Lecture 20 — Hidden markov models; intro to GANs.

Alex Schwing and Matus Telgarsky

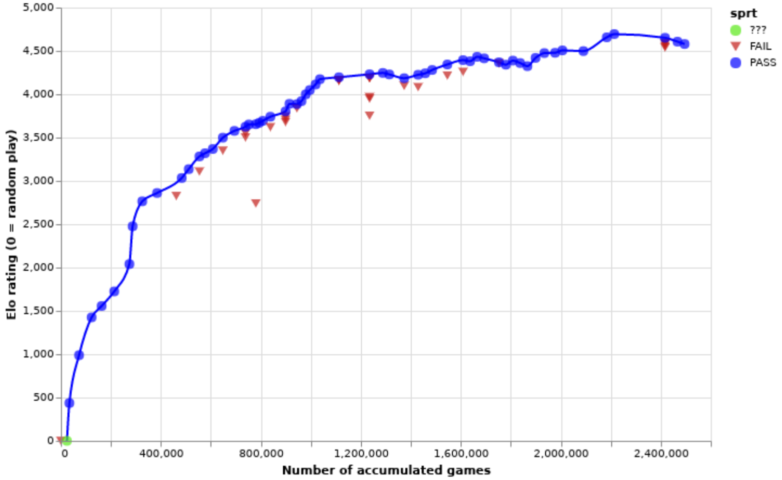
April 5, 2018

Announcements.

- Midterms available in **TA** office hours.
  - 8 more days for regrade requests.
- Homework 12 pushed back 1 week.
- Ongoing questions:
  - Midterm solutions available ...?
  - Videos ...?

A fun project.

lczero — open source, distributed AlphaGo Zero for chess.



Active Users

223 users in the last day have played 214791 games

Where is E-M still used?

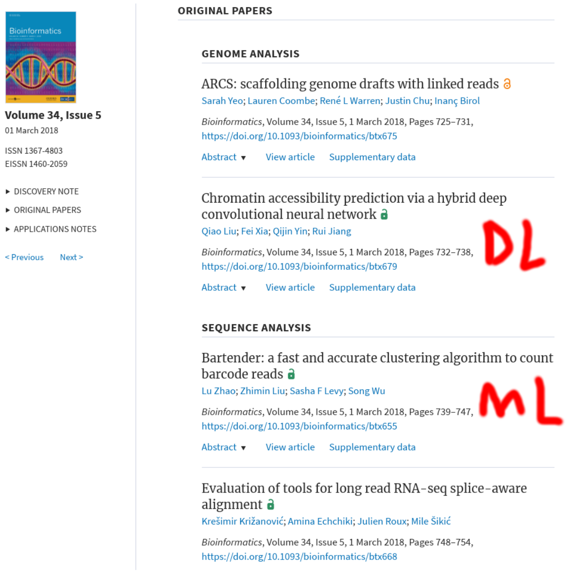
This was asked last time.

- The shortest answer: many applied fields still use E-M, Gaussian mixtures, HMMs (e.g., speech, bioinformatics), however slowly it seems everything is transitioning to deep learning!

Therefore the natural question still becomes: **why learn this stuff?**

- To be able to discuss current ML, compare ML methods.
- To be able to reason about ML methods in different ways.
- Always useful to learn more tools.
- Cynical view: worst case, analogous to learning basic algorithms for coding interviews. . .

Regarding bioinformatics.



Volume 34, Issue 5
01 March 2018

ISSN 1367-4803
EISSN 1460-2059

▸ DISCOVERY NOTE
▸ ORIGINAL PAPERS
▸ APPLICATIONS NOTES

< Previous Next >

ORIGINAL PAPERS

GENOME ANALYSIS

ARCS: scaffolding genome drafts with linked reads
Sarah Yeo; Lauren Coombe; René L Warren; Justin Chu; Inanç Birol

Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 725–731,
https://doi.org/10.1093/bioinformatics/btx675

Abstract ▾ View article Supplementary data

Chromatin accessibility prediction via a hybrid deep convolutional neural network
Qiao Liu; Fei Xia; Qijin Yin; Rui Jiang

