# 2018 ECE228 Poster session

**Project discussion, 22 May: Mandatory but ungraded.**

**Thanks for doing this**

**June 4, 6pm** deadline for submitting poster for printing (pdf preferred). TAs have to print 43 posters. Dropbox link or email TA

https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

**June 5, 5-8pm Atkinson Hall: Poster and Pizza. Easels available.**

**June 15, 8am** deadline for submitting report and code. (we have 43 reports to read in 3 days!) use dropbox link or email TA

https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

Evaluation

Report:30%

Poster: 10% (as displayed)

Code: 10% (should run automatically)

# 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

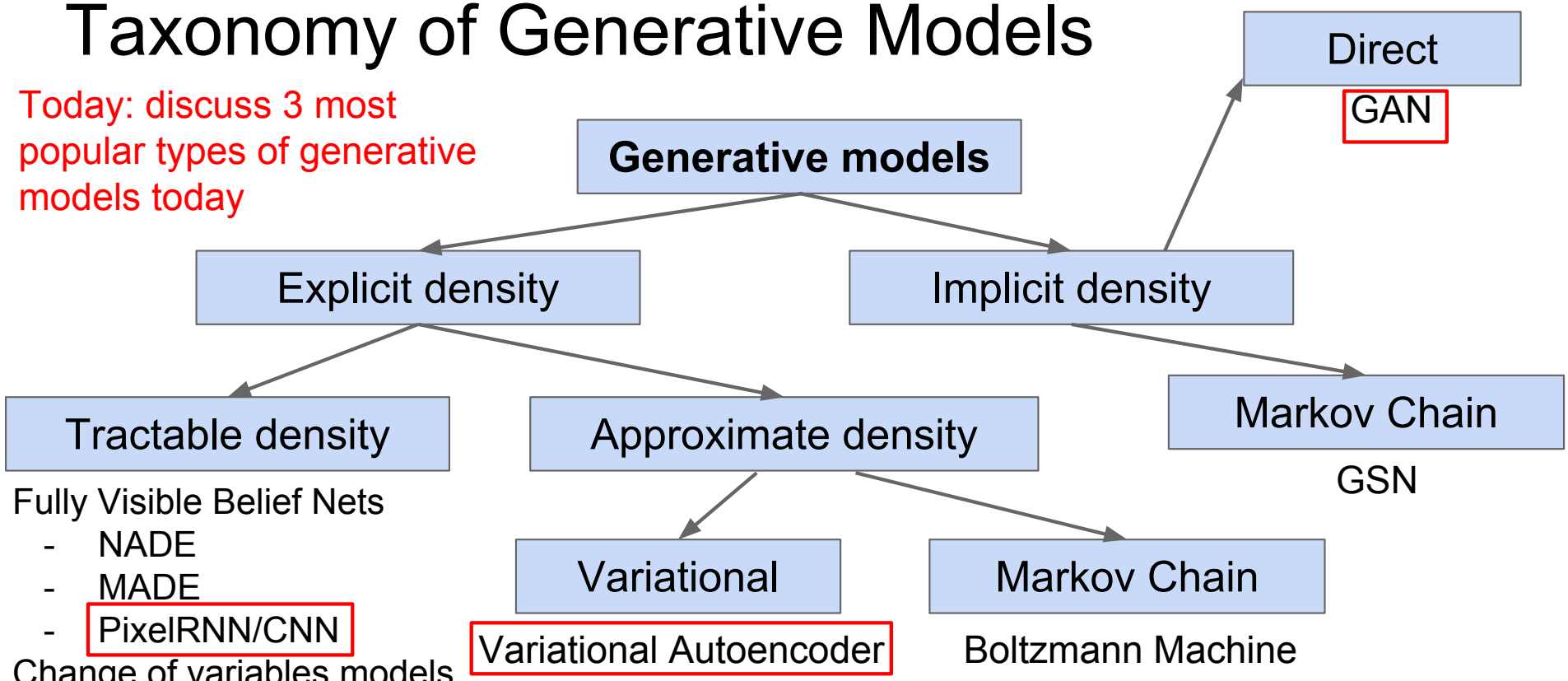
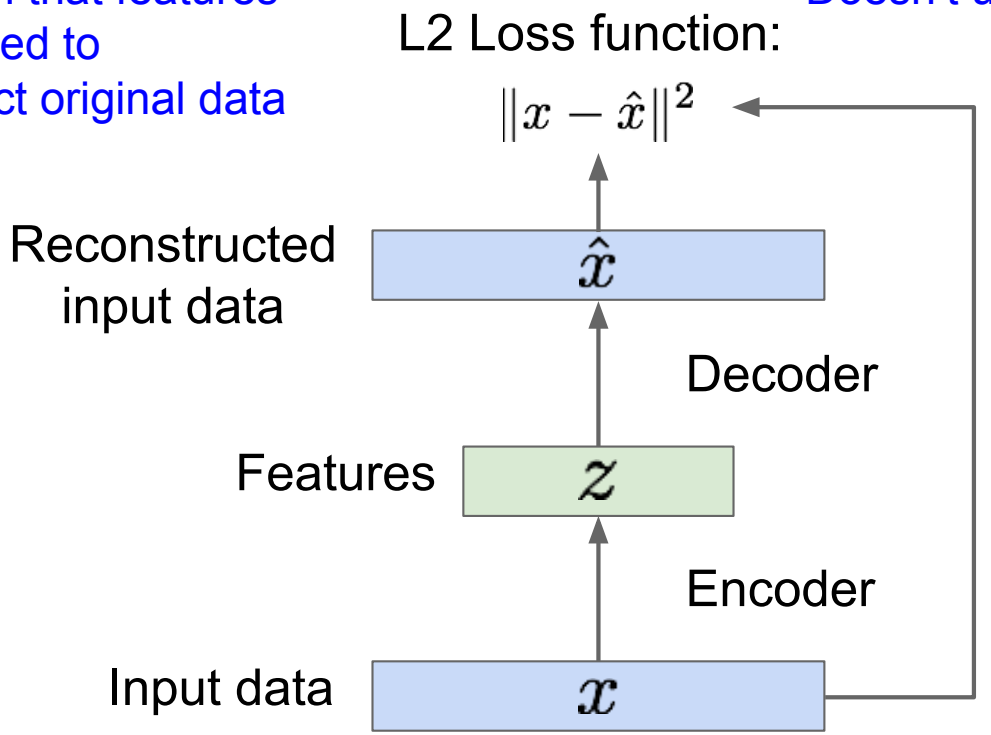


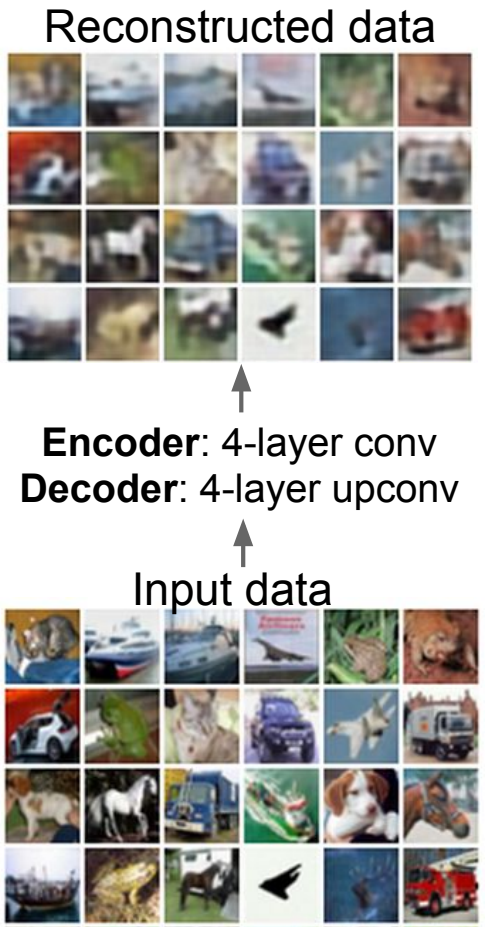

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# Some background first: Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!




