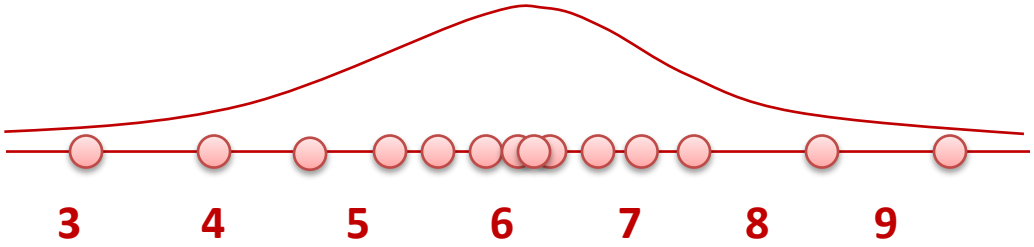
Non-parametric density estimation

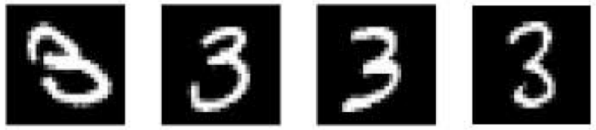
Aarti Singh

Machine Learning 10-315
Nov 18, 2019

What is density/distribution estimation?



Distribution of sleep time



Distribution of intensities at a pixel

Distribution of Heads/Tails for a coin

Distribution of a random variable P(X = x)

Discrete random variable – probability mass function

Continuous random variable – probability density function

Goal: Given $X_1, X_2, \ldots, X_n$, estimate P(X)

Parametric methods (distribution estimation)

- Assume some model

  Discrete random variable - Bernoulli, Multinomial, Poisson

  Continuous random variable – Gaussian, Exponential, Laplace

  Mixtures!

- Estimate parameters ($\mu, \sigma^2, \theta, w, \beta$) using MLE/MAP and plug in

- **Pro** – need few data points to learn parameters
- **Con** – Strong distributional assumptions, not satisfied in practice

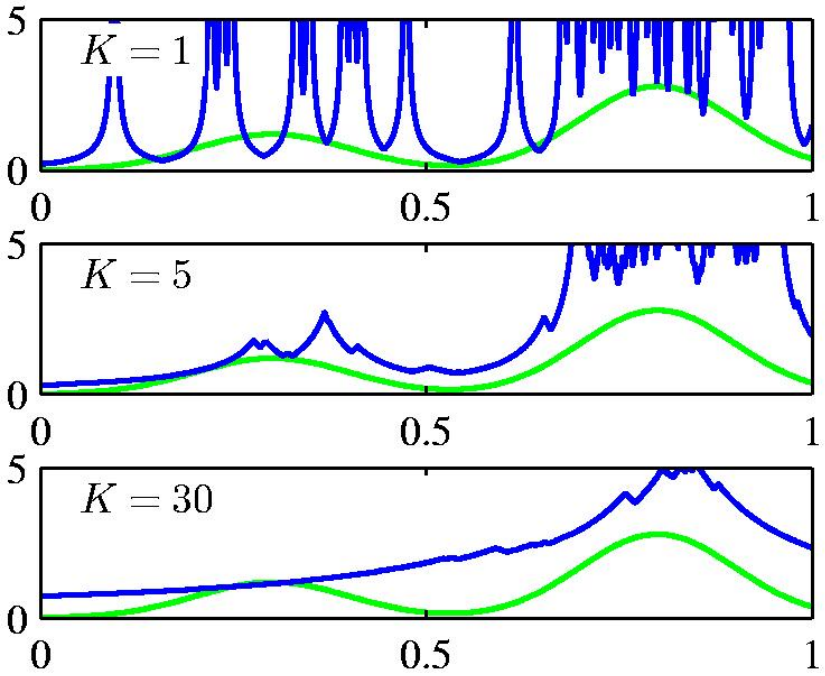
Non-Parametric methods

- Typically don't make any distributional assumptions
- As we have more data, we should be able to learn more complex models
- Let number of parameters scale with number of training data

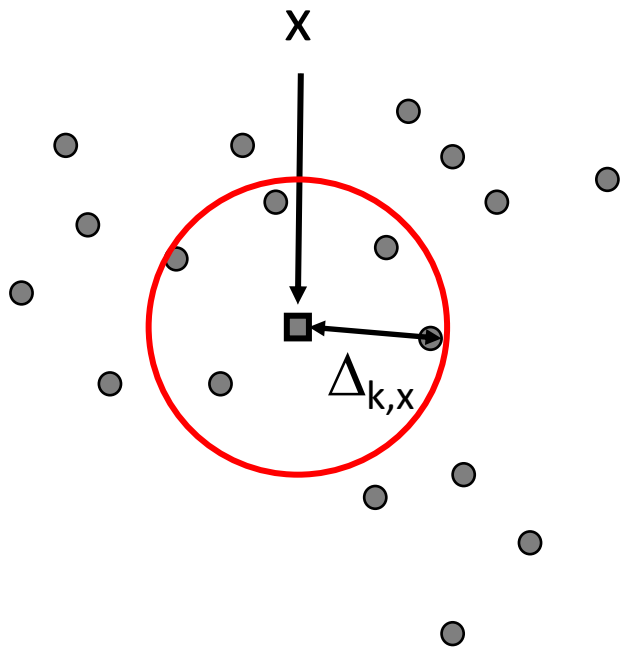
- Some nonparametric methods for classification and regression
  - Decision Trees
  - k-NN (k-Nearest Neighbor) Classifier
  - Kernel regression
- Some nonparametric methods for density estimation – k-NN, Histograms, Kernel density estimation

k-NN density estimation

$$\widehat{p}(x) = \frac{k}{n\Delta_{k,x}}$$



k acts as a smoother.



Not very popular for density estimation – spiked estimates

Histogram density estimate

Partition the feature space into distinct bins with widths Δ_i and count the number of observations, n_i, in each bin.

$$\widehat{p}(x) = \frac{n_i}{n\Delta_i} \mathbf{1}_{x \in \mathrm{Bin}_i}$$

"Local relative frequency"

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- Δ acts as a smoothing parameter.

