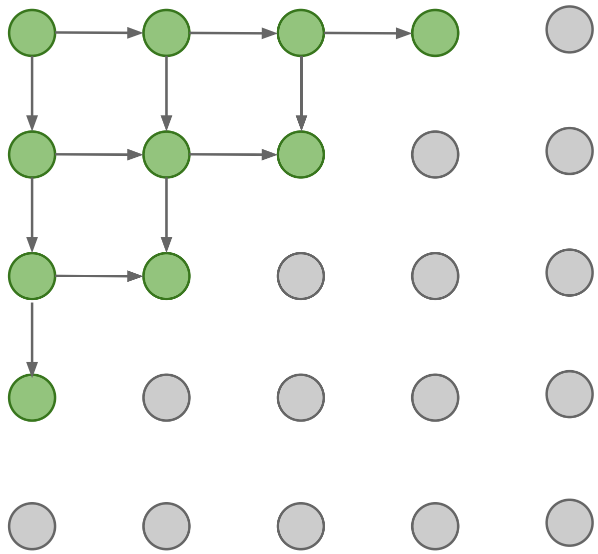
Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)



Drawback: sequential generation is slow!

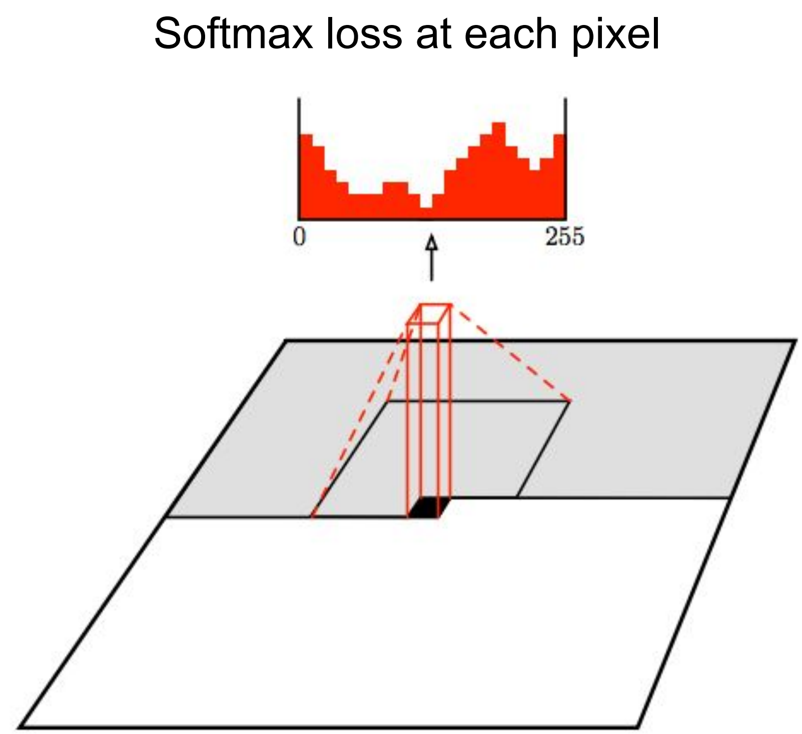
PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$



PixelRNN and PixelCNN

Pros:

- Can explicitly compute likelihood p(x)
- Explicit likelihood of training data gives good evaluation metric
- Good samples

Con:

- Sequential generation => slow

Improving PixelCNN performance

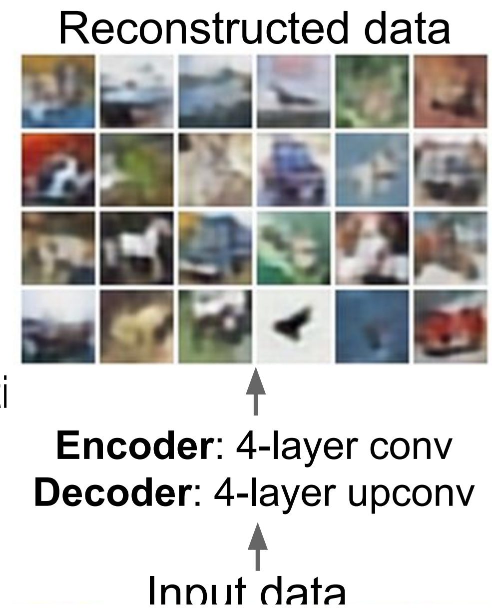
- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc…

See

- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)

Bayes rule

Some background first: Autoencoders



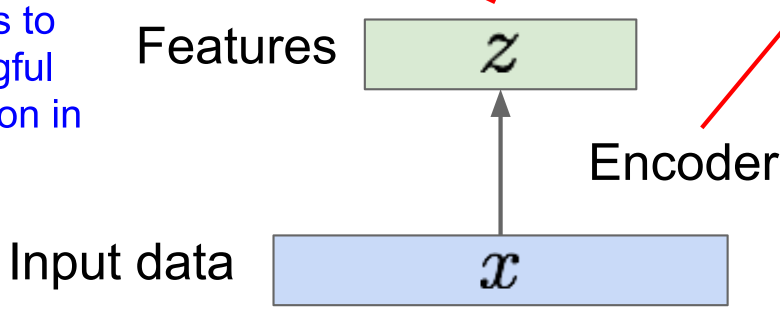
Unsupervised approach for learning a lower-dimensional feature representati
from unlabeled training data

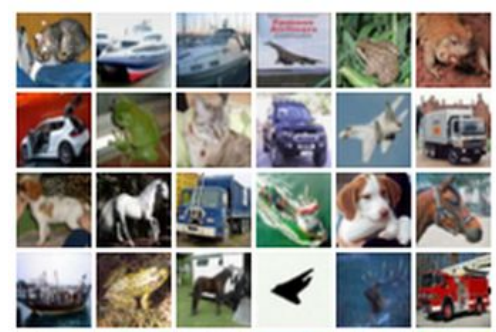
z usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