After training,
throw away decoder

# 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

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

# So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

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

VAEs define intractable density function with latent **z**:

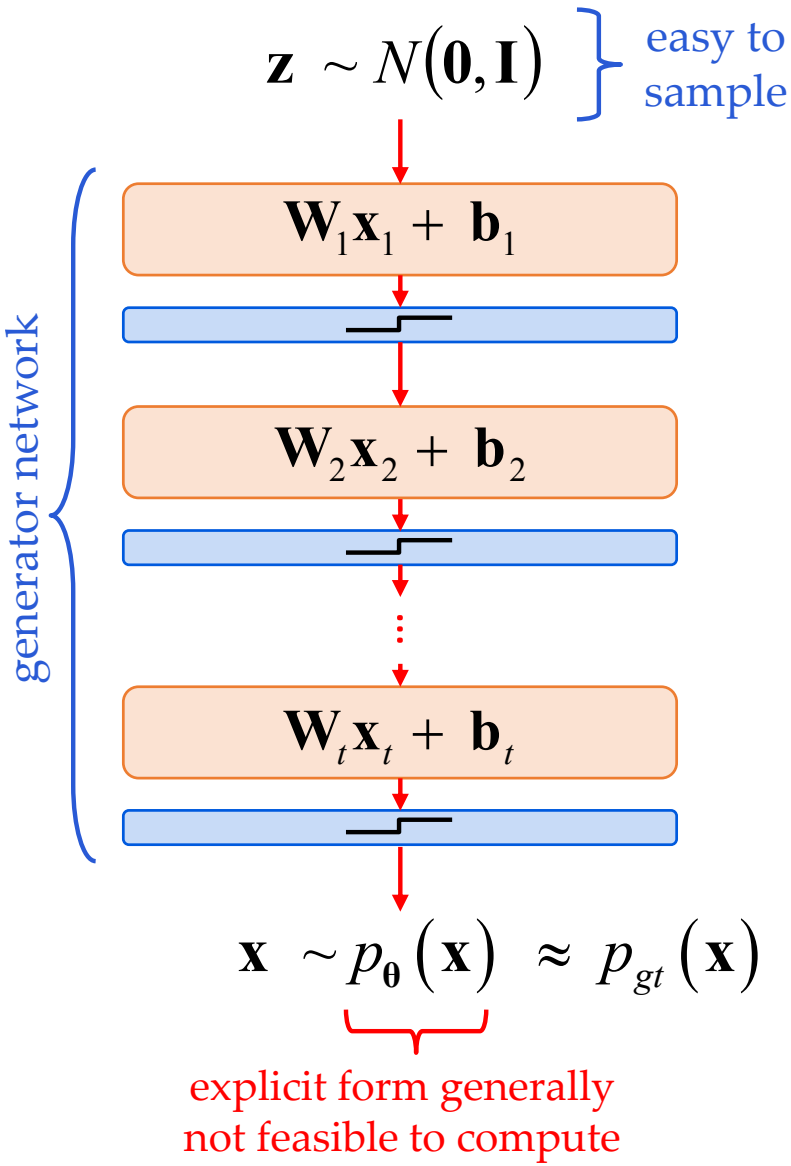
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead

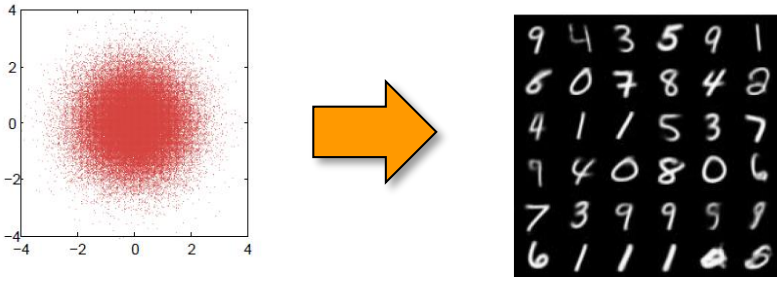
What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

# Implicit Deep Generative Modeling




## Example



## Popular Example

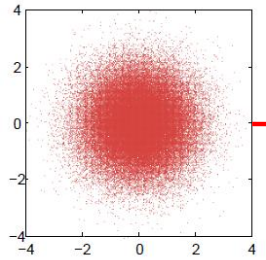
**Generative Adversarial Networks**:

- Based on game theory, Nash equilibrium
- 8679 citations [Goodfellow et al., 2014]

# Generative Adversarial Networks (GANs)

noise samples

$\left\{\mathbf{z}^{(j)}\right\}_{j=1}^{m}, \ \mathbf{z}^{(j)} \sim N\left(\mathbf{z} \mid \mathbf{0}, \mathbf{I}\right)$



Generator Network

$G_{\boldsymbol{\theta}}$

fake samples

real samples

$\left\{\mathbf{x}^{(i)}\right\}_{i=1}^{n}, \mathbf{x}^{(i)} \in \mathbb{R}^{d} \sim p_{gt}\left(\mathbf{x}\right)$

Discriminator Network → real/fake

$D_{\boldsymbol{\varphi}} : \mathbb{R}^{d} \rightarrow (0,1)$

Binary classification:

$D_{\boldsymbol{\varphi}}\left(\underbrace{\mathbf{x}^{(i)}}_{\text{real}}\right) \approx 1, \quad D_{\boldsymbol{\varphi}}\left[G_{\boldsymbol{\theta}}\left(\underbrace{\mathbf{z}^{(j)}}_{\text{fake}}\right)\right] \approx 0$

**Basic GAN objective (cross-entropy-based):**

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \ \mathrm{E}_{p_{gt}(\mathbf{x})}\left[\log D_{\boldsymbol{\varphi}}\left(\mathbf{x}\right)\right] + \mathrm{E}_{N(\mathbf{z}|\mathbf{0},\mathbf{I})}\left[\log\left(1 - D_{\boldsymbol{\varphi}}\left[G_{\boldsymbol{\theta}}\left(\mathbf{z}\right)\right]\right)\right]$$

expectations approximated with samples

[Goodfellow et al., 2014]

# GAN Strengths

State-of-the-art GAN models generate highly realistic samples, e.g., StyleGAN [Karras et al, 2019]:

**real**

**fake**

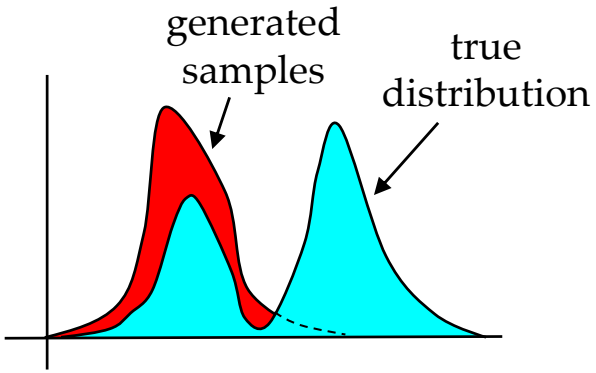
Examples from http://www.whichfaceisreal.com/

# GAN Weaknesses

- Training involves potentially unstable minimax problem, iterations may diverge, be sensitive to tuning.

[Lucic et al., 2018]

- Can be susceptible to mode collapse:




**low sample diversity**

[Arora and Zhang, 2017]

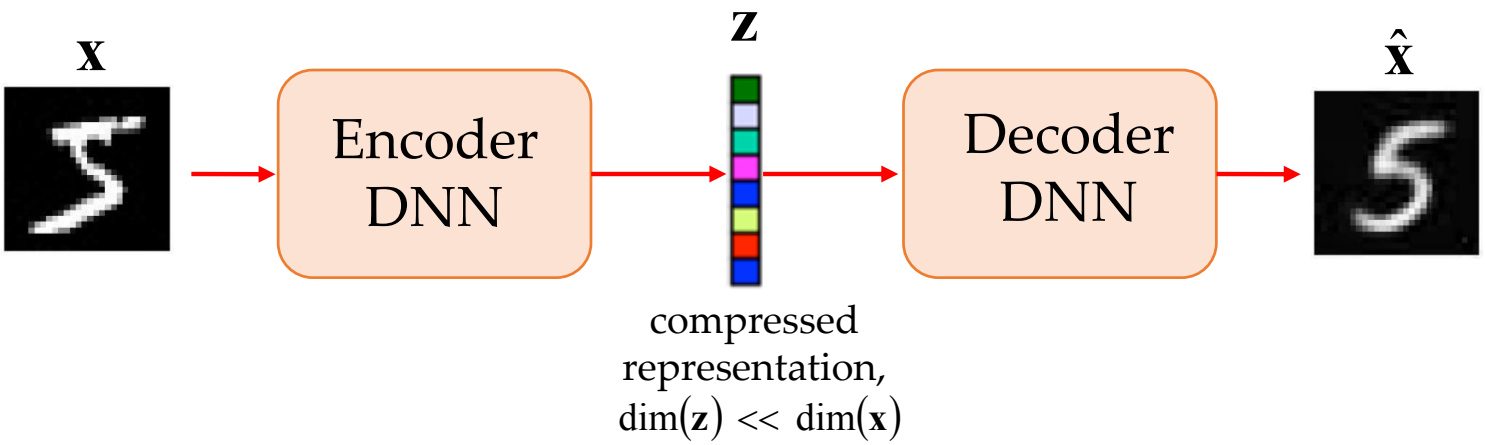
- No explicit density estimate $p_{\theta}(\mathbf{x}) \approx p_{gt}(\mathbf{x})$, cannot infer the latent code that produced a sample:

$$p_{\theta}(\mathbf{z} \mid \mathbf{x}) \; ?$$

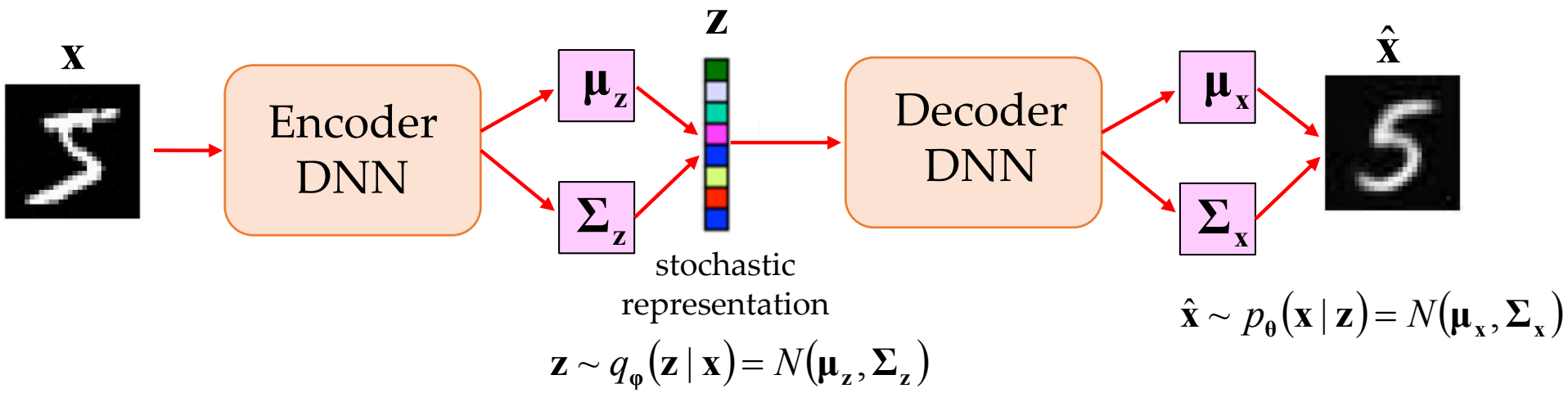
cannot compute low-dimensional representation

# Comparison with an Autoencoder

## Autoencoder (AE):




## VAE:



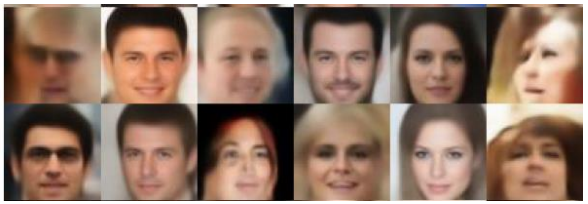
# Variational Autoencoders
## (details in Part II)

**Advantages:**

- Less prone to mode collapse than GANs, more stable training.
- Provides explicit estimate of latent distribution $p_{\theta}(\mathbf{z}|\mathbf{x})$; many applications in representation learning.
- Natural generalization of dimensionality reduction tools in common use for signal processing (Part III).

**Disadvantages:**

- Optimizes a bound on the data likelihood, not exact likelihood (but conditions for when bound is tight discussed in Part IV).
- Generated samples usually inferior to GANs ...

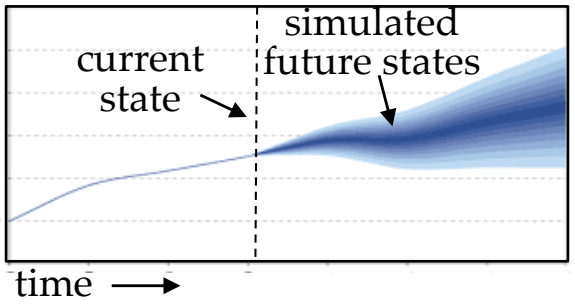


… although improvements possible (Part IV).

# Representative Applications

## Generative models in general:

- Model-based reinforcement learning:




[Finn et al., 2016]

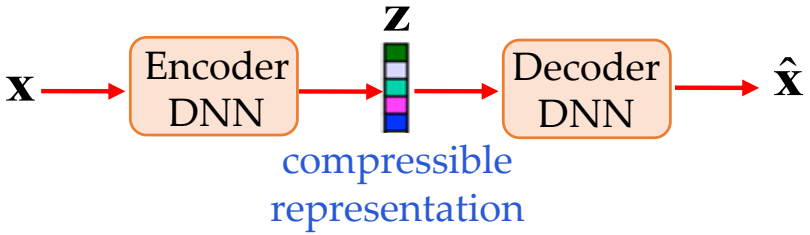
- Image-to-image translation:

[Isola et al., 2016]

- Many more, a generic unsupervised learning tool

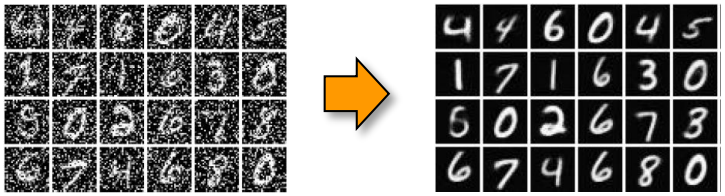
## VAEs in particular:

- Compression:



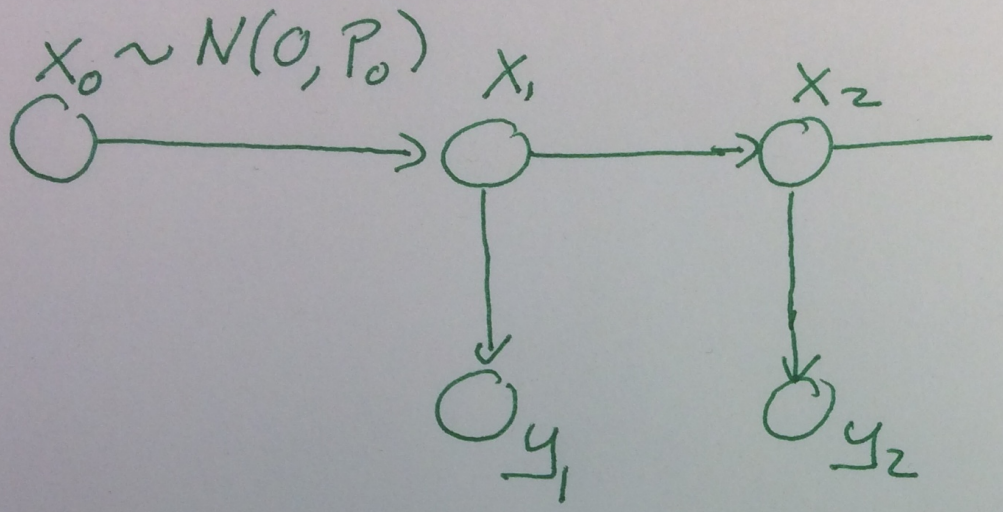

[Ballé et al., 2018]

- Data cleaning, outlier removal:



[Dai et al., 2018]

# State space model




State Eq.

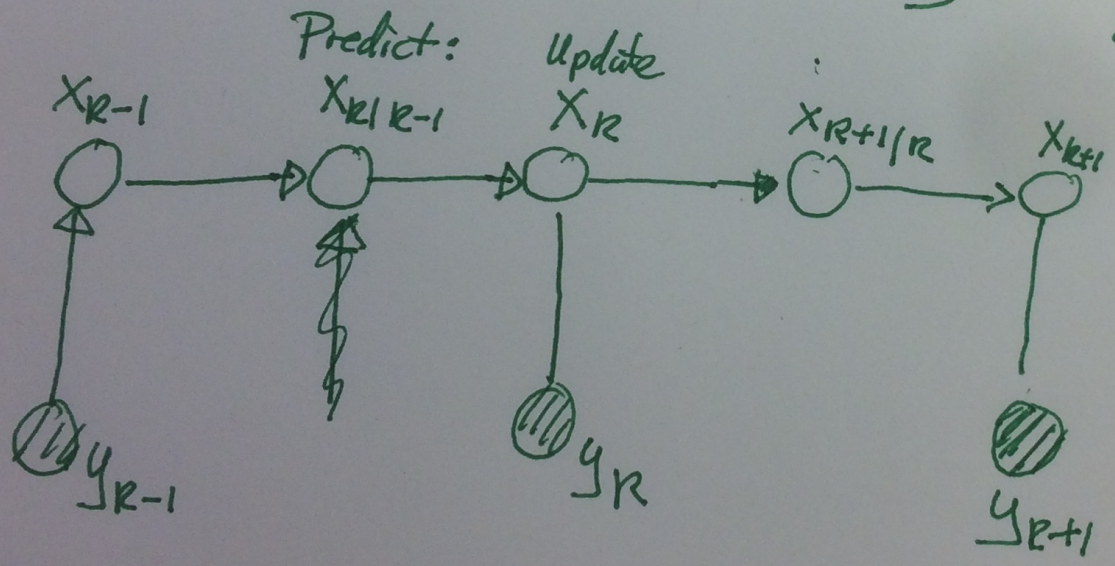
$$x_{k+1} = M_k x_k + \delta_k$$

Measurement Eq

$$y_k = H_k x_k + v_k$$

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

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



# 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.

# Summary of the Kalman filter

## Prediction step

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

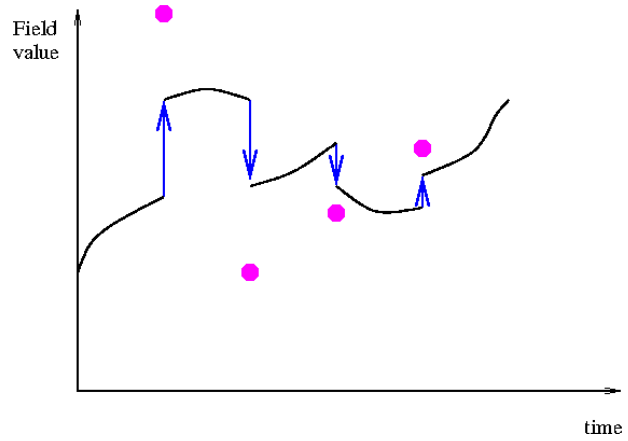
Covariance update: $\mathbf{P}_{k+1|k} = \mathbf{M}_k \mathbf{P}_{k|k} \mathbf{M}_k^T + \mathbf{Q}_k.$