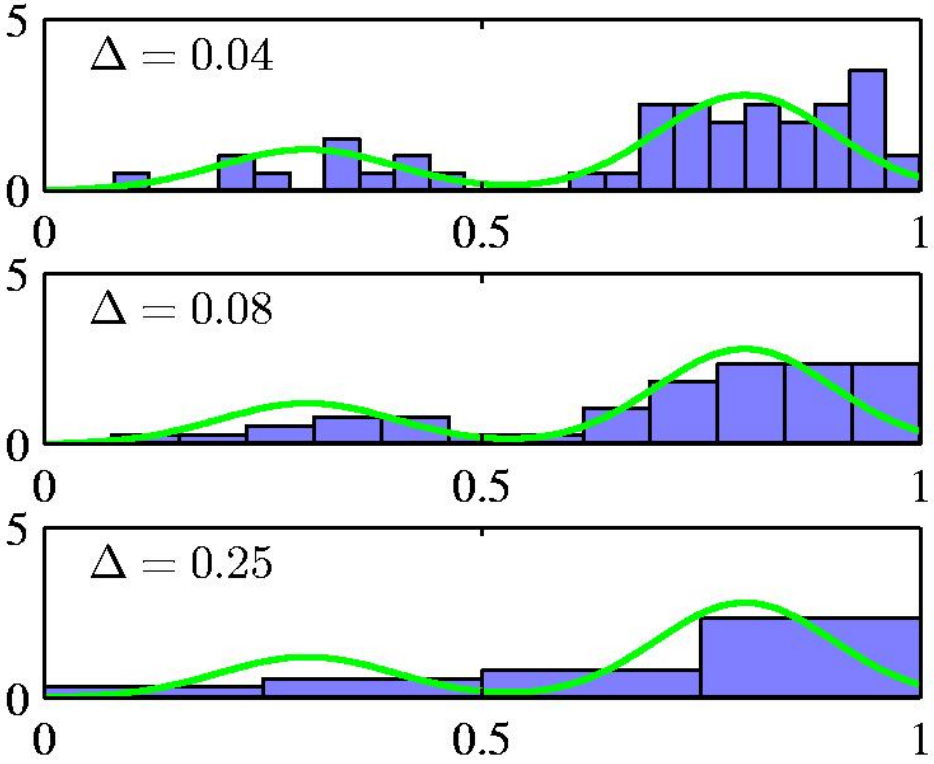


Image src: Bishop book

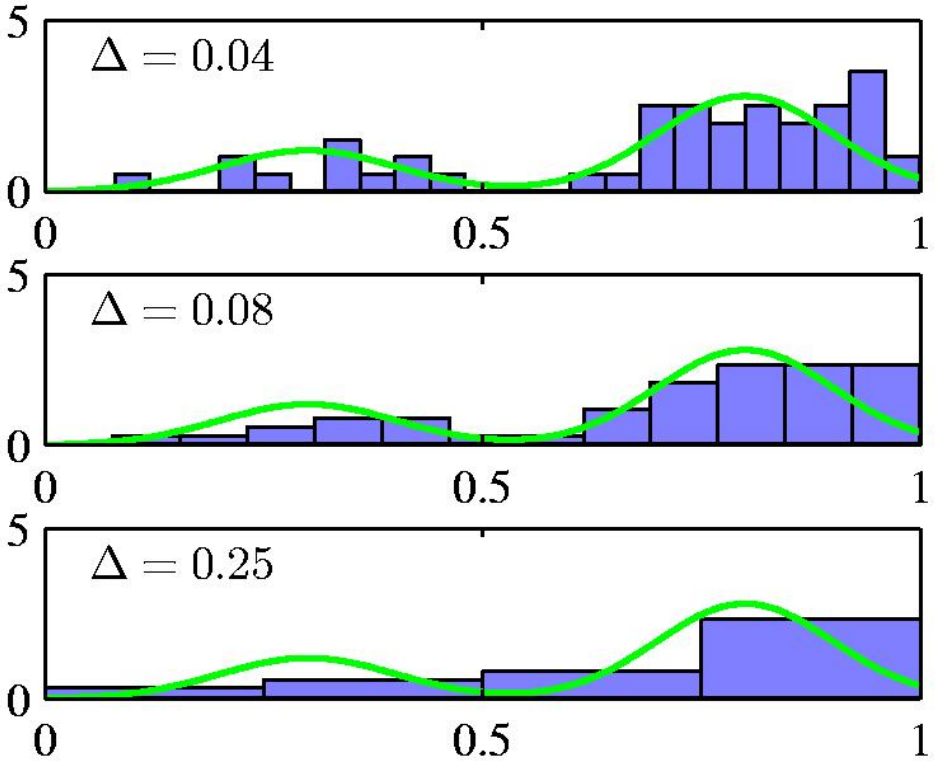
Effect of histogram bin width

$$\widehat{p}(x) = \frac{n_i}{n\Delta} \mathbf{1}_{x \in \mathrm{Bin}_i}$$

Small Δ, large #bins
Good fit but unstable
(few points per bin)
"**Small bias**, **Large variance**"

Large Δ, small #bins
Poor fit but stable
(many points per bin)
"**Large bias**, **Small variance**"

bins = $1/\Delta$



Histogram as MLE

- Underlying model – density is constant on each bin

  Parameters p_j : density in bin j

  Note $\sum_j p_j = 1/\Delta$ since $\int p(x)dx = 1$

- Maximize likelihood of data under probability model with parameters p_j

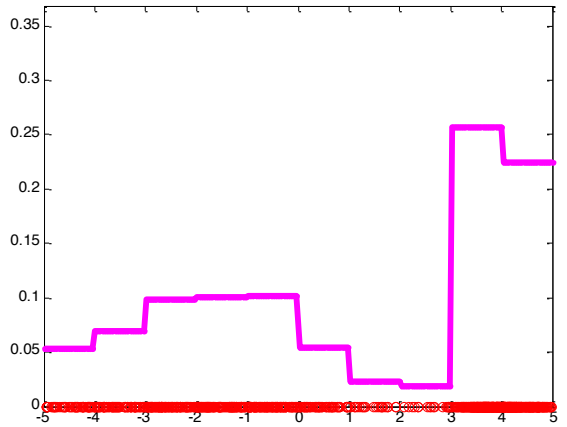
$$\hat{p}(x) = \arg\max_{\{p_j\}} P(X_1, \ldots, X_n; \{p_j\}_{j=1}^{1/\Delta}) \quad \text{s.t.} \quad \sum_j p_j = 1/\Delta$$

- Show that histogram density estimate is MLE under this model

Kernel density estimate

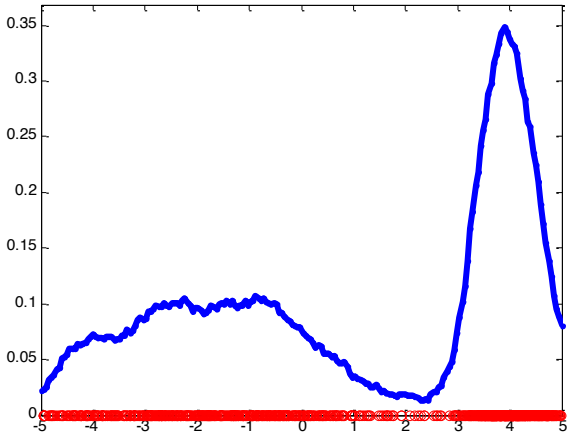
- Histogram – blocky estimate

$$\widehat{p}(x) = \frac{1}{\Delta} \frac{\sum_{j=1}^{n} \mathbf{1}_{X_j \in \mathrm{Bin}_x}}{n}$$



- Kernel density estimate aka “Parzen/moving window method”

$$\widehat{p}(x) = \frac{1}{\Delta} \frac{\sum_{j=1}^{n} \mathbf{1}_{||X_j - x|| \leq \Delta}}{n}$$