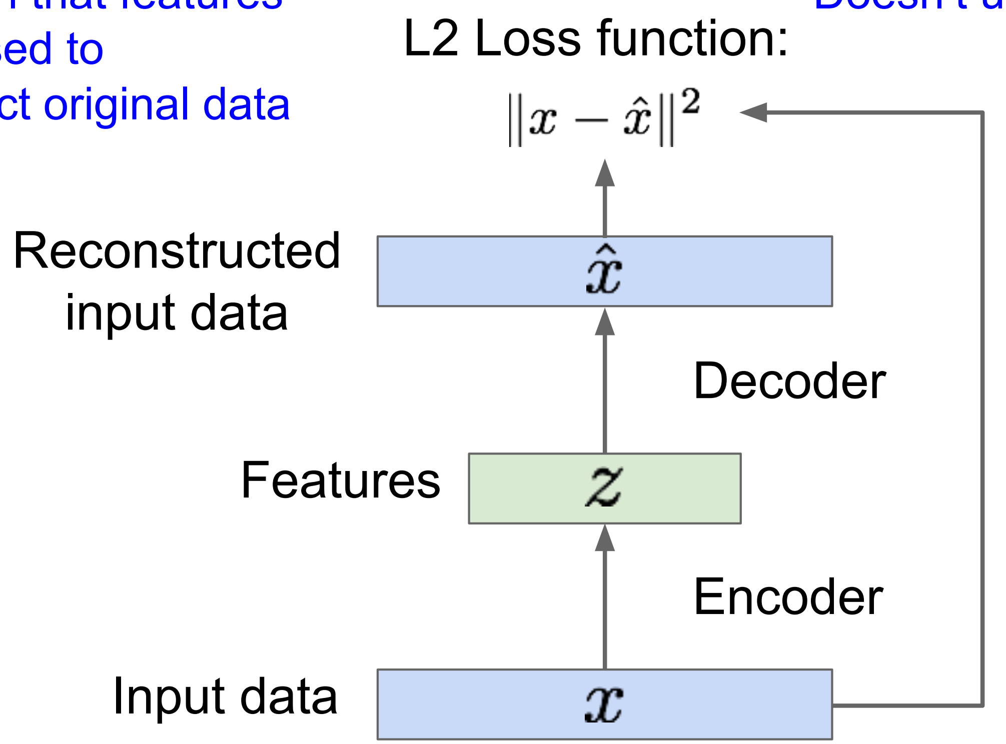


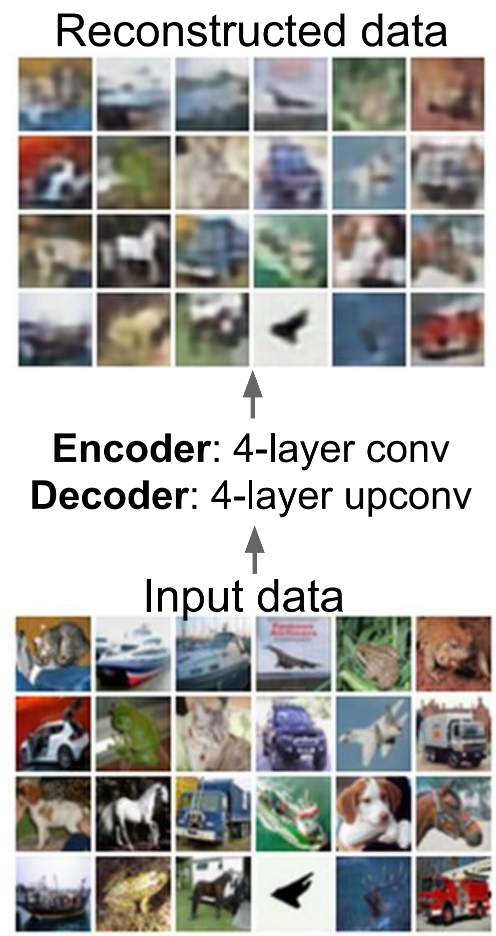


Some background first: Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

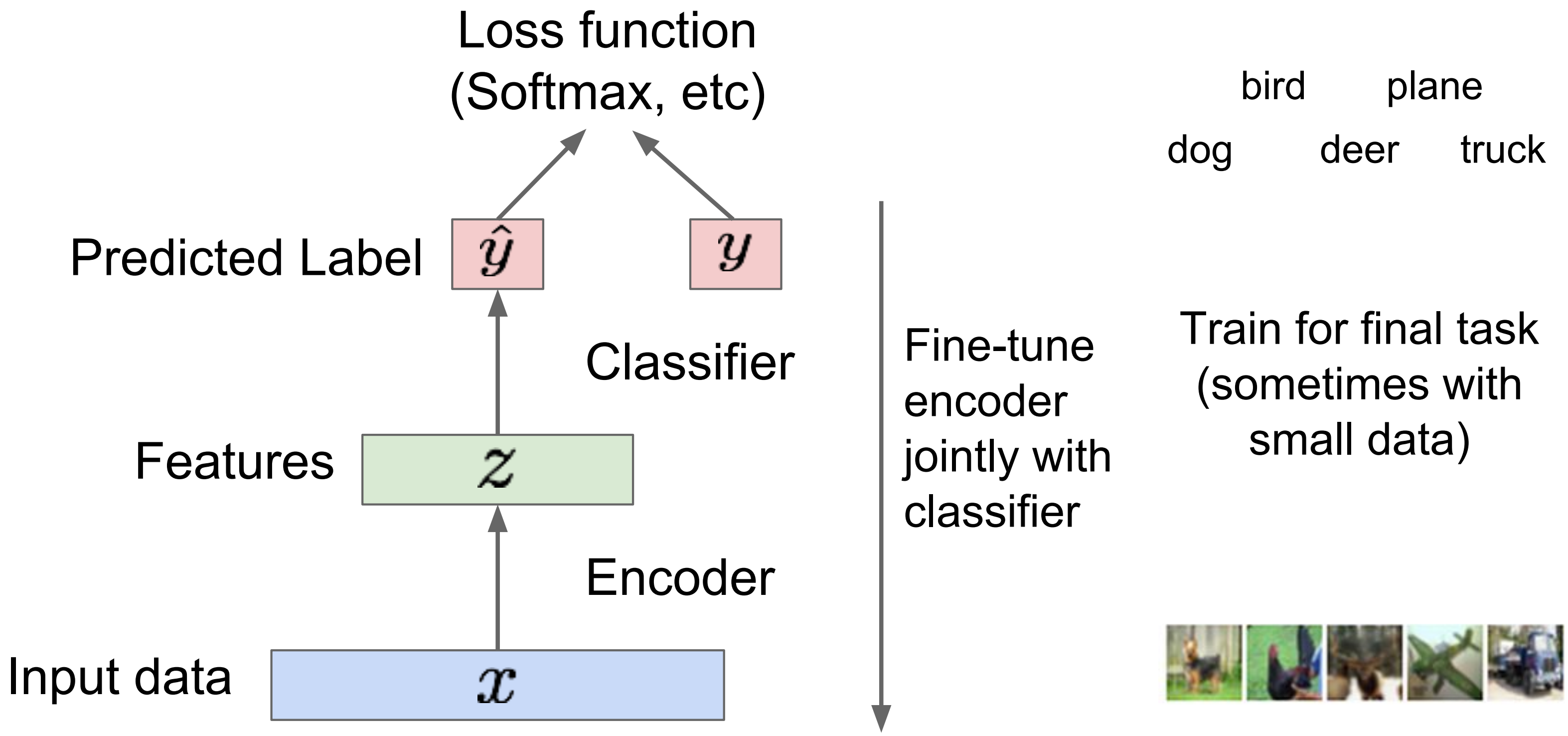




After training,
throw away decoder

Some background first: Autoencoders

Encoder can be used to initialize a **supervised** model



Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data. Can we generate new images from an autoencoder?