## 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

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right)$$

we have

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{L}^{-1}\right)$$

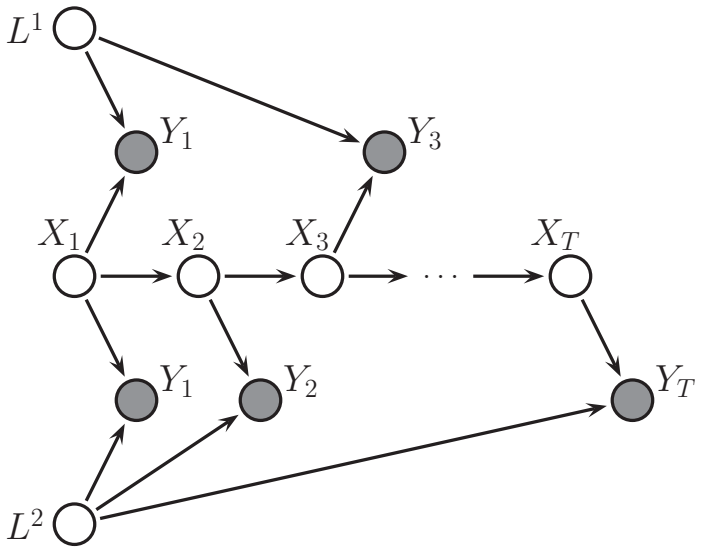
$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b}, \mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}})$$

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

where

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

# Simultaneous location and mapping (SLAM)




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.

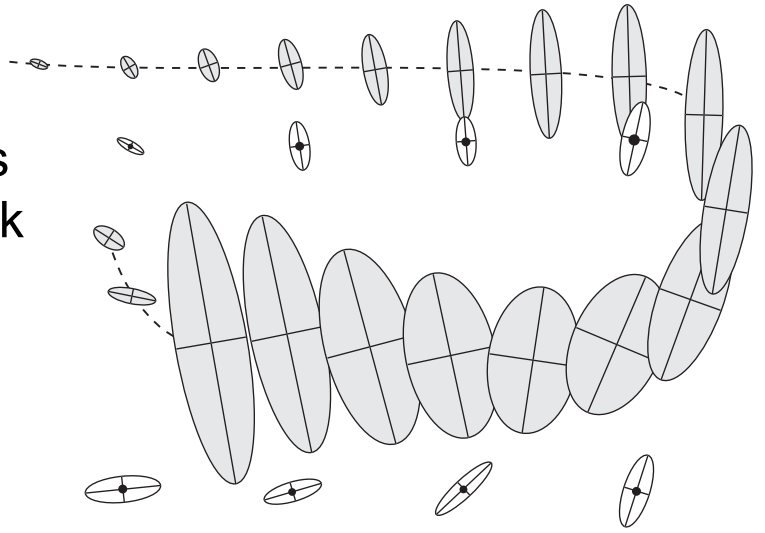


Illustration of the SLAM problem. (a) A robot starts at the top left and moves clockwise in a circle back to where it started. We see how the posterior uncertainty about the robot's location increases and then decreases as it returns to a familar location, closing the loop. If we performed smoothing,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

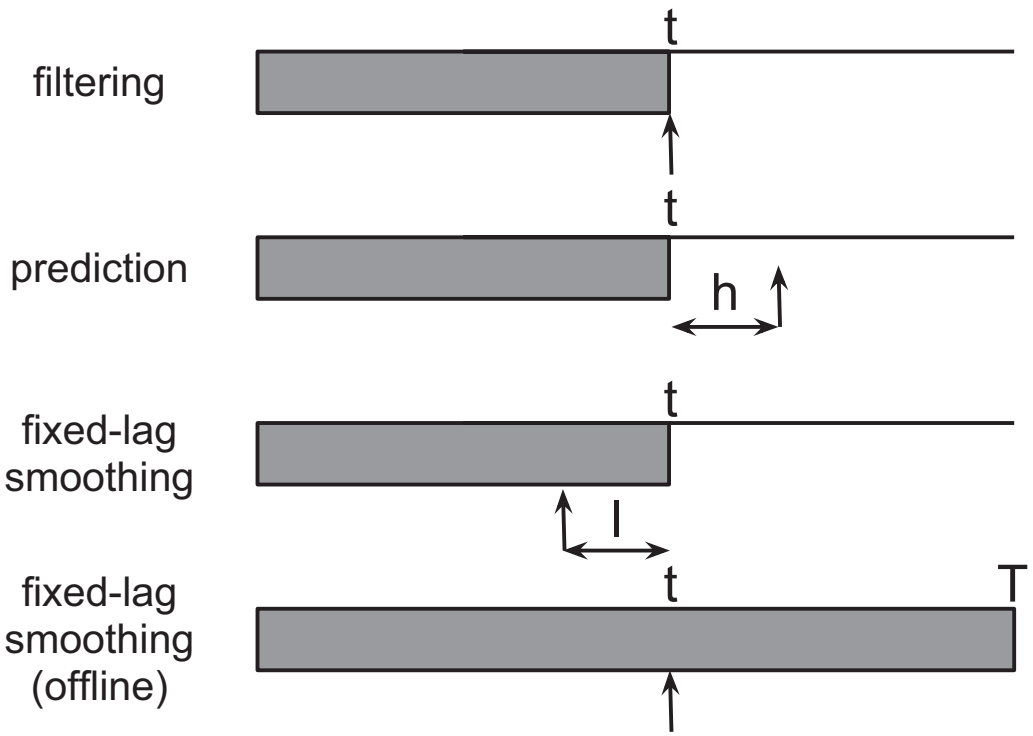
# Constant velocity model

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.

# State space model interference



# Kalman smoother

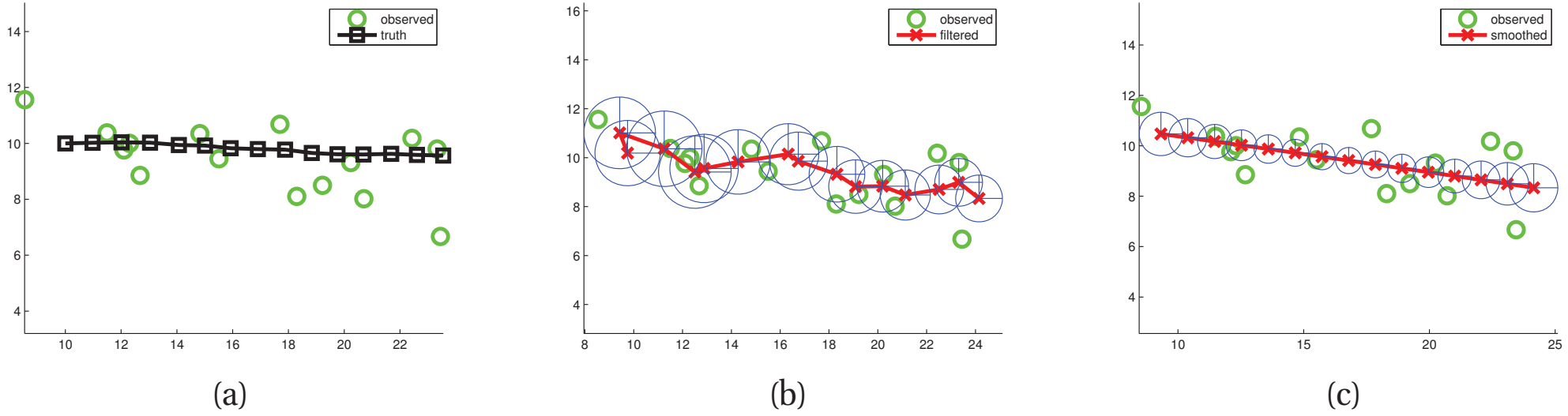



**Figure 18.1** Kalman filtering and smoothing. (a) Observations (green cirles) are generated by an object moving to the right (true location denoted by black squares). (b) **Filtered estimated** is shown by dotted red line. Red cross is the posterior mean, blue circles **are 95% confidence** ellipses 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.