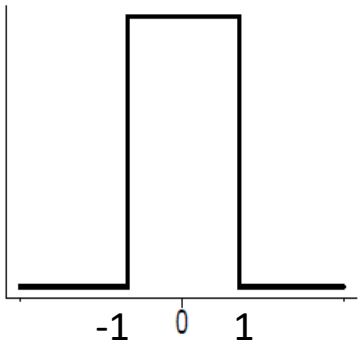
Kernel density estimate

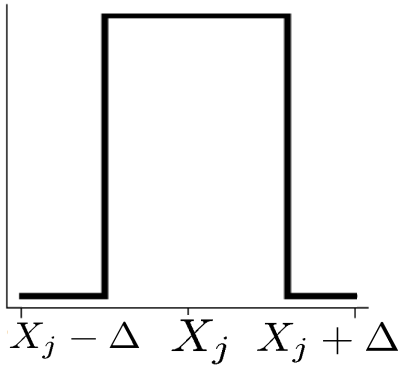
- $\widehat{p}(x) = \frac{1}{\Delta} \frac{\sum_{j=1}^{n} K\left(\frac{X_j - x}{\Delta}\right)}{n}$ more generally

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

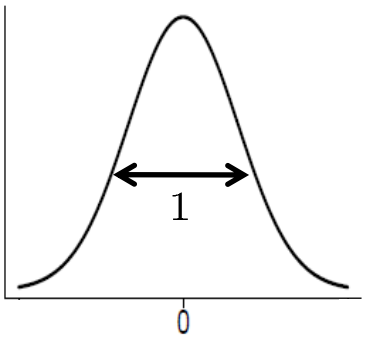
$$K(x) = \frac{1}{2} I(x),$$

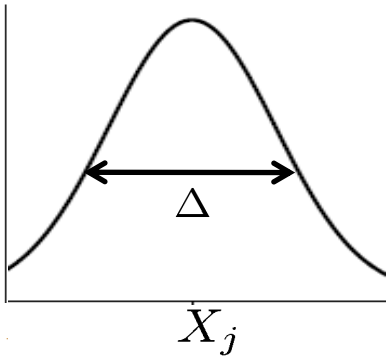




Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

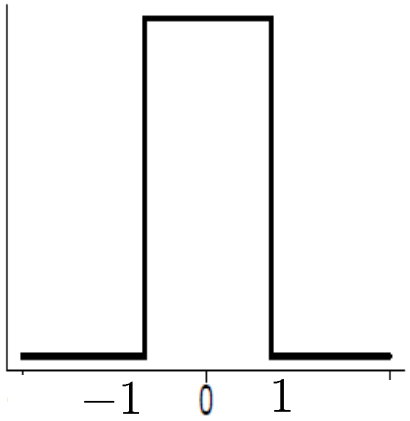




Kernels

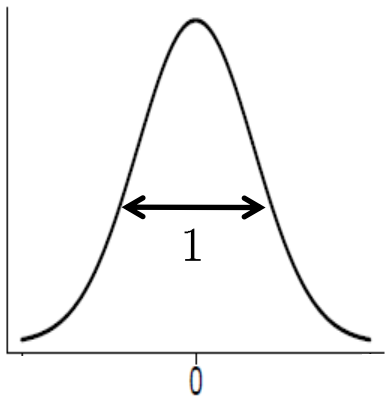
boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$



Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



Any kernel function that satisfies

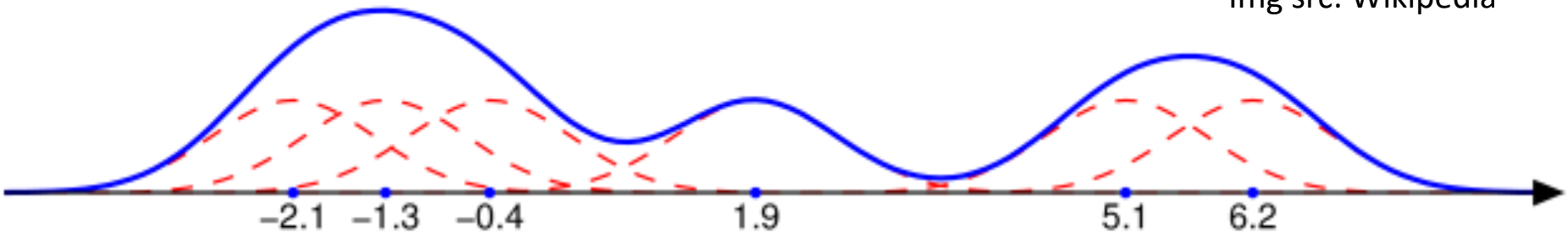
$$K(x) \geq 0,$$

$$\int K(x)dx = 1$$

Kernel density estimation

- Place small "bumps" at each data point, determined by the kernel function.
- The estimator consists of a (normalized) "sum of bumps".

Img src: Wikipedia



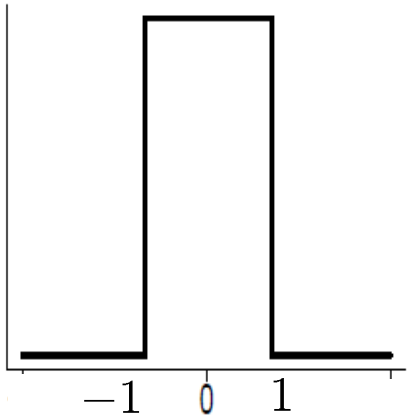
Gaussian bumps (red) around six data points and their sum (blue)

- Note that where the points are denser the density estimate will have higher values.