DL

Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 732–738,
https://doi.org/10.1093/bioinformatics/btx679

Abstract ▾ View article Supplementary data

SEQUENCE ANALYSIS

Bartender: a fast and accurate clustering algorithm to count barcode reads
Lu Zhao; Zhimin Liu; Sasha F Levy; Song Wu

ML

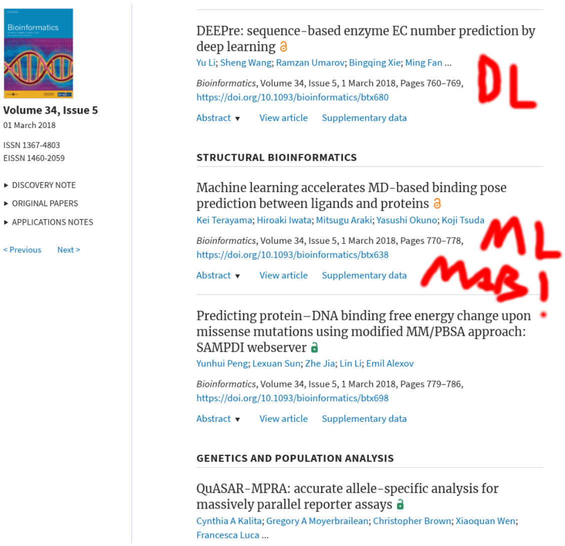
Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 739–747,
https://doi.org/10.1093/bioinformatics/btx655

Abstract ▾ View article Supplementary data

Evaluation of tools for long read RNA-seq splice-aware alignment
Krešimir Križanović; Amina Echchiki; Julien Roux; Mile Šikić

Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 748–754,
https://doi.org/10.1093/bioinformatics/btx668

Regarding bioinformatics.



Volume 34, Issue 5
01 March 2018

ISSN 1367-4803
EISSN 1460-2059

▸ DISCOVERY NOTE
▸ ORIGINAL PAPERS
▸ APPLICATIONS NOTES

< Previous Next >

DEEPre: sequence-based enzyme EC number prediction by deep learning
Yu Li; Sheng Wang; Ramzan Umarov; Bingqing Xie; Ming Fan ...
Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 760–769, https://doi.org/10.1093/bioinformatics/btx680
Abstract ▾ View article Supplementary data

DL

STRUCTURAL BIOINFORMATICS

Machine learning accelerates MD-based binding pose prediction between ligands and proteins
Kei Terayama; Hiroaki Iwata; Mitsugu Araki; Yasushi Okuno; Koji Tsuda
Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 770–778, https://doi.org/10.1093/bioinformatics/btx638
Abstract ▾ View article Supplementary data

ML
MaB!

Predicting protein–DNA binding free energy change upon missense mutations using modified MM/PBSA approach: SAMPDI webserver
Yunhui Peng; Lexuan Sun; Zhe Jia; Lin Li; Emil Alexov
Bioinformatics, Volume 34, Issue 5, 1 March 2018, Pages 779–786, https://doi.org/10.1093/bioinformatics/btx698
Abstract ▾ View article Supplementary data

GENETICS AND POPULATION ANALYSIS

QuASAR-MPRA: accurate allele-specific analysis for massively parallel reporter assays
Cynthia A Kalita; Gregory A Moyerbrailean; Christopher Brown; Xiaoquan Wen; Francesca Luca ...

(etc.)

Schedule for today.

Schedule for today.

- Graphical models; Hidden Markov Models intro.
- More E-M: learning Hidden Markov Models.
- Kernel density estimates (“Parzen windows”).
- GAN intro: distributions and neural networks.
- GAN intro: sampling with neural networks.

Graphical models; Hidden Markov Models intro.

Graphical models; Hidden Markov Models intro.

Recall the sampling story for GMMs:

$$\begin{aligned} Y &\sim \text{Discrete}(\pi_1, \ldots, \pi_k) && \text{choose cluster;} \\ X|Y=j &\sim \mathcal{N}(\mu_j, \Sigma_k) && \text{choose point.} \end{aligned}$$

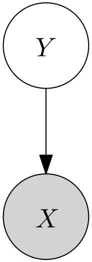
Y is **latent/hidden/unobserved**, X is **observed**.