Variational Bayes summary

Bayes $\mathrm{p}(x|y) = \frac{\mathrm{p}(y|x)\mathrm{p}(y)}{\mathrm{p}(x)}$

Optimizing posterior $\mathrm{p}(x|y)$

You can also optimize the evidence (type II likelihood) $\mathrm{p}(\mathrm{y})$

Bishop Ch 10 Approximate inference
Variational inference

Observations $X = [x_1, \ldots, x_N]$

With latent parameter $Z = [z_1, \ldots, z_N]$

And probability $\mathrm{p}(\mathrm{X}, \mathrm{Z})$

We like to find an approximation to $\mathrm{p}(\mathrm{X}, \mathrm{Z})$ and the evidence $\mathrm{p}(\mathrm{Z})$

A good guess is a factorized distribution

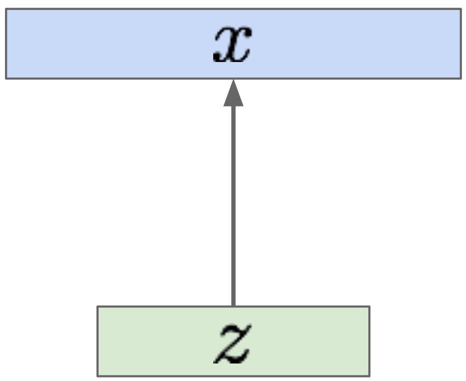
$\mathrm{p}(\mathrm{X}, \mathrm{Z})$= $\prod_{n=1}^{N} z_n$

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$



Sample from true prior $p_{\theta^*}(z)$

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian.

Conditional p(x|z) is complex (generates image) => represent with neural network

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

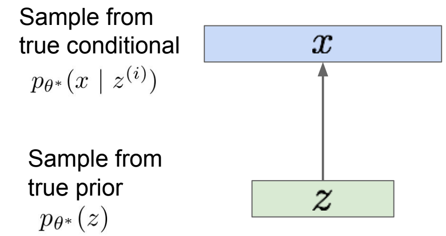
Q: What is the problem with this?

Intractable!

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from underlying unobserved (latent) representation **z**



Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Intractible to compute p(x|z) for every z!

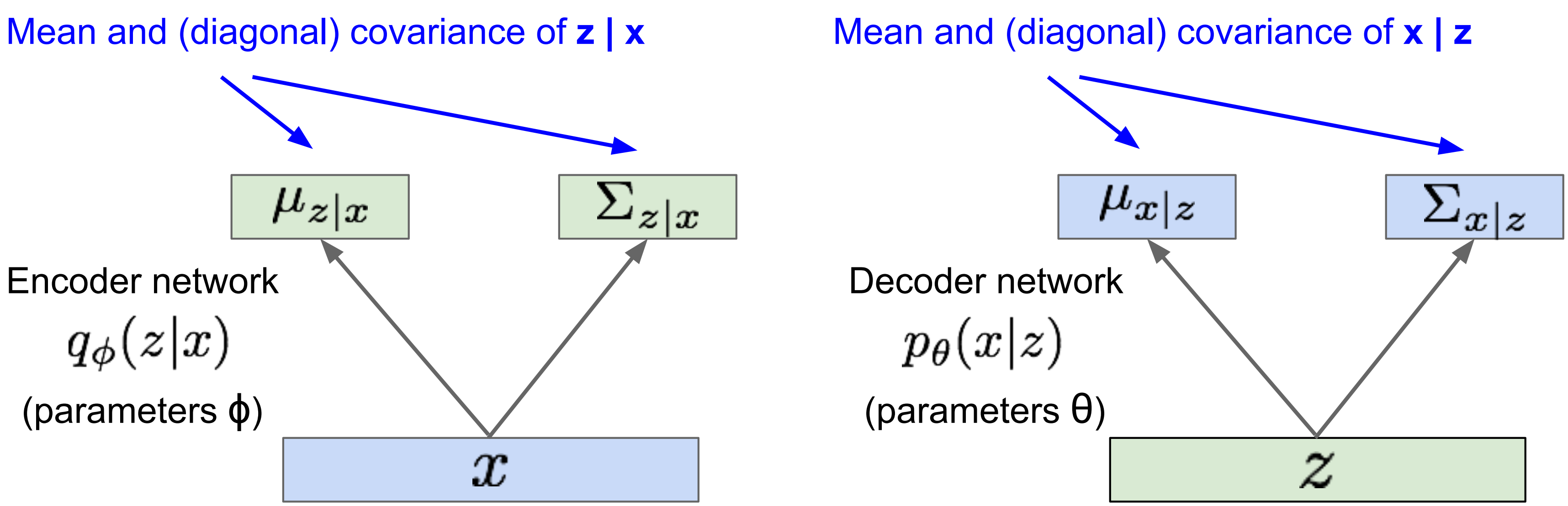
Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize

Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Reconstruct the input data

Make approximate posterior distribution close to prior

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

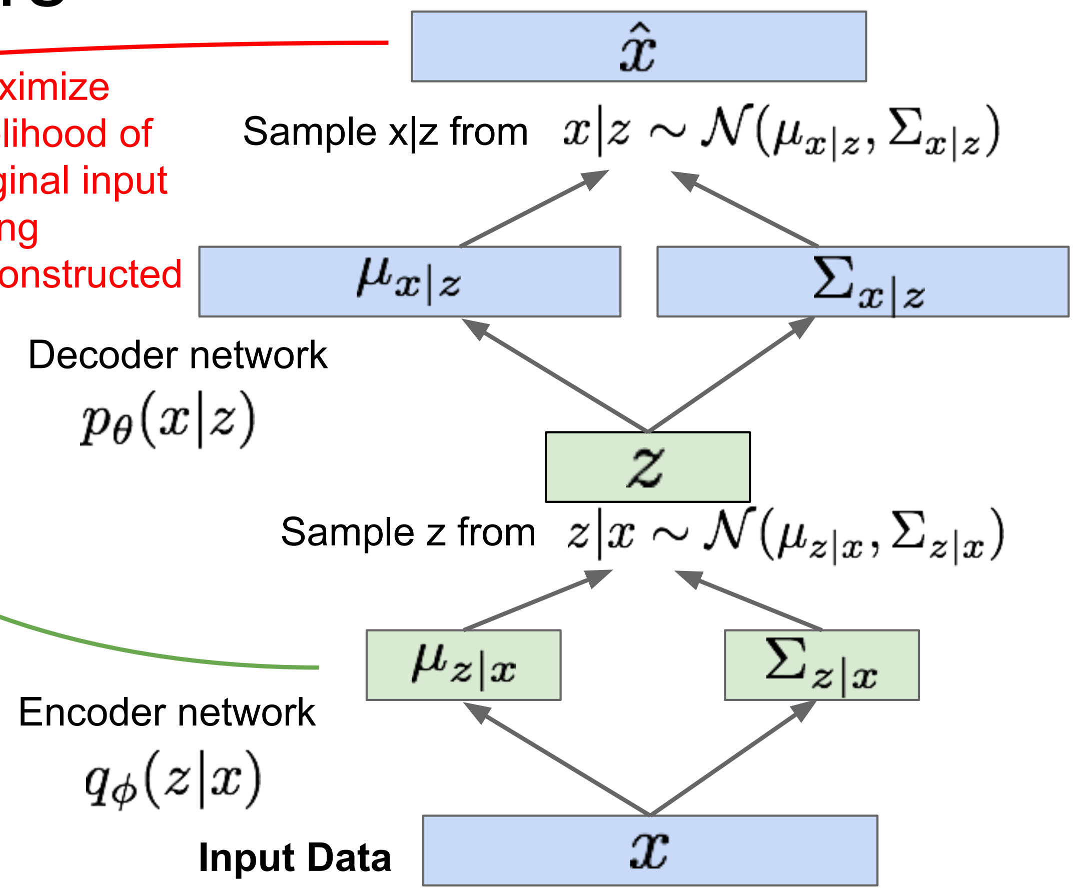
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

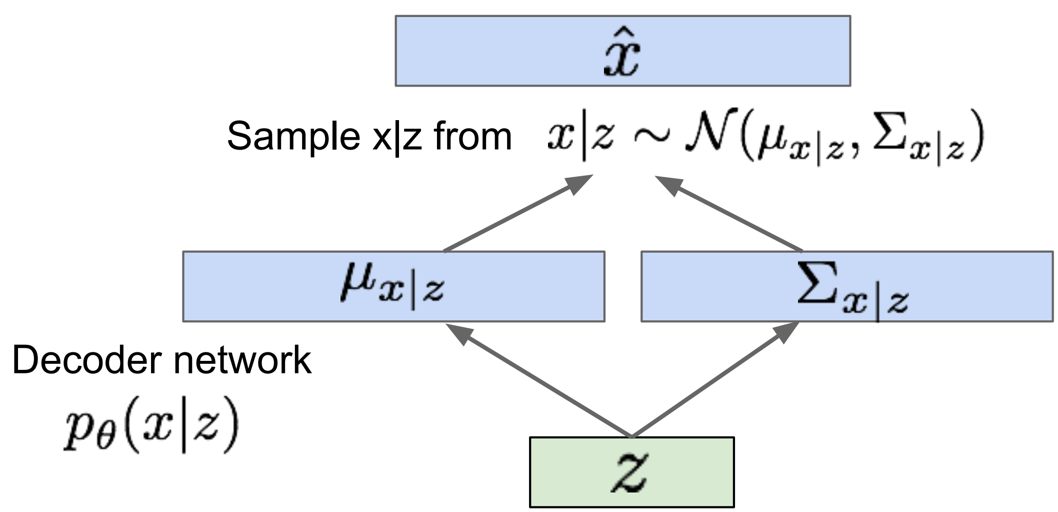
Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior



Variational Autoencoders: Generating Data!

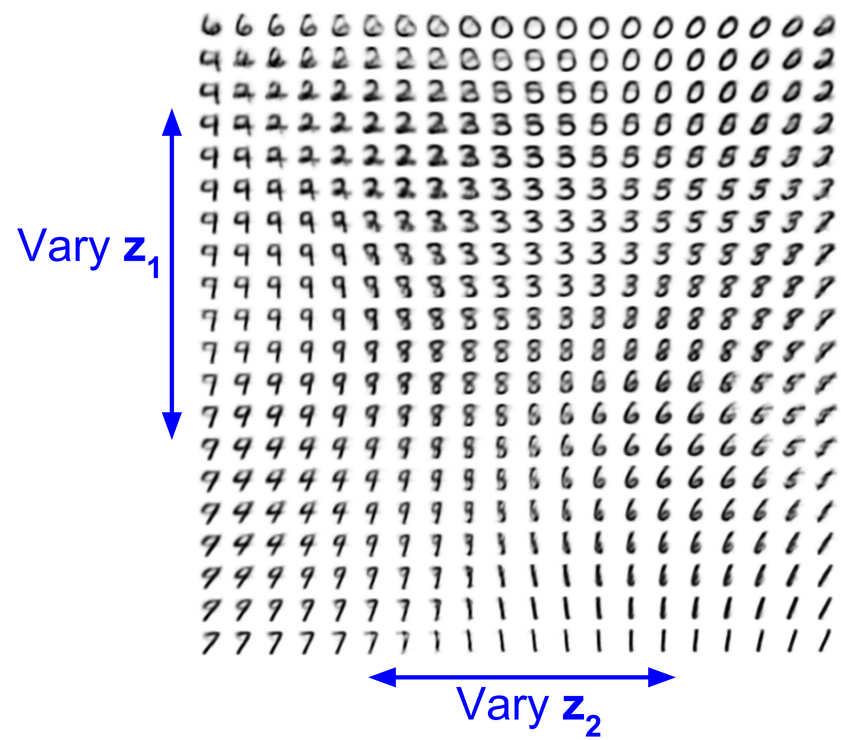
Use decoder network. Now sample z from prior!