Predict N steps ahead

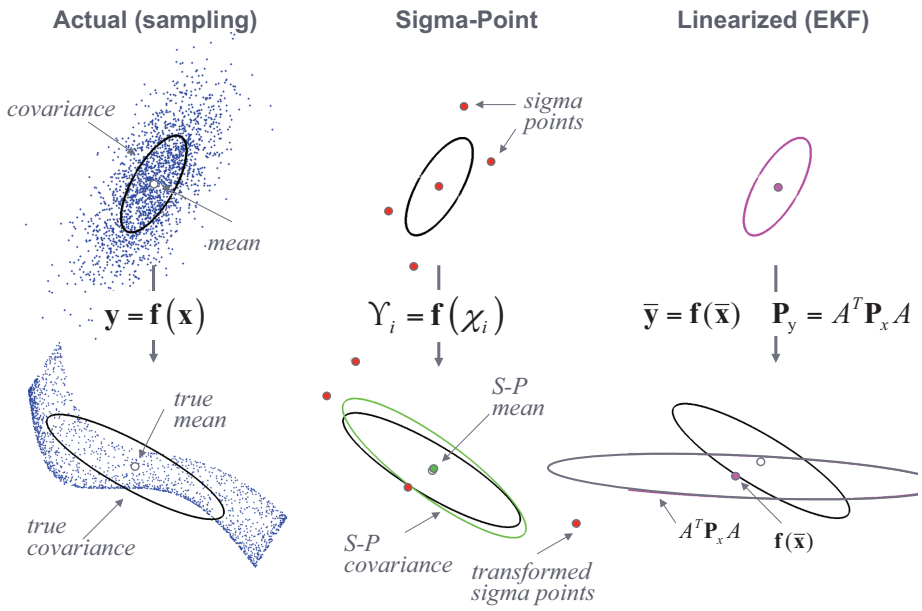
SLAM (Simultaneous Location and Mapping)

Kalman smoother

RLS (Recursive least squares)

Advanced KF:

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

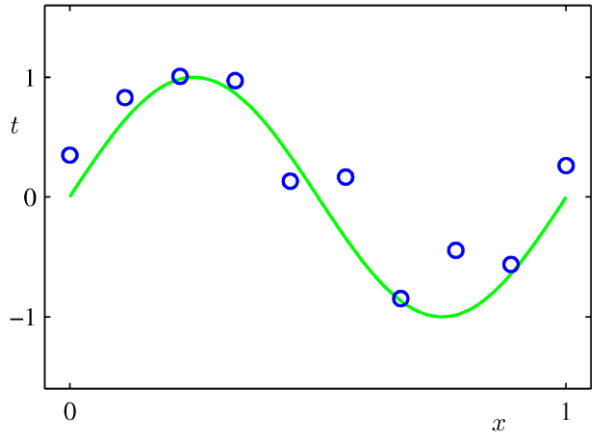


# **Linear regression:** Linear Basis Function Models (1)

Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

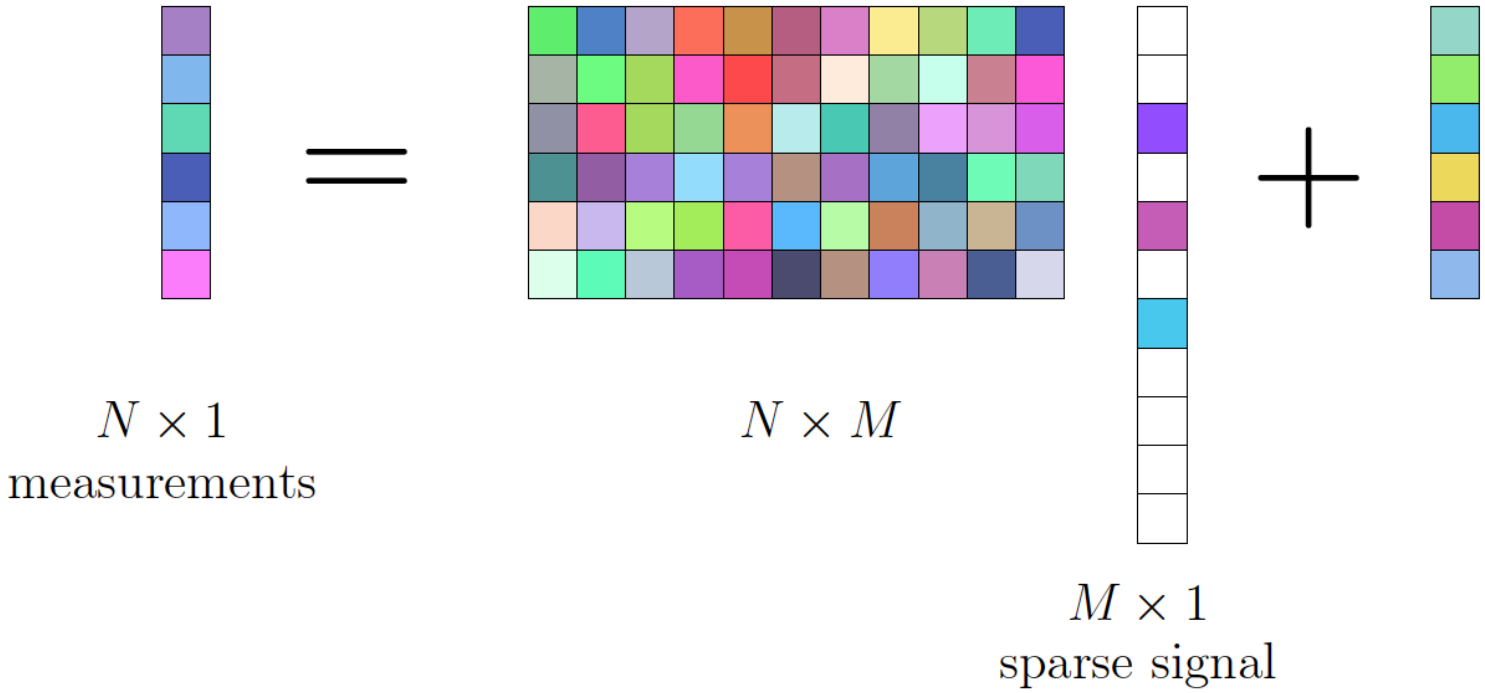
- where $\phi_j(x)$ are known as *basis functions*.
- Typically, $\phi_0(x) = 1$, so that $w_0$ acts as a bias.
- Simplest case is linear basis functions: $\phi_d(x) = x_d$.




$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

# Sparse model

Model : $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$ , $\mathbf{x}$ is sparse



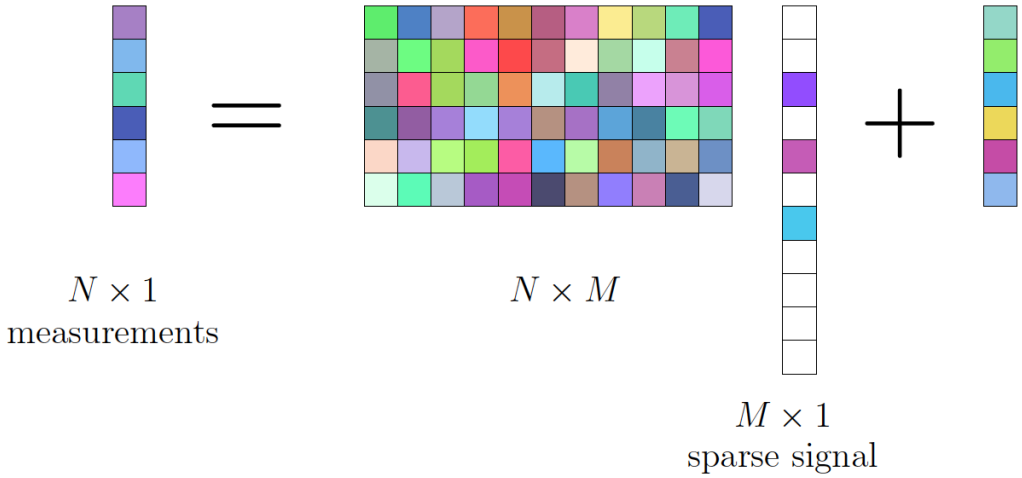

- y : measurements, A : dictionary
- n : noise, x : sparse weights
- Dictionary (A) − either from physical models or learned from data (dictionary learning)