Graphical models; Hidden Markov Models intro.

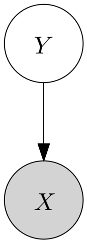
Recall the sampling story for GMMs:

$$\begin{aligned} Y &\sim \text{Discrete}(\pi_1, \ldots, \pi_k) && \text{choose cluster;} \\ X|Y=j &\sim \mathcal{N}(\mu_j, \Sigma_k) && \text{choose point.} \end{aligned}$$

Y is **latent/hidden/unobserved**, X is **observed**.
Graphical Model:



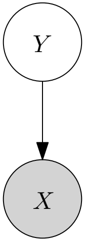
Graphical model for GMMs.



Basic rules (there are more complicated variants)**:**

- Nodes denote random variables.
- Edges denote conditional dependence.
- Shaded nodes are observed; unshaded are unobserved.

Graphical model for GMMs.



Basic rules (there are more complicated variants)**:**

- Nodes denote random variables.
- Edges denote conditional dependence.
- Shaded nodes are observed; unshaded are unobserved.

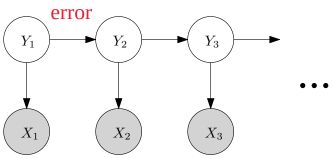
Gaussian likelihood according to the graphical model:

$$\begin{aligned} p(X_1, \ldots, X_n) &= \sum_{j_1 \in [k], \ldots, j_n \in [k]} p(X_1, \ldots, X_n, Y_1 = j_1, \ldots, Y_n = j_n) \\ &= \sum_{j_1 \in [k], \ldots, j_n \in [k]} \prod_{i=1}^{n} p(Y_i = j_i) p(X_i | Y_i = j_i). \end{aligned}$$

Hidden Markov Models: basics.

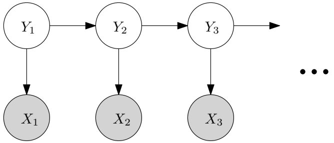
- As with GMM:
  - **Observed** random variables $(X_1, \ldots, X_n)$.
  - **Latent variables** $(Y_1, \ldots, Y_n)$.
  - Conditional independence of observations given latent variables: e.g., $p(X_i | X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_t, Y_1, \ldots Y_t) = p(X_i | Y_i)$.
- **Unlike GMMs:** $(Y_1, \ldots, Y_t)$ have dependencies!
  - **Markov assumption:** Y_{i+1} depends *only* on Y_i.
- Graphical model:

conditional probability



Hidden Markov Models: likelihood.

- Graphical model.



- Likelihood.

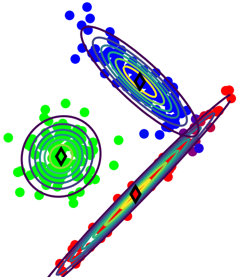
$$
\begin{aligned}
&p(X_1, \dots, X_n, Y_1, \dots, Y_n) \\
&= p(Y_1)p(X_1, \dots, X_n, Y_2, \dots, Y_n | Y_1) \\
&= p(Y_1)p(X_1|Y_1)p(X_2, \dots, X_n, Y_2, \dots, Y_n | Y_1) \\
&= p(Y_1)p(X_1|Y_1)p(X_2|Y_2)p(Y_2|Y_1)p(X_3, \dots, X_n, Y_2, \dots, Y_n | Y_2, Y_1 \\
&= p(Y_1)p(X_1|Y_1)p(X_2|Y_2)p(Y_2|Y_1)p(X_3, \dots, X_n, Y_2, \dots, Y_n | Y_2) \\
&= p(Y_1) \left(\prod_{i=1}^{n} p(X_i|Y_i) \right) \left(\prod_{i=2}^{n} p(Y_i|Y_{i-1}) \right).
\end{aligned}
$$

Hidden Markov Models: parameters.