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

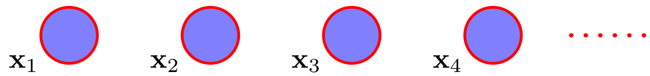
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Data manifold for 2-d **z**



Markov models, Bishop 13.1

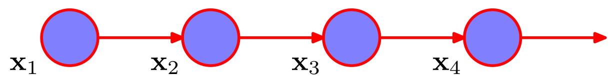
I.I.D model



Markov model

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}). \quad (13.1)$$

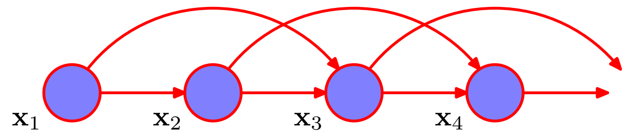
First order Markov chain



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_{n-1}). \quad (13.2)$$

Markov models, Bishop 13.1

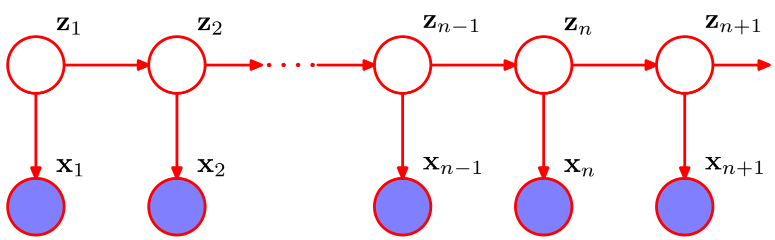
Second order Markov chain



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \prod_{n=3}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \qquad (13.4)$$

With K states, how many parameters?

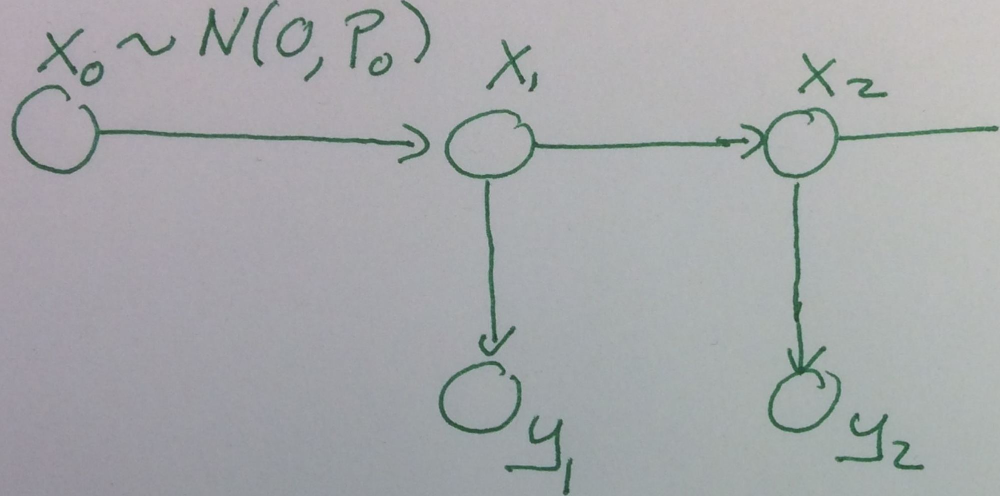
State space model



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n). \qquad (13.6)$$

Hidden Markov chain
Linear dynamical systems

State space model



State Eq.

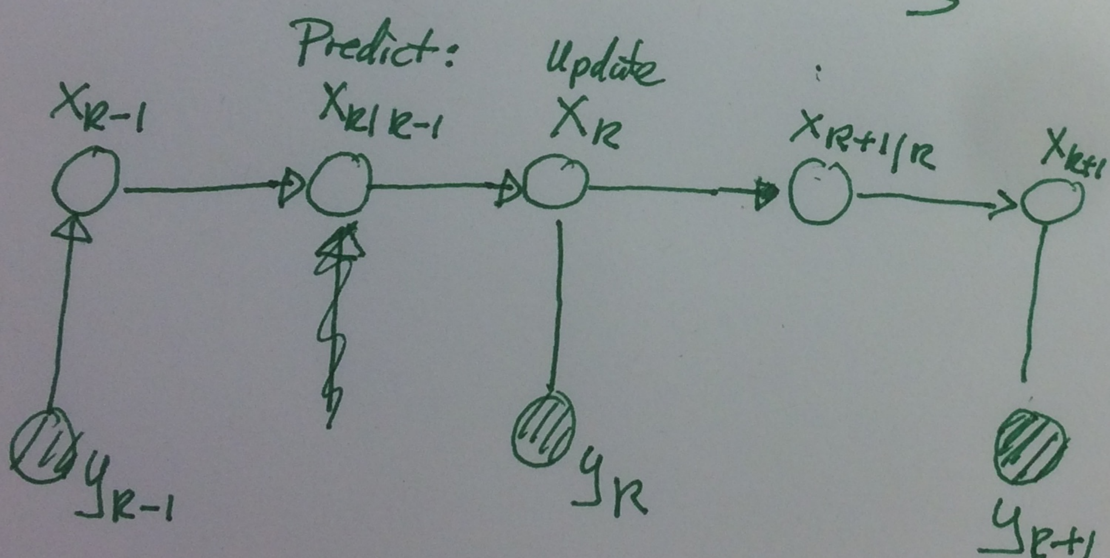
$$x_{k+1} = M_k x_k + \delta_k$$

Measurement Eq

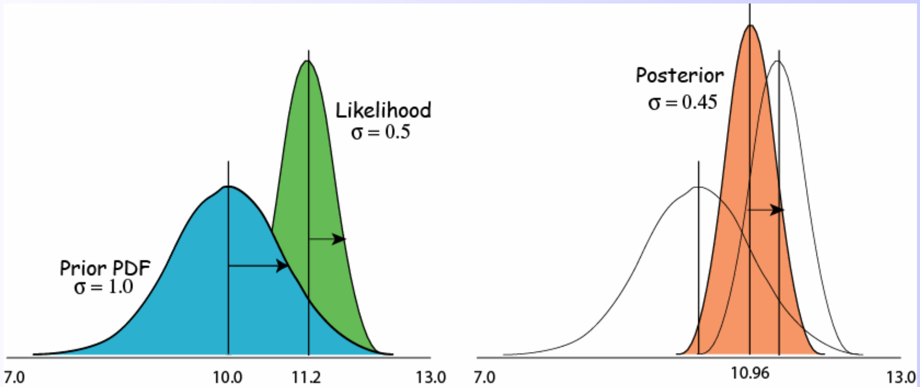
$$y_k = H_k x_k + v_k$$

$$\delta_k \sim N(0, Q_k)$$

$$v_k \sim N(0, R_k)$$



Product of Gaussians=Gaussian:



One data point problem

For the general linear inverse problem we would have

Prior: $$p(\boldsymbol{m}) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_o)^T C_m^{-1}(\boldsymbol{m}-\boldsymbol{m}_o)\right\}$$

Likelihood: $$p(\boldsymbol{d}|\boldsymbol{m}) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{d}-G\boldsymbol{m})^T C_d^{-1}(\boldsymbol{d}-G\boldsymbol{m})\right\}$$

Posterior PDF

$$\propto \exp\left\{-\frac{1}{2}[(\boldsymbol{d}-G\boldsymbol{m})^T C_d^{-1}(\boldsymbol{d}-G\boldsymbol{m}) + (\boldsymbol{m}-\boldsymbol{m}_o)^T C_m^{-1}(\boldsymbol{m}-\boldsymbol{m}_o)]\right\}$$

$$\propto \exp\left\{-\frac{1}{2}\left[\mathbf{m}-\hat{\mathbf{m}}\right]^T \mathbf{S}^{-1}\left[\mathbf{m}-\hat{\mathbf{m}}\right]\right\}$$

$$\mathbf{S}^{-1} = \mathbf{G}^T\mathbf{C}_d^{-1}\mathbf{G} + \mathbf{C}_m^{-1}$$

$$\hat{\mathbf{m}} = \left(\mathbf{G}^T\mathbf{C}_d^{-1}\mathbf{G} + \mathbf{C}_m^{-1}\right)^{-1}\left(\mathbf{G}^T\mathbf{C}_d^{-1}\mathbf{d} + \mathbf{C}_m^{-1}\mathbf{m}_0\right)$$

$$= \mathbf{m}_0 + \left(\mathbf{G}^T\mathbf{C}_d^{-1}\mathbf{G} + \mathbf{C}_m^{-1}\right)^{-1}\mathbf{G}^T\mathbf{C}_d^{-1}\left(\mathbf{d}-\mathbf{G}\mathbf{m}_0\right)$$

The Model

Consider the discrete, linear system,

$$\mathbf{x}_{k+1} = \mathbf{M}_k \mathbf{x}_k + \mathbf{w}_k, \quad k = 0, 1, 2, \ldots, \tag{1}$$

where

- $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector at time t_k
- $\mathbf{M}_k \in \mathbb{R}^{n \times n}$ is the state transition matrix (mapping from time t_k to t_{k+1}) or model
- $\{\mathbf{w}_k \in \mathbb{R}^n; k = 0, 1, 2, \ldots\}$ is a white, Gaussian sequence, with $\mathbf{w}_k \sim N(\mathbf{0}, \mathbf{Q}_k)$, often referred to as model error
- $\mathbf{Q}_k \in \mathbb{R}^{n \times n}$ is a symmetric positive definite covariance matrix (known as the model error covariance matrix).

Some of the following slides are from: Sarah Dance, University of Reading

The Observations

We also have discrete, linear observations that satisfy

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \quad k = 1, 2, 3, \ldots, \tag{2}$$

where

- $\mathbf{y}_k \in \mathbb{R}^p$ is the vector of actual measurements or observations at time t_k
- $\mathbf{H}_k \in \mathbb{R}^{n \times p}$ is the observation operator. Note that this is not in general a square matrix.
- $\{\mathbf{v}_k \in \mathbb{R}^p; k = 1, 2, \ldots\}$ is a white, Gaussian sequence, with $\mathbf{v}_k \sim N(\mathbf{0}, \mathbf{R}_k)$, often referred to as observation error.
- $\mathbf{R}_k \in \mathbb{R}^{p \times p}$ is a symmetric positive definite covariance matrix (known as the observation error covariance matrix).

We assume that the initial state, $\mathbf{x}_0$ and the noise vectors at each step, $\{\mathbf{w}_k\}$, $\{\mathbf{v}_k\}$, are assumed mutually independent.

The Prediction and Filtering Problems

We suppose that there is some uncertainty in the initial state, i.e.,

$$\mathbf{x}_0 \sim N(0, \mathbf{P}_0) \tag{3}$$

with $\mathbf{P}_0 \in \mathbb{R}^{n \times n}$ a symmetric positive definite covariance matrix.

The problem is now to compute an improved estimate of the stochastic variable $\mathbf{x}_k$, provided $\mathbf{y}_1, \ldots \mathbf{y}_j$ have been measured:

$$\widehat{\mathbf{x}}_{k|j} = \widehat{\mathbf{x}}_{k|y_1,\ldots,y_j}. \tag{4}$$

- When $j = k$ this is called the filtered estimate.
- When $j = k - 1$ this is the one-step predicted, or (here) the predicted estimate.

- The Kalman filter (Kalman, 1960) provides estimates for the linear discrete prediction and filtering problem.
- We will take a minimum variance approach to deriving the filter.
- We assume that all the relevant probability densities are Gaussian so that we can simply consider the mean and covariance.
- Rigorous justifcation and other approaches to deriving the filter are discussed by Jazwinski (1970), Chapter 7.

Prediction

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{M}_k \boldsymbol{x}_k + \boldsymbol{\delta}_k = \boldsymbol{x}'_k + \boldsymbol{\delta}_k$$

$$\boldsymbol{x}'_k \sim$$

$$\boldsymbol{\delta}_k \sim$$

$$\boldsymbol{x}_{k+1|k} \sim$$

$$\widehat{\boldsymbol{x}}_{k+1|k} =$$

$$\boldsymbol{P}_{k+1|k} =$$

Prediction step

We first derive the equation for one-step prediction of the mean using the state propagation model (1).

$$\begin{aligned}\widehat{\mathbf{x}}_{k+1|k} &= \mathbb{E}\left[\mathbf{x}_{k+1}|\mathbf{y}_1,\ldots\mathbf{y}_k\right], \\ &= \mathbb{E}\left[\mathbf{M}_k\mathbf{x}_k + \mathbf{w}_k\right], \\ &= \mathbf{M}_k\widehat{\mathbf{x}}_{k|k} \end{aligned} \tag{5}$$

The one step prediction of the covariance is defined by,

$$\mathbf{P}_{k+1|k} = \mathbb{E}\left[(\mathbf{x}_{k+1} - \widehat{\mathbf{x}}_{k+1|k})(\mathbf{x}_{k+1} - \widehat{\mathbf{x}}_{k+1|k})^T | \mathbf{y}_1, \ldots \mathbf{y}_k\right]. \tag{6}$$

Exercise: Using the state propagation model, (1), and one-step prediction of the mean, (5), show that

$$\mathbf{P}_{k+1|k} = \mathbf{M}_k \mathbf{P}_{k|k} \mathbf{M}_k^T + \mathbf{Q}_k. \tag{7}$$

Filtering Step

At the time of an observation, we assume that the update to the mean may be written as a linear combination of the observation and the previous estimate:

$$\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\widehat{\mathbf{x}}_{k|k-1}), \tag{8}$$

where $\mathbf{K}_k \in \mathbb{R}^{n\times p}$ is known as the Kalman gain and will be derived shortly.