Choice of Kernels

boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$

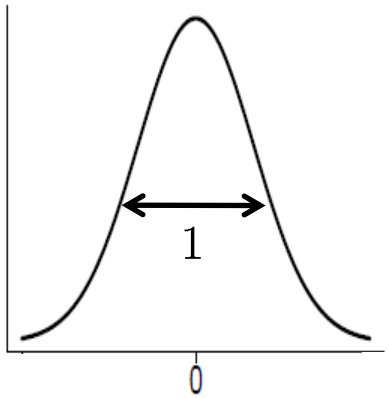


Finite support

– only need local points to compute estimate

Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

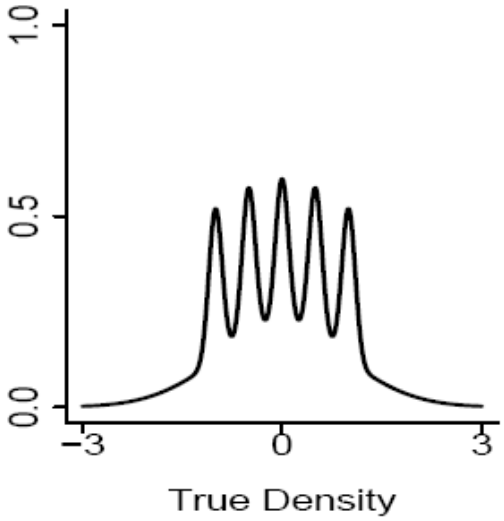


Infinite support

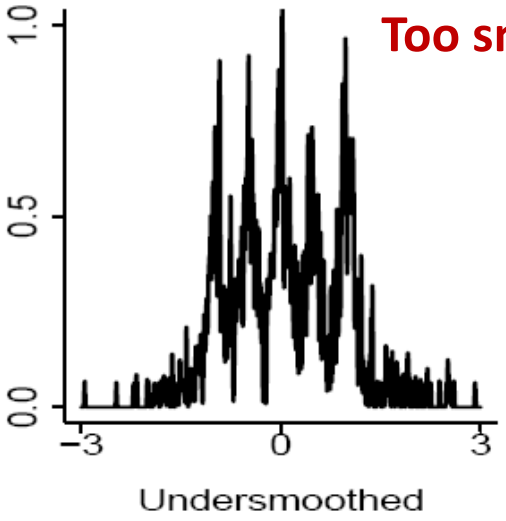
- need all points to compute estimate

-But quite popular since smoother

Choice of kernel bandwidth



True Density



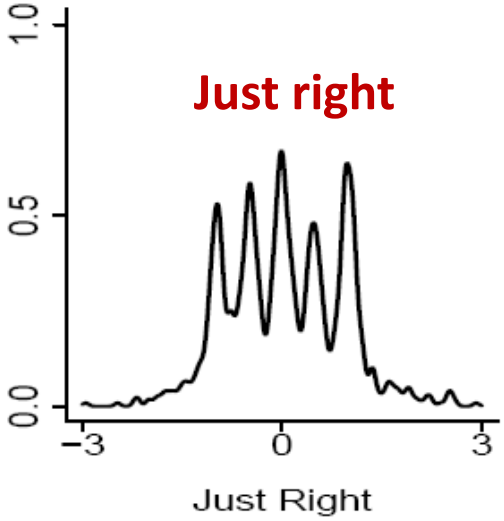
Undersmoothed

Too small

Image Source:
Larry's book – All of Nonparametric Statistics

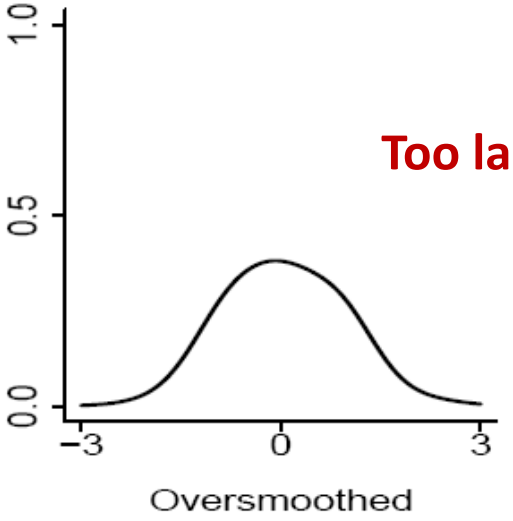
Bart-Simpson Density

Just right



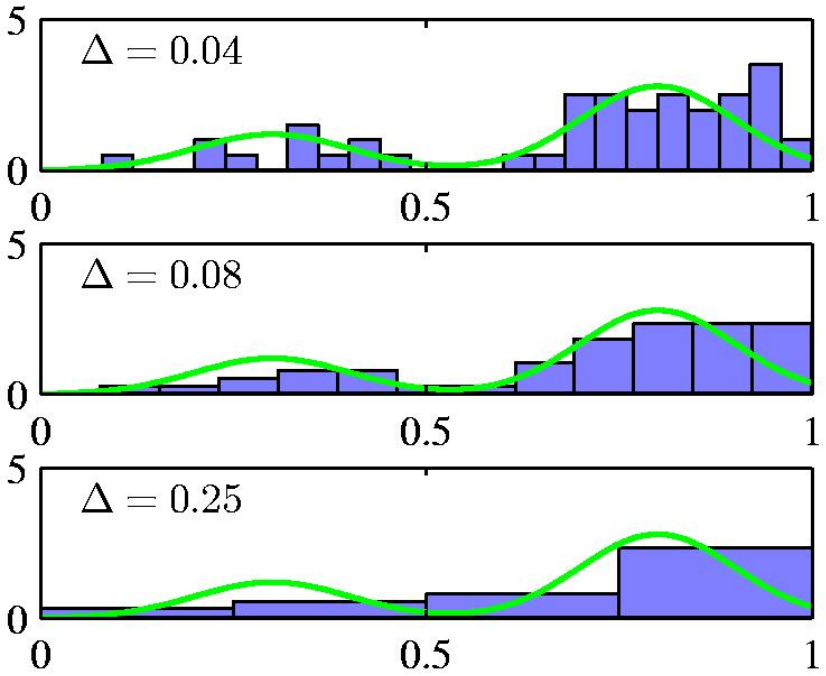
Just Right

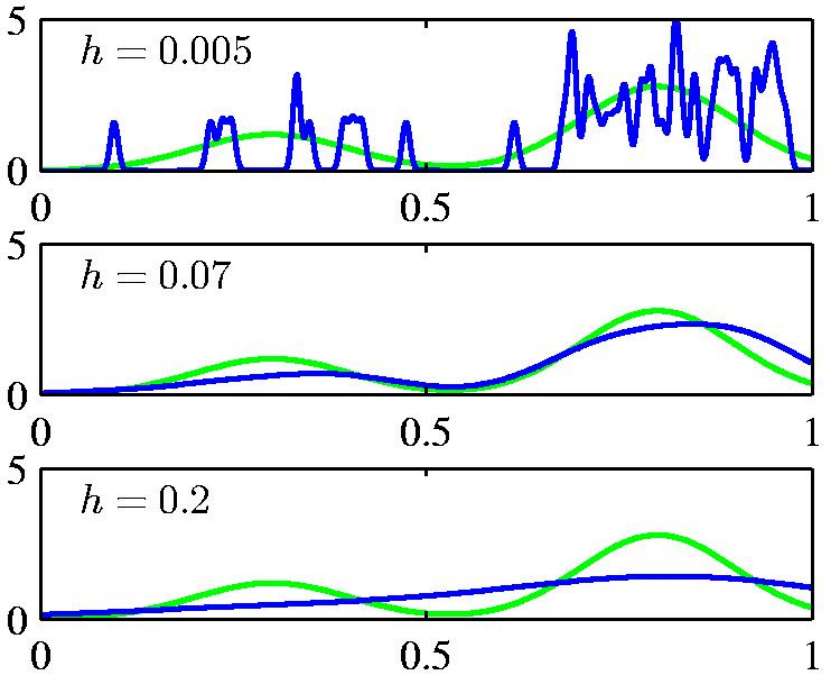
Too large



Oversmoothed

Histograms vs. Kernel density estimation





Δ = h acts as a smoother.