# Sparse processing

- Linear regression (with sparsity constraints)
  - An underdetermined system of equations has many solutions
  - Utilizing x is sparse it can often be solved
  - This depends on the structure of A (RIP – Restricted Isometry Property)
- Various sparse algorithms
  - Convex optimization (Basis pursuit / LASSO / $L_1$ regularization)
  - Greedy search (Matching pursuit / OMP)
  - Bayesian analysis (Sparse Bayesian learning / SBL)
- Low-dimensional understanding of high-dimensional data sets
- Also referred to as compressive sensing (CS)

# Different applications, but the same algorithm

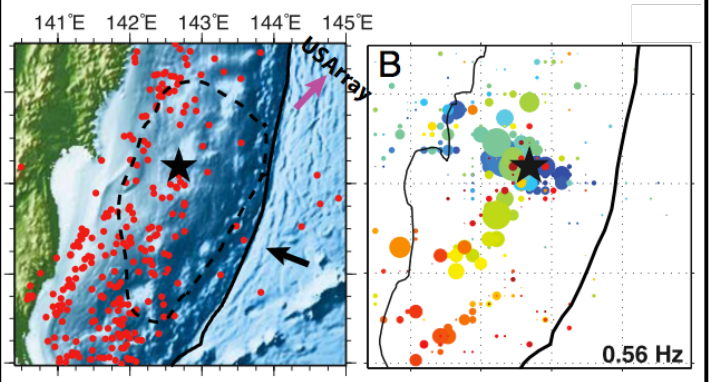
Model : $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$, $\mathbf{x}$ is sparse




| y | A | x |
|---|---|---|
| Frequency signal | DFT matrix | Time-signal |
| Compressed-Image | Random matrix | Pixel-image |
| Array signals | Beam weight | Source-location |
| Reflection sequence | Time delay | Layer-reflector |

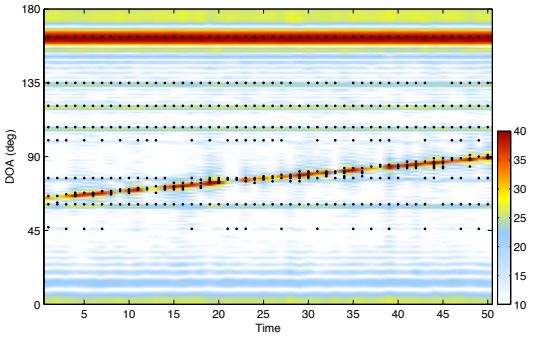
# CS approach to geophysical data analysis

## CS of Earthquakes



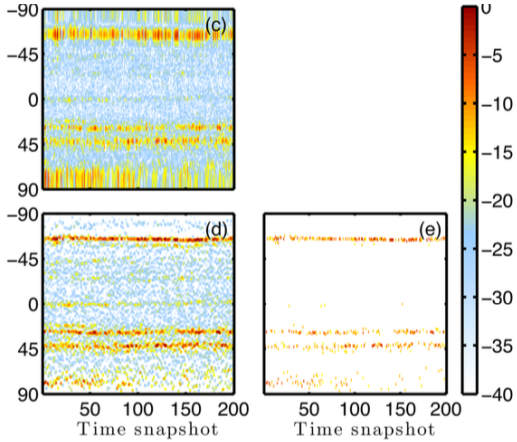

Yao, GRL 2011, PNAS 2013

## Sequential CS



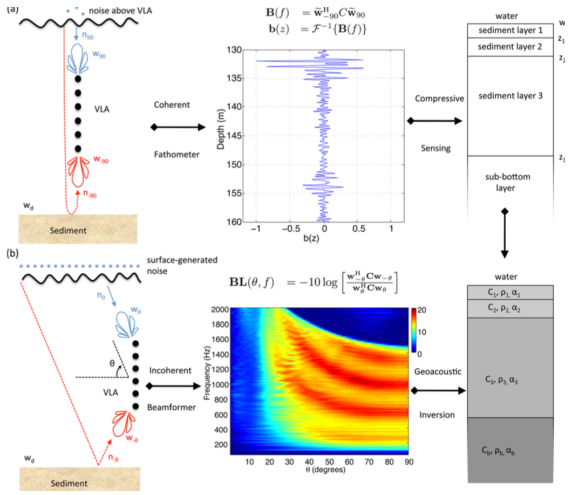

Mecklenbrauker, TSP 2013

## CS beamforming



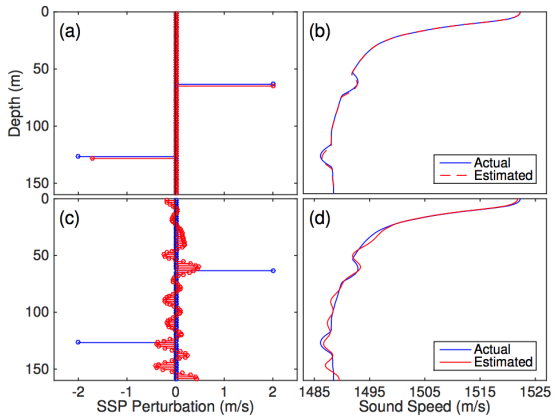

Xenaki, JASA 2014, 2015
Gerstoft JASA 2015

## CS fathometer



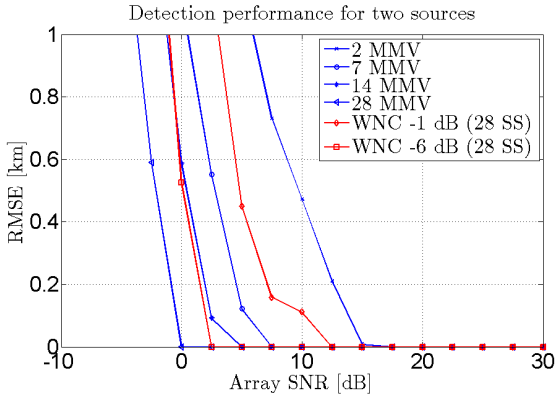

Yardim, JASA 2014

## CS Sound speed estimation




Bianco, JASA 2016

## CS matched field




Gemba, JASA 2016

# Sparse signals /compressive signals are important

- We don't need to sample at the Nyquist rate
- Many signals are sparse, but are solved them under non-sparse assumptions
  - Beamforming
  - Fourier transform
  - Layered structure
- Inverse methods are inherently sparse: We seek the simplest way to describe the data
- All this requires **new developments**
  - Mathematical theory
  - New algorithms (interior point solvers, convex optimization)
  - Signal processing
  - New applications/demonstrations

# Sparse Recovery

- We try to find the sparsest solution which explains our noisy measurements
- $L_0$-norm

- Here, the $L_0$-norm is a shorthand notation for counting the number of non-zero elements in x.

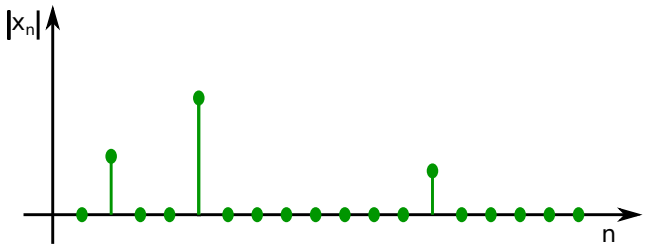
# Sparse Recovery using $L_0$-norm

Underdetermined problem

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \quad M < N$$

Prior information

$\mathbf{x}$: K-sparse, $K \ll N$




$$\|\mathbf{x}\|_0 = \sum_{n=1}^{N} 1_{x_n \neq 0} = K$$

Not really a norm: $\|a\mathbf{x}\|_0 = \|\mathbf{x}\|_0 \neq |a|\|\mathbf{x}\|_0$

There are only few sources with unknown locations and amplitudes

- $L_0$-norm solution involves exhaustive search
- Combinatorial complexity, not computationally feasible