- Still have parameters for $p(X_i | Y_i = j)$.
  E.g., if this is Gaussian, have parameters (μ_j, Σ_j).
- Still have parameters $(\pi_1, \ldots, \pi_k)$ for Y_1.
- For $(Y_2, \ldots, Y_k)$ have **transition probabilities** $p(Y_{i+1} = j' | Y_i = j)$.
  These are assumed **homogeneous/time-invariant:** e.g., $p(Y_{i+1} = j' | Y_i = j) = p(Y_{i+2} = j' | Y_{i+1} = j)$.
  Write these as a matrix $A \in [0,1]^{k \times k}$:
  $p(Y_{i+1} = l | Y_i = k) = A_{jl}$.
- Depiction: like GMM, but have time series over hidden states!

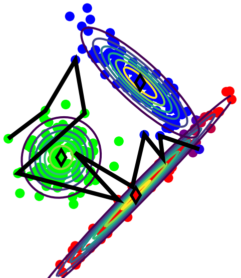
Hidden Markov Models: parameters.

- Still have parameters for $p(X_i | Y_i = j)$.
  E.g., if this is Gaussian, have parameters (μ_j, Σ_j).
- Still have parameters $(\pi_1, \dots, \pi_k)$ for Y_1.
- For $(Y_2, \dots, Y_k)$ have **transition probabilities** $p(Y_{i+1} = j' | Y_i = j)$.
  These are assumed **homogeneous/time-invariant:** e.g., $p(Y_{i+1} = j' | Y_i = j) = p(Y_{i+2} = j' | Y_{i+1} = j)$.
  Write these as a matrix $A \in [0,1]^{k \times k}$:
  $p(Y_{i+1} = l | Y_i = k) = A_{jl}$.
- Depiction: like GMM, but have time series over hidden states!



Hidden Markov Models: parameters.

- Still have parameters for $p(X_i|Y_i = j)$.
  E.g., if this is Gaussian, have parameters (μ_j, Σ_j).
- Still have parameters $(\pi_1, \ldots, \pi_k)$ for Y_1.
- For $(Y_2, \ldots, Y_k)$ have **transition probabilities** $p(Y_{i+1} = j'|Y_i = j)$.
  These are assumed **homogeneous/time-invariant:** e.g., $p(Y_{i+1} = j'|Y_i = j) = p(Y_{i+2} = j'|Y_{i+1} = j)$.
  Write these as a matrix $A \in [0,1]^{k \times k}$:
  $p(Y_{i+1} = l|Y_i = k) = A_{jl}$.
- Depiction: like GMM, but have time series over hidden states!



Hidden Markov Models: applications.

- Speech modeling; e.g., phonemes (/A/, /a/, /b/, . . .).
  - Multivariate Gaussian is over amplitude or frequency window.
  - Transition matrix A allows self-loops!
    Very useful: imagine saying a word slowly.
- Sequence alignment in biology.
- See Murphy Chapter 17 for more applications.
  (Many are being replaced with DNNs, RNNs, . . . !)

Other applications of graphical models.

- Popular in sciences, for interpretability, and easy inclusion of domain knowledge?
- Example: phylogenetic trees (“Mr. Bayes”).

More E-M: learning Hidden Markov Models.

More E-M: learning Hidden Markov Models.

Another interpretation of E-M
(useful for HMMs, and appears on homework?):
E-M maximizes the **expected complete log-likelihood**

$$\mathbb{E}_\theta\left[\ln p_\theta(X_1,\ldots,X_n,\overset{\text{random variables}}{\underline{Y_1,\ldots,Y_n}})|x_1,\ldots,x_n\right],$$

where "$\mathbb{E}_\theta$" means the distribution uses the learned parameters θ, and "$|x_1,\ldots,x_n$" means we condition on the observed data.

More E-M: learning Hidden Markov Models.

Another interpretation of E-M
(useful for HMMs, and appears on homework?):
E-M maximizes the **expected complete log-likelihood**

$$\mathbb{E}_\theta \left[\ln p_\theta(X_1, \ldots, X_n, Y_1, \ldots, Y_n) | x_1, \ldots, x_n \right],$$

where "$\mathbb{E}_\theta$" means the distribution uses the learned parameters θ, and "$|x_1, \ldots, x_n$" means we condition on the observed data.