Nonparametric density estimation

- Histogram $\widehat{p}(x) = \frac{n_i}{n\Delta} \mathbf{1}_{x \in \mathrm{Bin}_i}$
- Kernel density est $\widehat{p}(x) = \frac{n_x}{n\Delta}$

Fix Δ, estimate number of points within Δ of x (n_i or n_x) from data

Fix $n_x = k$, estimate Δ from data (volume of ball around x that contains k training pts)

- k-NN density est $\widehat{p}(x) = \frac{k}{n\Delta_{k,x}}$

Summary

- Non-parametric approaches

Four things make a nonparametric/memory/instance based/lazy learner:

1. *A distance metric, dist(x,X_i)*
   Euclidean (and many more)
2. *How many nearby neighbors/radius to look at?*
   k, Δ/h
3. *A weighting function (optional)*
   W based on kernel K
4. *How to fit with the local points?*
   Average, Majority vote, Weighted average, Poly fit

Summary

- Parametric vs Nonparametric approaches
  - Nonparametric models place very mild assumptions on the data distribution and provide good models for complex data

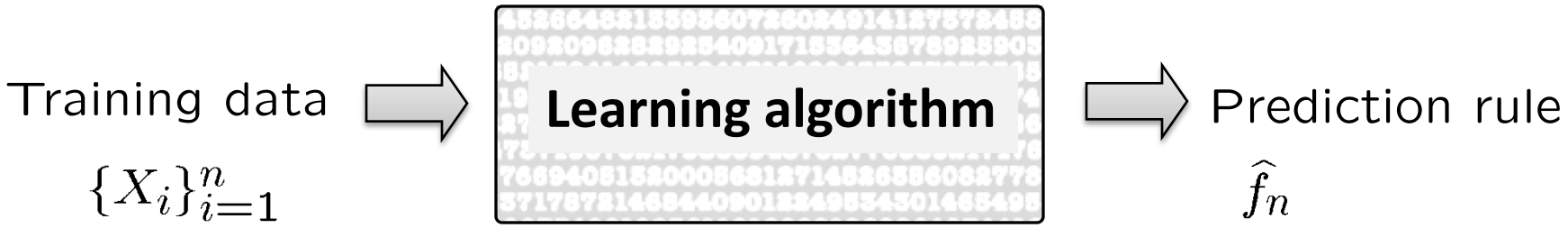
    Parametric models rely on very strong (simplistic) distributional assumptions

  - Nonparametric models (not histograms) requires storing and computing with the entire data set.

    Parametric models, once fitted, are much more efficient in terms of storage and computation.

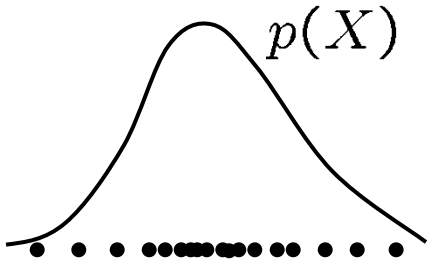
Unsupervised Learning

"Learning from unlabeled/unannotated data" (without supervision)



What can we predict from unlabeled data?

- Density estimation



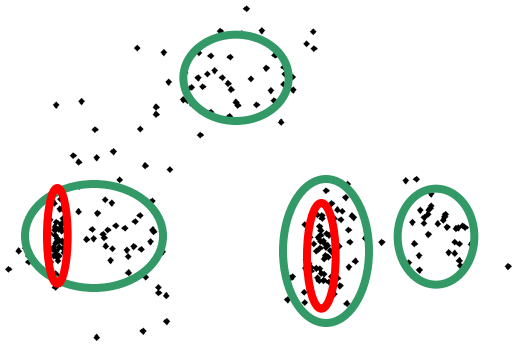
Unsupervised Learning

"Learning from unlabeled/unannotated data" (without supervision)

Training data $\{X_i\}_{i=1}^n$ → **Learning algorithm** → Prediction rule $\hat{f}_n$

What can we predict from unlabeled data?

- Density estimation
- Groups or clusters in the data



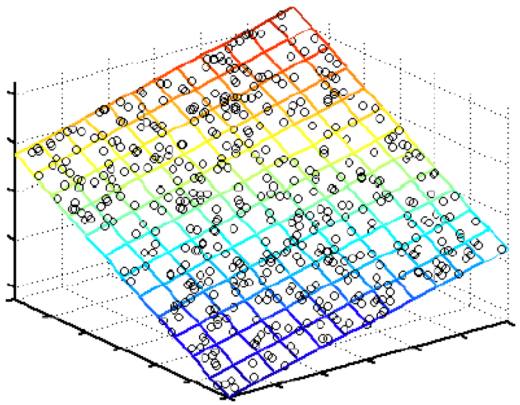
Unsupervised Learning

"Learning from unlabeled/unannotated data" (without supervision)

Training data $\{X_i\}_{i=1}^n$ → **Learning algorithm** → Prediction rule $\widehat{f}_n$

What can we predict from unlabeled data?

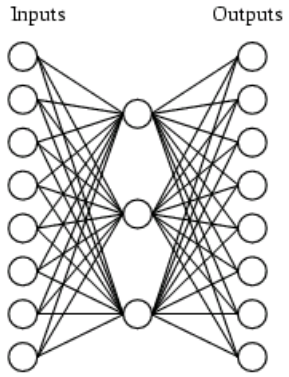
- Density estimation
- Groups or clusters in the data
- Dimensionality reduction