But first we consider the covariance associated with this estimate:

$$\mathbf{P}_{k|k} = \mathbb{E}\left[(\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k})^T|\mathbf{y}_1, \ldots \mathbf{y}_k\right]. \tag{9}$$

Using the observation update for the mean (8) we have,

$$\begin{aligned}
\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k} &= \mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\widehat{\mathbf{x}}_{k|k-1}) \\
&= \mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k\widehat{\mathbf{x}}_{k|k-1}), \\
&\quad \text{replacing the observations with their model equivalent,} \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)(\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1}) - \mathbf{K}_k\mathbf{v}_k. \qquad (10)
\end{aligned}$$

Thus, since the error in the prior estimate, $\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1}$ is uncorrelated with the measurement noise we find

$$\begin{aligned}
\mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbb{E}\left[(\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1})^T\right](\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T \\
&\quad + \mathbf{K}_k\mathbb{E}\left[\mathbf{v}_k\mathbf{v}_k^T\right]\mathbf{K}_k^T. \qquad (11)
\end{aligned}$$

Simplification of the a posteriori error covariance formula

Using this value of the Kalman gain we are in a position to simplify the Joseph form as

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}. \tag{15}$$

Exercise: Show this.

Note that the covariance update equation is independent of the actual measurements: so $\mathbf{P}^{k|k}$ could be computed in advance.

Summary of the Kalman filter

Prediction step

Mean update: $\widehat{\mathbf{x}}_{k+1|k} = \mathbf{M}_k \widehat{\mathbf{x}}_{k|k}$

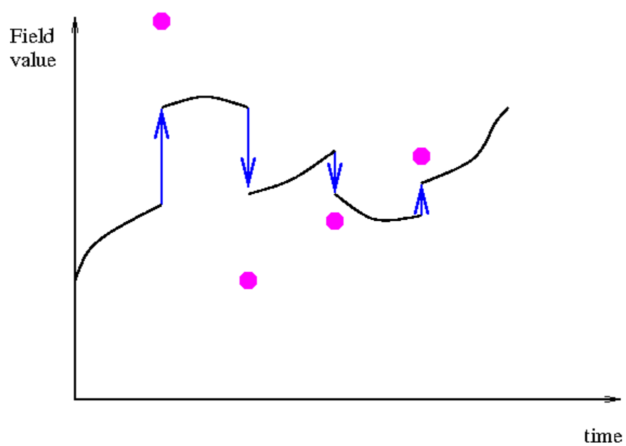
Covariance update: $\mathbf{P}_{k+1|k} = \mathbf{M}_k \mathbf{P}_{k|k} \mathbf{M}_k^T + \mathbf{Q}_k.$

Observation update step

Mean update: $\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k \widehat{\mathbf{x}}_{k|k-1})$

Kalman gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k)^{-1}$

Covariance update: $\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}.$



Bayes' Theorem for Gaussian Variables, Lecture 3

Given

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

we have

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b}, \mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b})+\boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \end{aligned}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$$

Bayes update

$$p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{x}_{k|k-1}) = p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k|k-1})$$

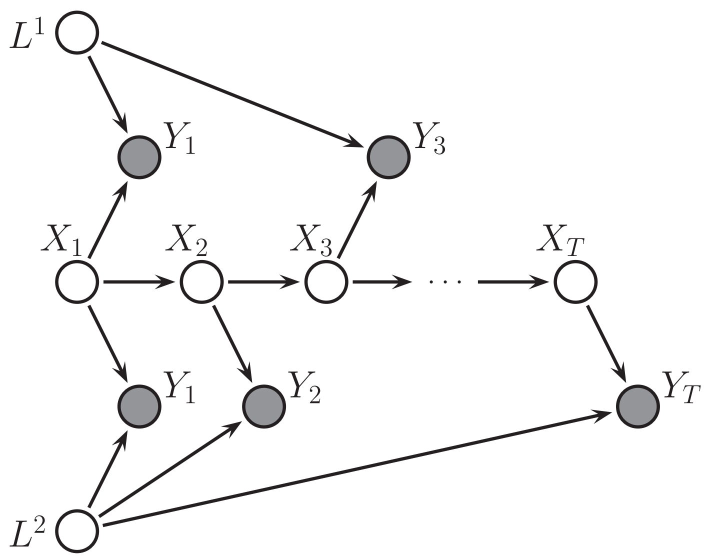
$$\boldsymbol{P}_k^{-1} =$$

$$\boldsymbol{P}_k = (\mathbf{I} - \mathbf{KH}_\mathrm{k})\boldsymbol{P}_{k|k-1}$$

$$\boldsymbol{K} = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$$

The Woodbury matrix identity is[4]

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1},$$



Graphical model underlying SLAM. L^i is the fixed location of landmark i, x_t is the robot location, and y_t is the observation. In this trace, the robot sees landmarks 1 and 2 at time 1, then just landmark 2, then just landmark 1, etc.

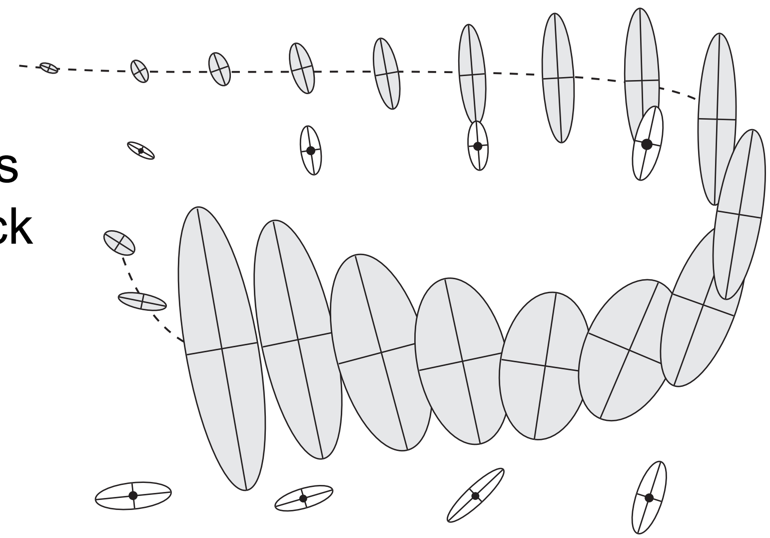


Illustration of the SLAM problem. (a) A robot starts at the top left and moves clockwise in a circle back to where it started. We see how the posterior uncertainty about the robot's location increases and then decreases as it returns to a familar location, closing the loop. If we performed smoothing, this new information would propagate backwards in time to disambiguate the entire trajectory.