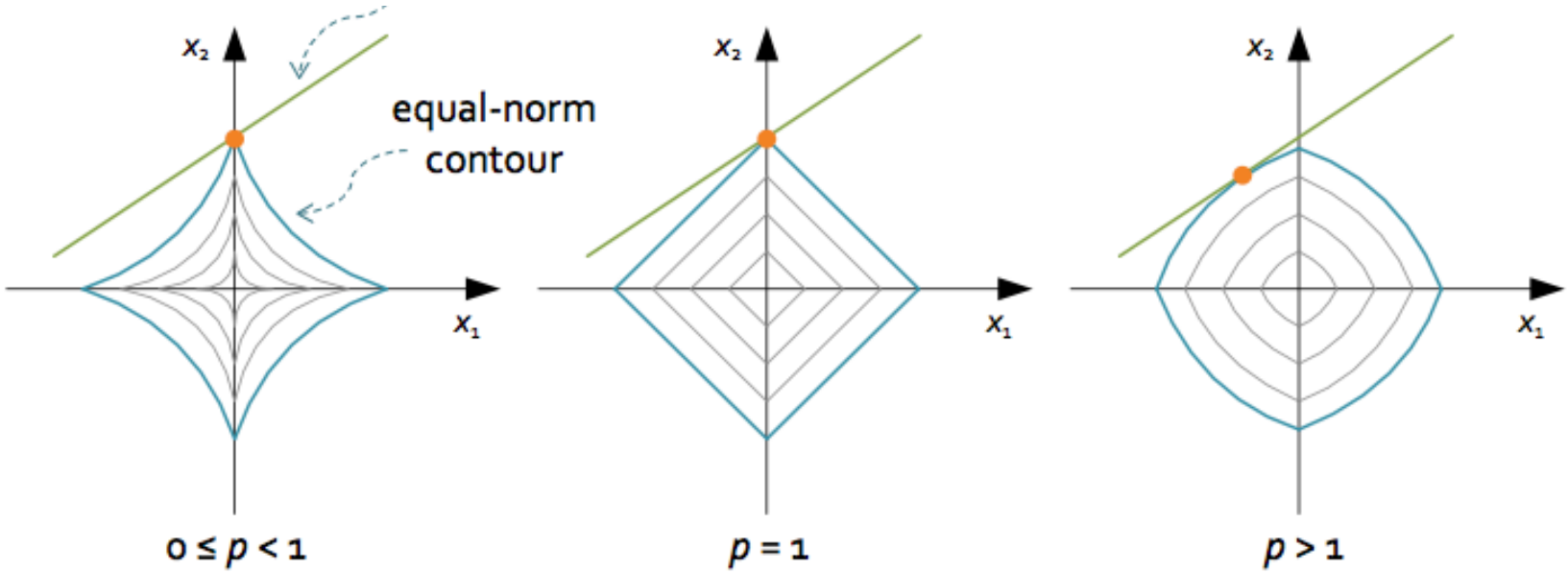
# $L_p$-norm

$$\| x \|_p = \left( \sum_{m=1}^{M} | x_m |^p \right)^{1/p} \quad \text{for} \quad p > 0$$

- Classic choices for $p$ are 1, 2, and ∞.
- We will misuse notation and allow also $p = 0$.

# $L_p$-norm (graphical representation)

$$\|x\|_p = \left( \sum_{m=1}^{M} |x_m|^p \right)^{1/p}$$



# Solutions for sparse recovery

- Exhaustive search
  - $L_0$ regularization, not computationally feasible
- Convex optimization
  - Basis pursuit / LASSO / $L_1$ regularization
- Greedy search
  - Matching pursuit / Orthogonal matching pursuit (OMP)
- Bayesian analysis
  - Sparse Bayesian Learning / SBL
- Regularized least squares
  - $L_2$ regularization, reference solution, not actually sparse

# Regularized least squares

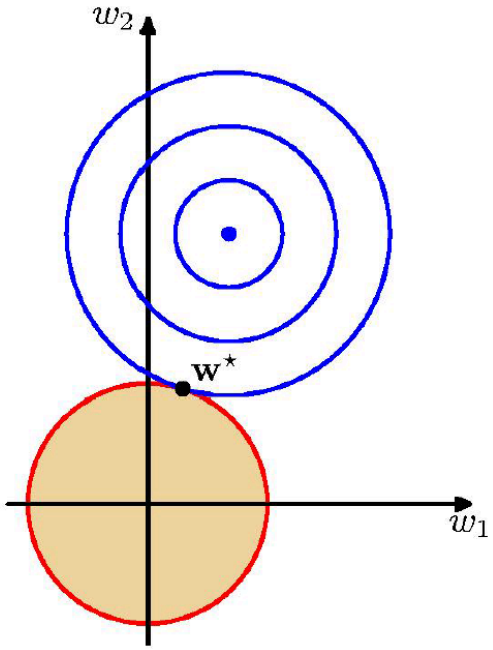
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

The squared weights penalty is mathematically compatible with the squared error function, giving a closed form for the optimal weights:

$$\mathbf{w}^* = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$

- Slide 8/9, Lecture 4
- Regularized least squares solution
- Solution not sparse

A picture of the effect of the regularizer




- The overall cost function is the sum of two parabolic bowls.
- The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The L2 regularizer just **shrinks** the weights.

# Basis Pursuit / LASSO / $L_1$ regularization

- The $L_0$-norm minimization is not convex and requires combinatorial search making it computationally impractical

- We make the problem convex by substituting the $L_1$-norm in place of the $L_0$-norm

$$\min_x \| \mathbf{x} \|_1 \qquad \text{subject to} \quad \| \mathbf{Ax} - \mathbf{b} \|_2 < \varepsilon$$

- This can also be formulated as

# The unconstrained -LASSO- formulation

Constrained formulation of the $\ell_1$-norm minimization problem:

$$\widehat{\mathbf{x}}_{\ell_1}(\epsilon) = \underset{\mathbf{x}\in\mathbb{C}^N}{\arg\min} \|\mathbf{x}\|_1 \text{ subject to } \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon$$

Unconstrained formulation in the form of least squares optimization with an $\ell_1$-norm regularizer:

$$\widehat{\mathbf{x}}_{\text{LASSO}}(\mu) = \underset{\mathbf{x}\in\mathbb{C}^N}{\arg\min} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \mu\|\mathbf{x}\|_1$$

For every $\epsilon$ exists a $\mu$ so that the two formulations are equivalent

Regularization parameter : $\mu$

# Basis Pursuit / LASSO / $L_1$ regularization

- Why is it OK to substitute the $L_1$-norm for the $L_0$-norm?
- What are the conditions such that the two problems have the same solution?

$$\min_x \| x \|_1 \quad \text{subject to} \quad \| Ax - b \|_2 < \varepsilon$$

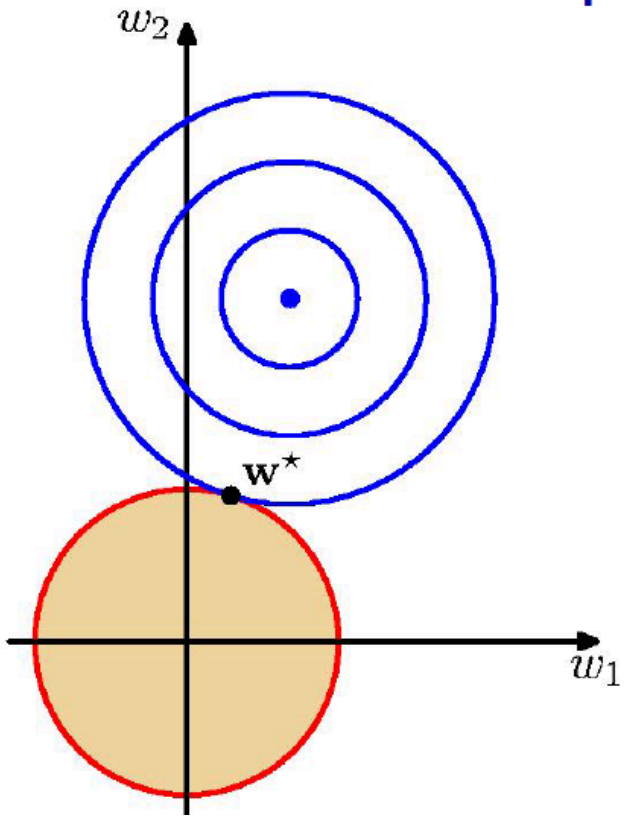
$$\min_x \| x \|_0 \quad \text{subject to} \quad \| Ax - b \|_2 < \varepsilon$$

- Restricted Isometry Property (RIP)

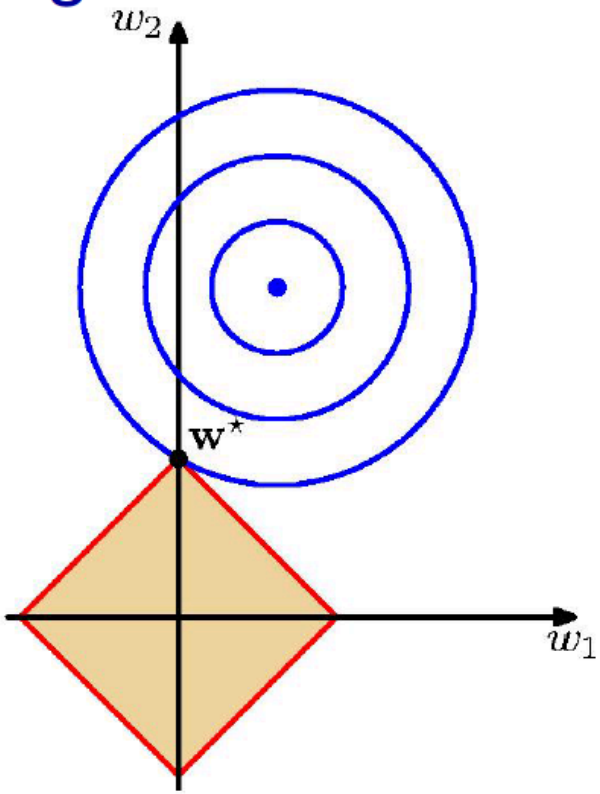
$$(1 - \delta_s)\|\boldsymbol{u}\|_2 \leq \|\boldsymbol{A_S u}\|_2 \leq (1 + \delta_s)\|\boldsymbol{u}\|_2$$