Dimensionality reduction using Neural nets

Auto-Encoders

Learning Hidden Layer Representations

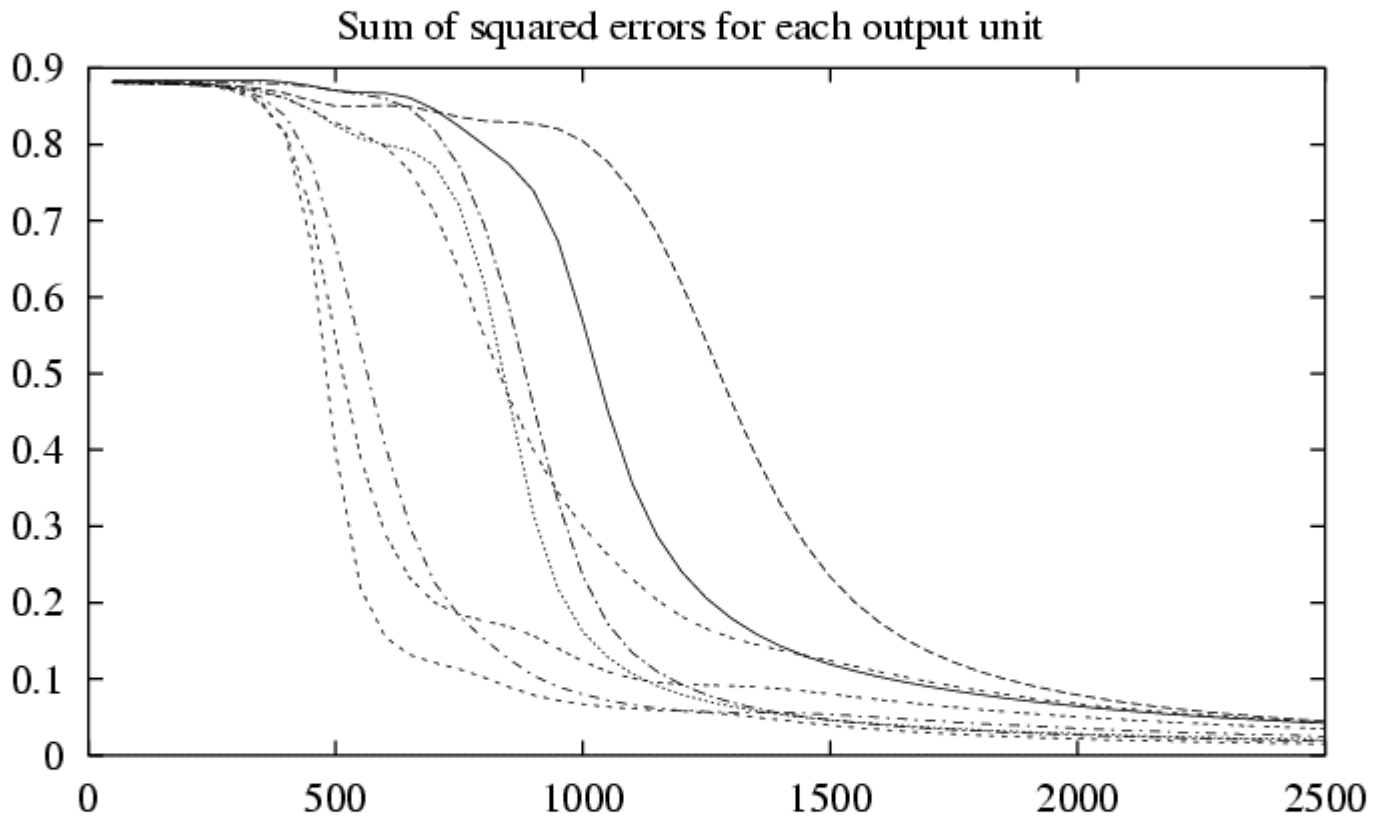


A target function:

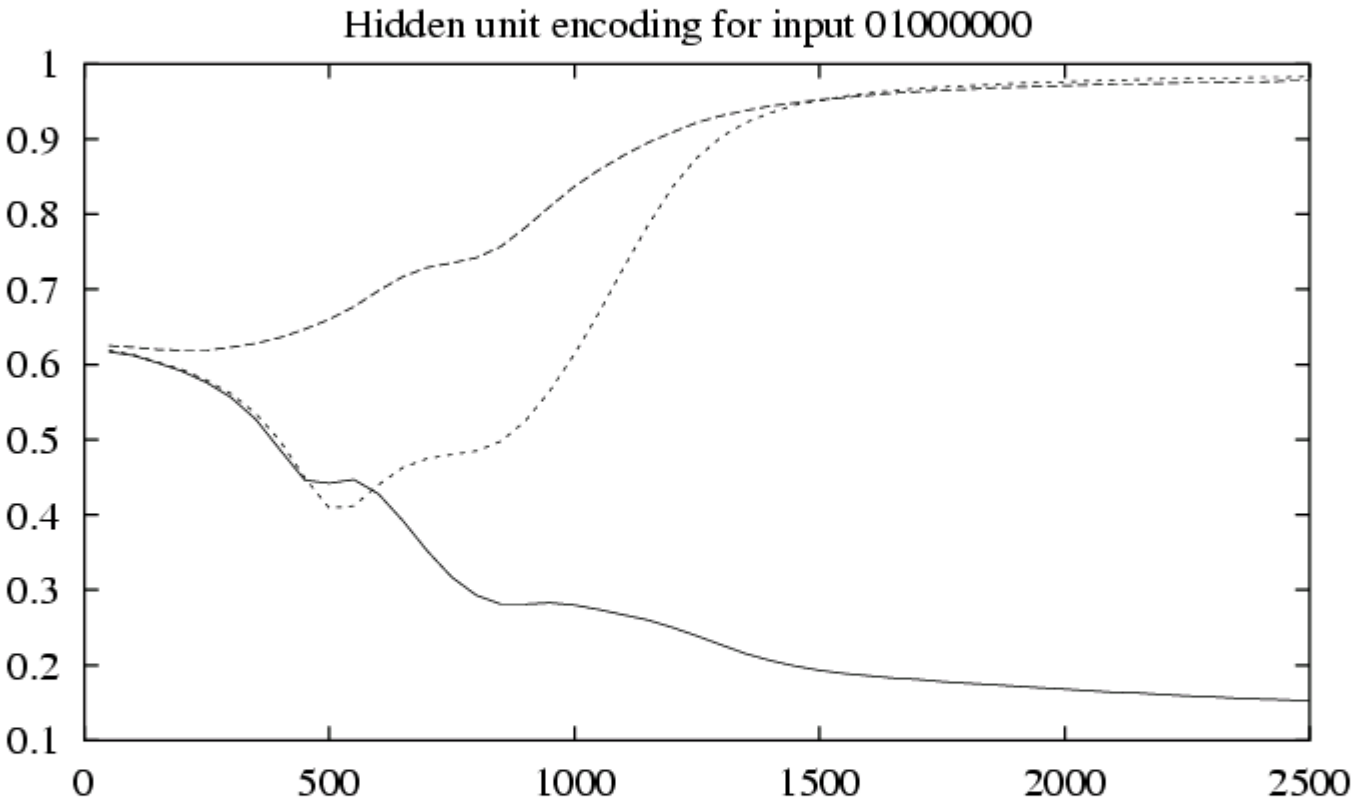
Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Can this be learned??

Training

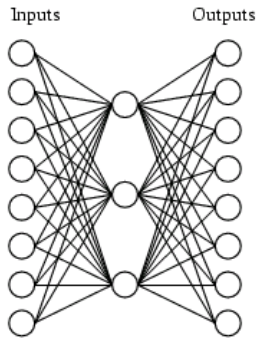

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

Training


Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500

Learning Hidden Layer Representations

A network:



Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Dimensionality reduction using Neural nets

Auto-Encoders

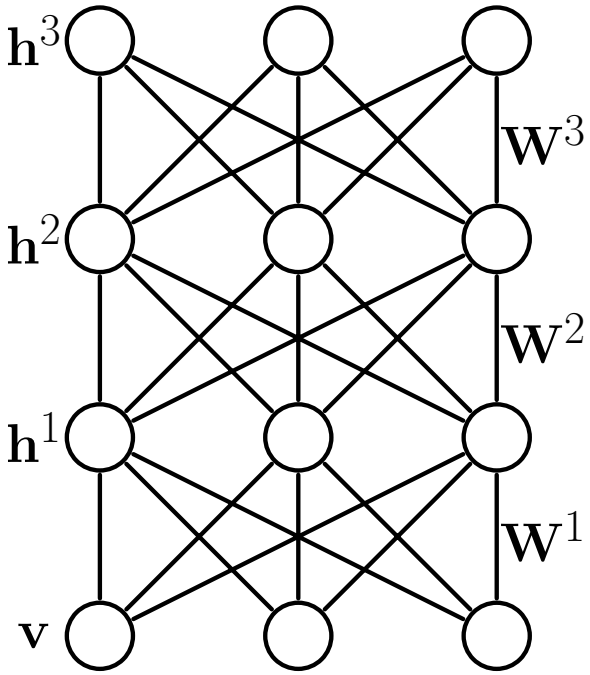
Density estimation using Neural nets

Deep Belief Networks
Generative Adversarial Networks (GANs)

Deep Boltzmann Machine (DBM)