Constant velocity model

Using a constant velocity CV track model for the source, the the state equation is given by

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} d_{k+1} \\ v_{k+1} \end{bmatrix} = \boldsymbol{M}_k \boldsymbol{x}_k + \boldsymbol{B}_k \varepsilon_k = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta^2 \\ 1 \end{bmatrix} \varepsilon_k$$

Note that the noise term on velocity is now an acceleration in the location-term.

Predict N steps ahead

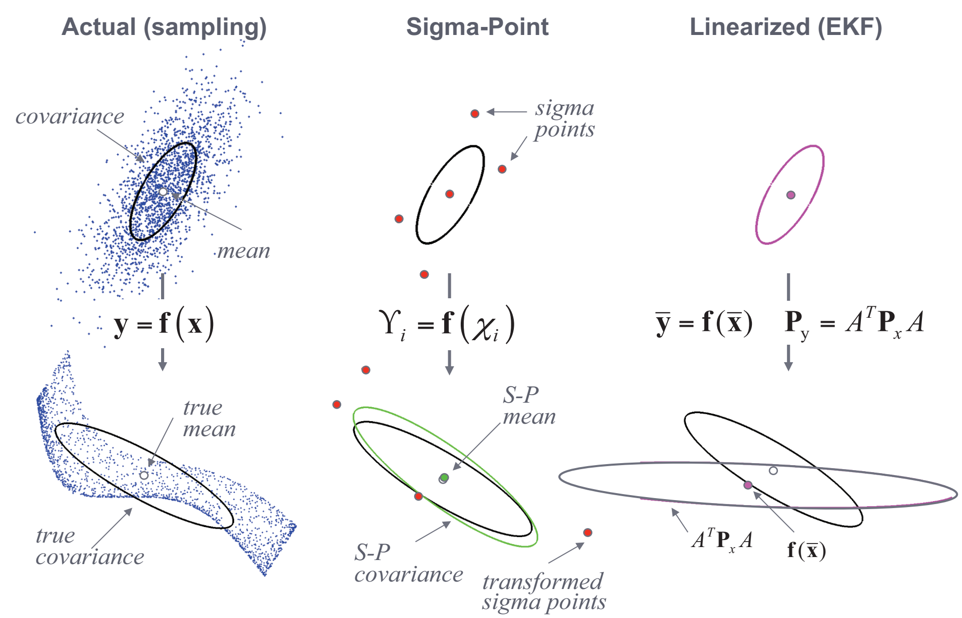
SLAM (Simultaneous Location and Mapping)

Kalman smoother

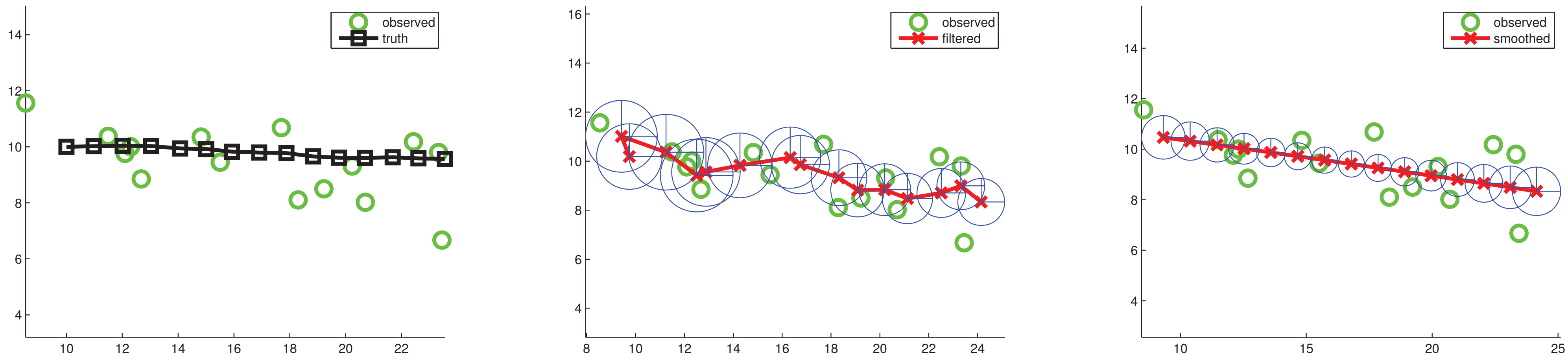
RLS (Recursive least squares)

Advanced KF:

- Ensample KF (EnKF)   non Gaussian
- Extended KF (EKF) non-linear
- Unscented KF (UKF) well chosen c
- … Particle Filter Nonlinear, non Ga



Kalman smoother



(a) (b) (c)

Figure 18.1 Kalman filtering and smoothing. (a) Observations (green cirles) are generated by an object moving to the right (true location denoted by black squares). (b) Filtered estimated is shown by dotted red line. Red cross is the posterior mean, blue circles are 95% confidence ellipses derived from the posterior covariance. For clarity, we only plot the ellipses every other time step. (c) Same as (b), but using offline Kalman smoothing. Figure generated by kalmanTrackingDemo.

Carrying On…

The book by Murphy has more details on ML.

Many interesting courses online and at UCSD.

Lots of opportunities also outside CS.

For next course, more class interaction (phone questions), more cody home work, **physics** better integrated.

Graphical models better integrated, Gaussian processes, sequential state models.

Murphy: "This books adopts the view that the best way to make machines that can learn from data is to use the *tools of probability theory*, which has been the mainstay of statistics and engineering for centuries. "

NOT USED

4:15-4:30: Bruce Cornuelle, Scripps Institution of Oceanography

"A less grand challenge: How can we merge machine learning with data assimilation? "

Peter: I propose that if data assimilation is posed "correctly" it is already machine leaning. Anyway looking forward to your talk.

Bruce: I agree, but most machine learning I know about doesn't build in prior known dynamics or let you understand what the machine has learned. If you have examples to the contrary, please give me references. I know about the attempts to "invert" the networks, though.

I also want to know the pdfs that the machine learning technique is optimal for, both in the data and the unknowns, in the way that L2 is optimal for gaussians and L1 is optimal for exponentials.