# Geometrical view (Figure from Bishop)

## Geometrical view of the lasso compared with a penalty on the squared weights




$L_2$ regularization



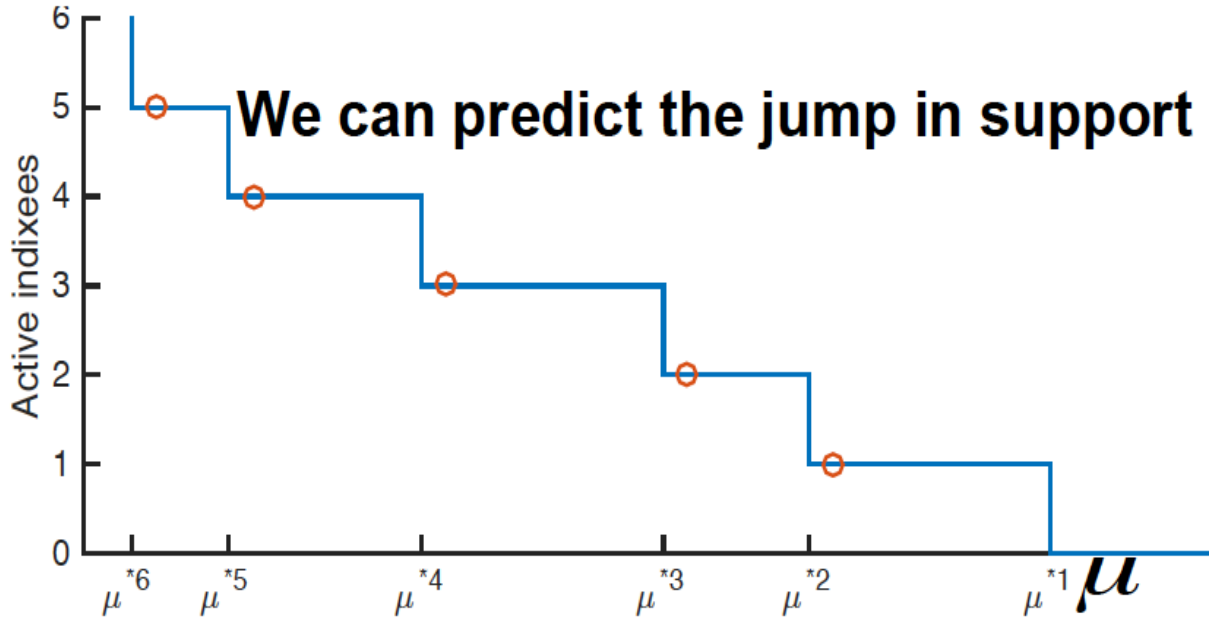

$L_1$ regularization

# Regularization parameter selection

The objective function of the LASSO problem:

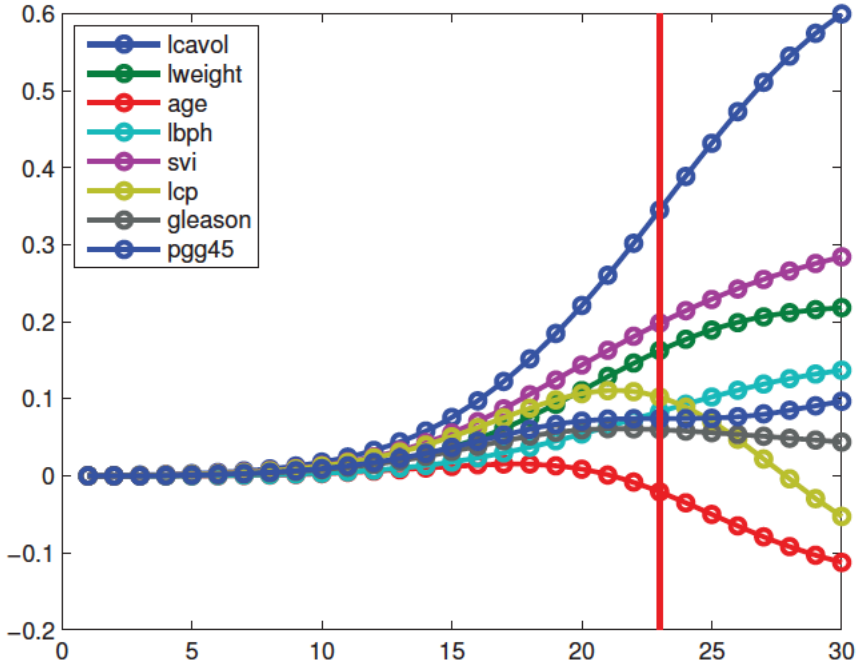
$$L(\mathbf{x}, \mu) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \mu\|\mathbf{x}\|_1$$

- Regularization parameter : $\mu$
- Sparsity depends on $\mu$
- $\mu$ large, x = 0
- $\mu$ small, non-sparse



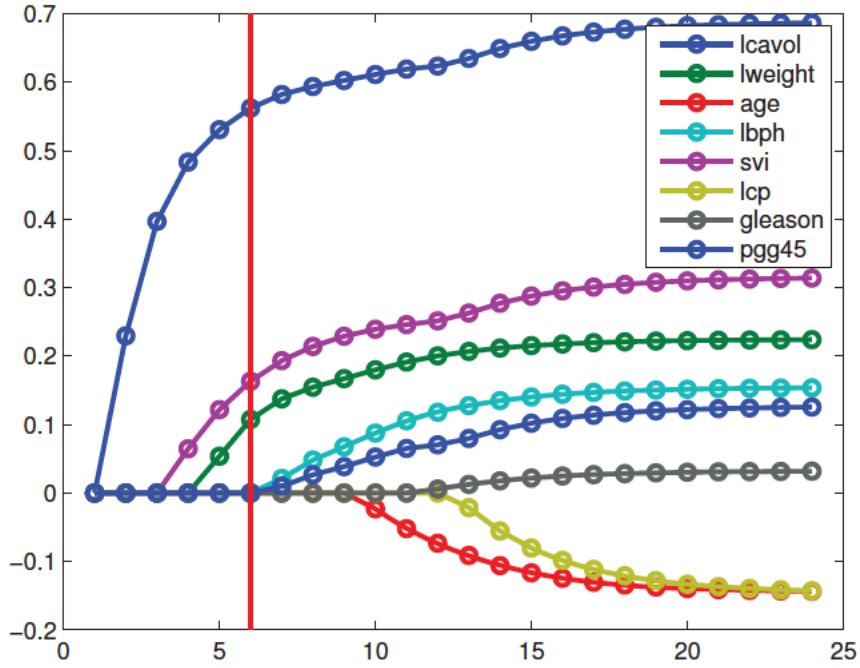
# Regularization Path (Figure from Murphy)

$L_2$ regularization




(a) $1/\mu$

$L_1$ regularization

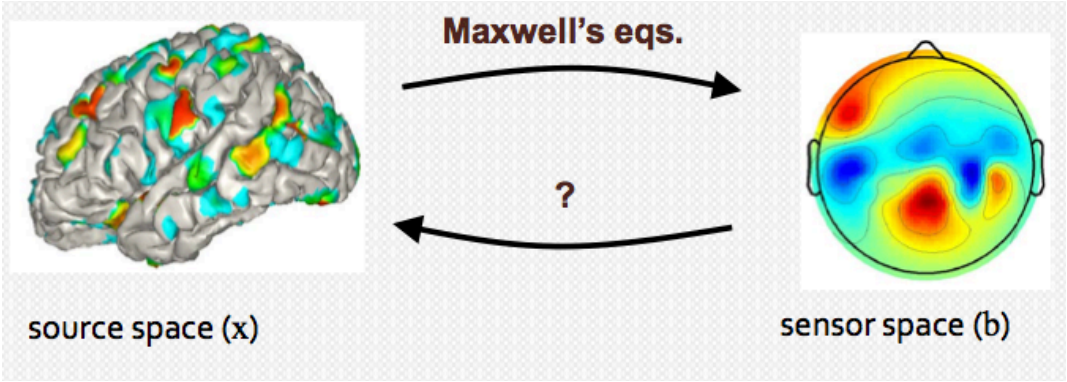



(b) $1/\mu$

- As regularization parameter $\mu$ is decreased, more and more weights become active
- Thus $\mu$ controls sparsity of solutions

# Applications

- MEG/EEG/MRI source location (earthquake location)
- Channel equalization
- Compressive sampling (beyond Nyquist sampling)
- Compressive camera!




- Beamforming
- Fathometer
- Geoacoustic inversion
- Sequential estimation