Deep Generative Model

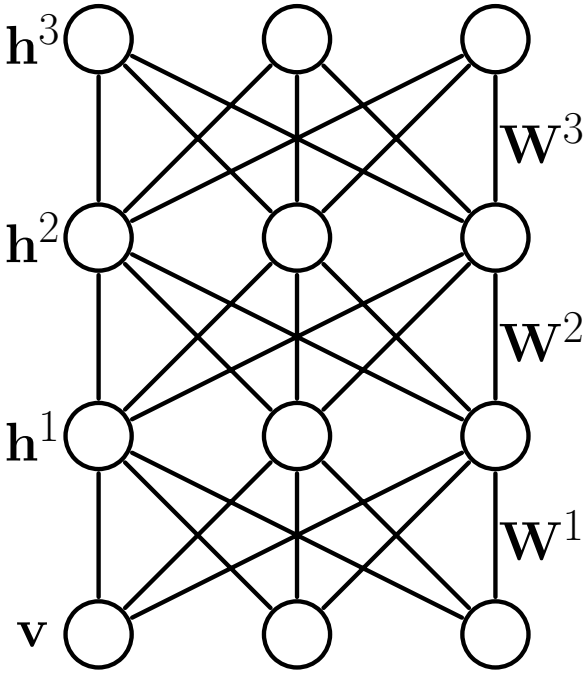
(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)



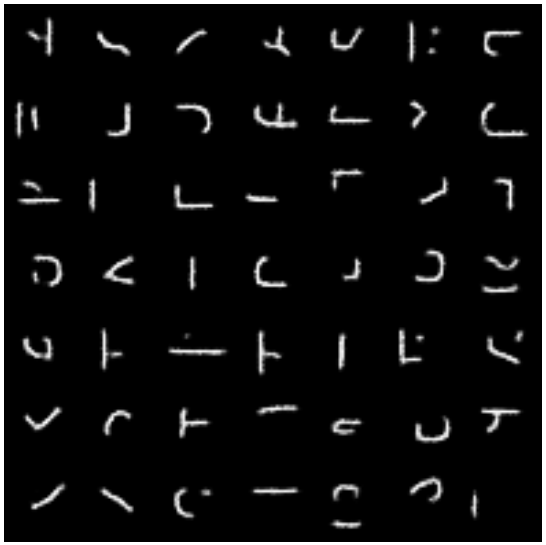
$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + {\mathbf{h}^{(1)}}^\top W^{(2)} \mathbf{h}^{(2)} + {\mathbf{h}^{(2)}}^\top W^{(3)} \mathbf{h}^{(3)}\right]$$

Bernoulli Markov Random Field

Hand-written Character Recognition

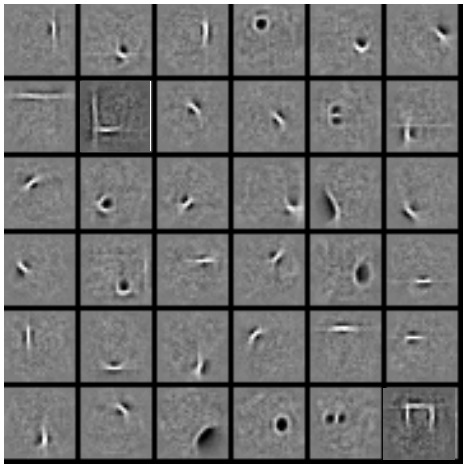


Learned 2nd-layer features



Strokes

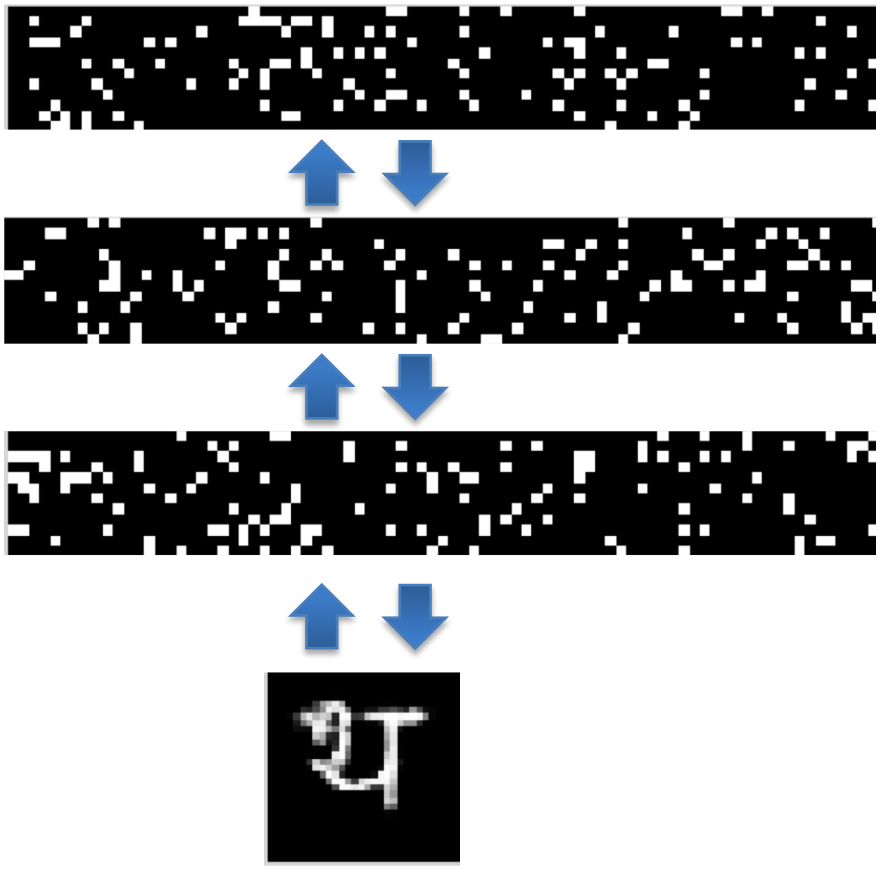
Learned 1st layer features



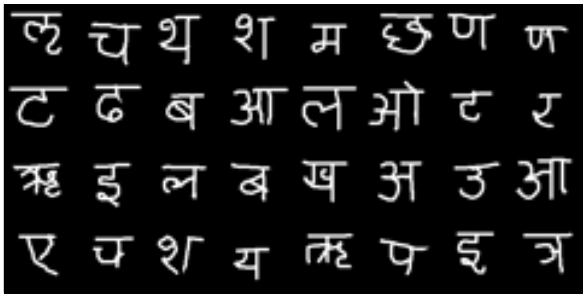
Edges

Deep Boltzmann Machines for Text Characters

Sanskrit



Model P(image)



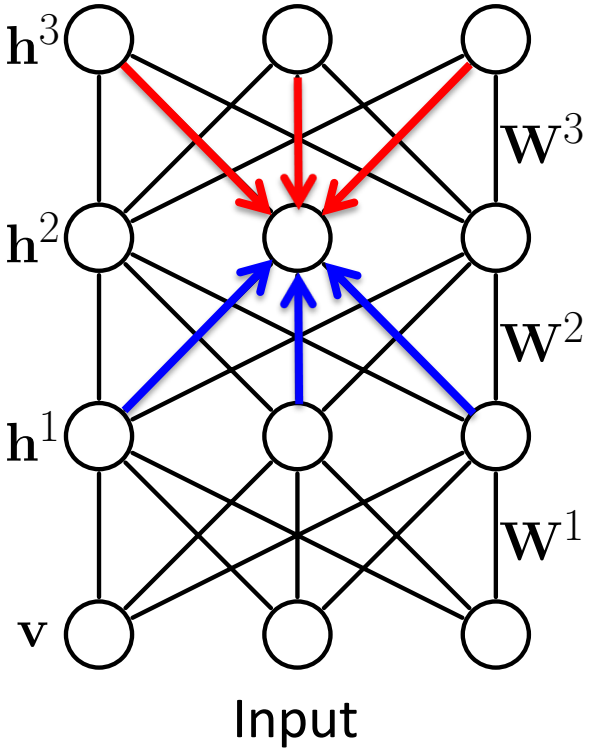
Subset of 25,000 characters from 50 alphabets around the world.

- 3,000 hidden variables
- 784 observed variables (28 by 28 images)
- About 2 million parameters

Bernoulli Markov Random Field

DBMs Model Formulation

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} + \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}\right]$$



$\theta = \{W^1, W^2, W^3\}$ model parameters

- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

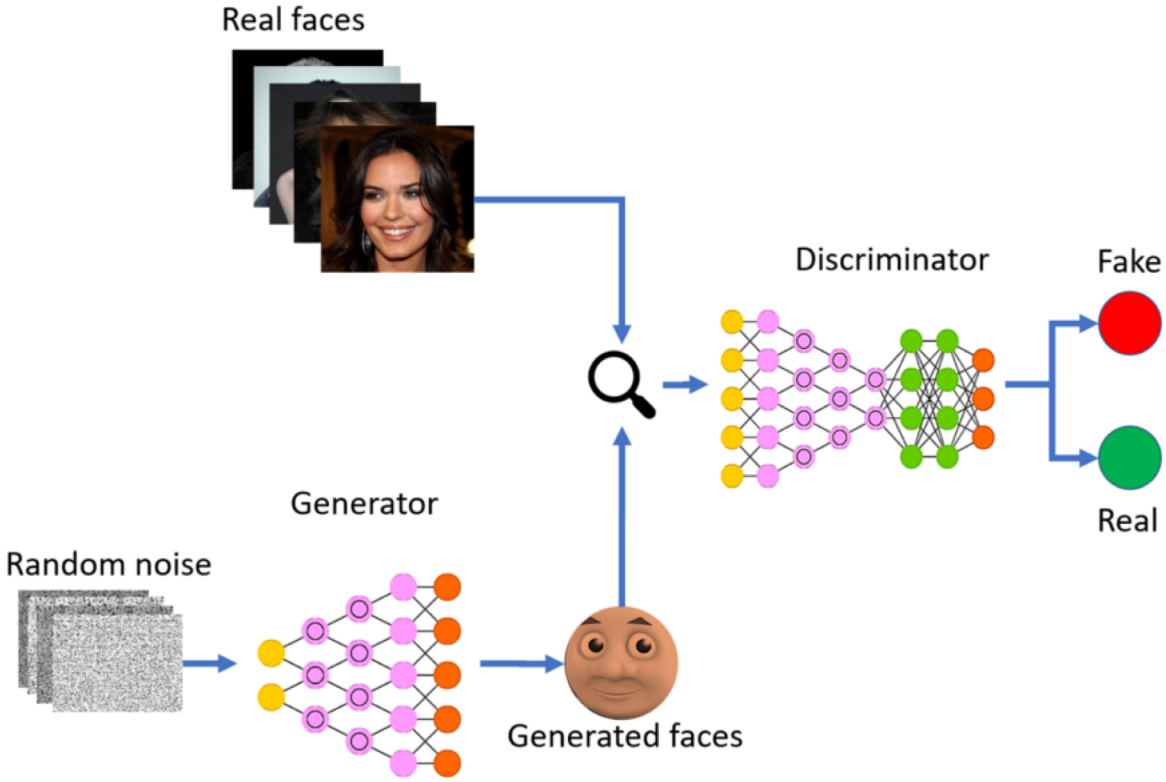
$$P(h_j^2 = 1 | \mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W_{kj}^3 h_k^3 + \sum_m W_{mj}^2 h_m^1\right)$$

Top-down Bottom-up

- Hidden variables are dependent even when **conditioned on the input**. – computationally quite challenging!

Generative Adversarial Networks (GANs)

Goal: Generate examples that look like real pool



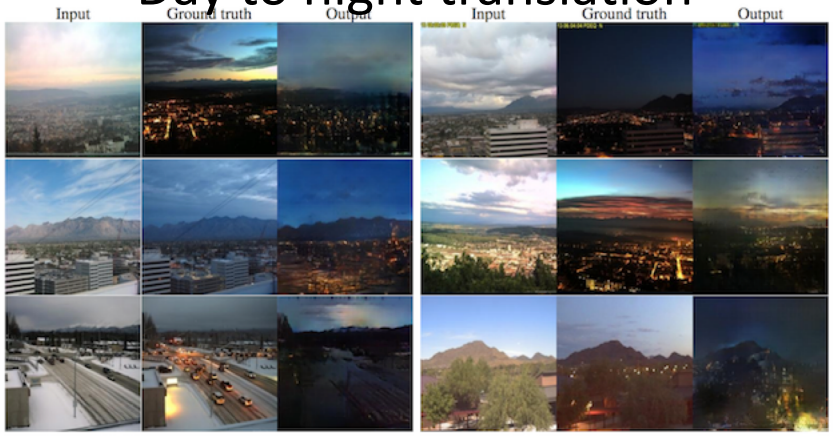
Generator – attempts to generate fake images that look like real images (minimize accuracy)

Discriminator – attempts to discriminate between fake and real images (maximize accuracy)

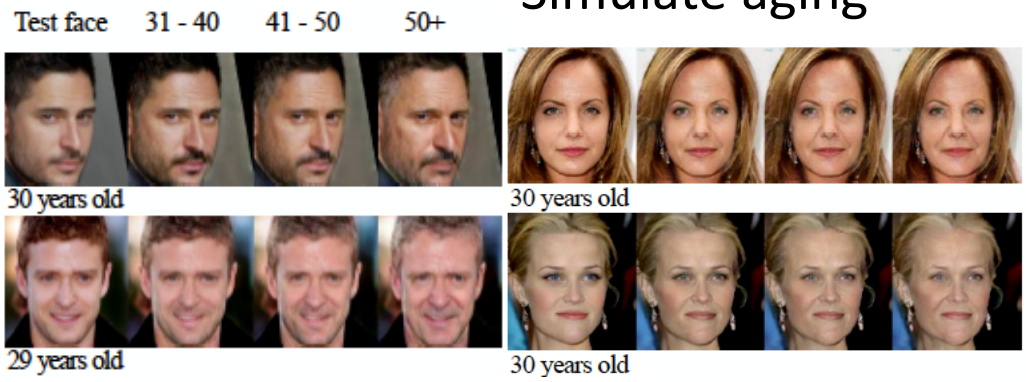
Generative Adversarial Networks (GANs)

Fake celebrities

Day to night translation



Simulate aging



Cartoon generation

Figure 7: Generated samples