For GMMs, this becomes

$$= \mathbb{E}_\theta \left[\sum_{i=1}^{n} \ln p_\theta(X_1 = x_1, Y_1) \right]$$

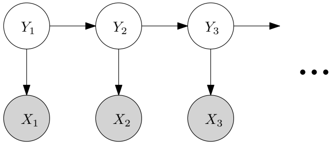
$$= \frac{1}{k} \sum_{i=1}^{n} \sum_{j=1}^{k} p_\theta(Y_i = j | X_1 = x) \ln p_\theta(X_1 = x_1, Y_1 = j),$$

which matches what we optimized before; specifically,

$$p_\theta(Y = j | X = x) = \frac{\pi_j p_\theta(X = x | Y = j)}{\sum_{l=1}^{k} \pi_l p_\theta(X = x | Y = l)}.$$

Expected complete log-likelihood for HMMs.

Graphical model:



Expected complete log-likelihood:

$$\mathbb{E}_\theta\left[\ln p_\theta(X_1,\ldots,X_n,Y_1,\ldots,Y_n)|x_1,\ldots,x_n\right]$$

$$= \mathbb{E}_\theta\left[\ln\left(p_\theta(Y_1)\left(\prod_{i=1}^n p_\theta(X_i|Y_i)\right)\left(\prod_{i=2}^n p_\theta(Y_i|Y_{i-1})\right)\right)\right]$$

$$= \sum_{j=1}^k p_\theta(Y_1|x_1,\ldots,x_k)\ln\pi_j + \sum_{i=1}^n\sum_{j=1}^k p_\theta(Y_i=j|x_1,\ldots,x_n)\ln p_\theta(X_i|Y_i)$$

$$+ \sum_{i\geq 2}\sum_{j,j'=1}^k p_\theta(Y_i=j,Y_{i-1}=j'|x_1,\ldots,x_n)\ln p_\theta(Y_i=j|Y_{i-1}=j')$$

Expected complete log-likelihood for HMMs.

Expected complete log-likelihood:

$$\sum_{j=1}^{k} p_\theta(Y_1|x_1,\ldots,x_k)\ln\pi_j + \sum_{i=1}^{n}\sum_{j=1}^{k} p_\theta(Y_i=j|x_1,\ldots,x_n)\ln p_\theta(X_i|Y_i=j)$$

$$+\sum_{i\geq 2}\sum_{j,j'=1}^{k} p_\theta(Y_i=j,Y_{i-1}=j'|x_1,\ldots,x_n)\ln p_\theta(Y_i=j|Y_{i-1}=j').$$

- M step for observable $p_\theta(X_i|Y_i)$ similar to mixture case; replace old $\sum_i A'_{ij}$ with new $\sum_i p_\theta(Y_i|x_1,\ldots,x_n)$.
- M step for π and $A_{jj'} = p_\theta(Y_i=j|Y_{i-1}=j')$ also easy.
- Real annoyance is computing the conditional probabilities (E step)!

E-step for HMMs.

- Need to compute $p_\theta(Y_1 | x_1, \ldots, x_k)$, $p_\theta(Y_i = j | x_1, \ldots, x_n)$, $p_\theta(Y_i = j, Y_{i-1} = j' | x_1, \ldots, x_n) \ln p_\theta(Y_i = j | Y_{i-1} = j')$.
- Boils down to a bunch of games with conditioning.
  Kindof cool but I decided to skip.
  See Murphy book (Chapter 17) for details.

Summary of HMMs.

- Graphical models give a succinct way to specify conditional dependencies of random variables.
- Expected complete log likelihood is another way to reason about E-M.
- HMMs allow dependence amongst latent variables.

Kernel density estimates (“Parzen windows”).

Let’s cover one more standard distribution modeling tool.

Kernel density estimates (“Parzen windows”).

Let’s cover one more standard distribution modeling tool.

- Let random draw $(x_i)_{i=1}^n$ from some density be given.
- Define $\hat{p}(x) := \frac{1}{n}\sum_{i=1}^n k\left(\frac{x-x_i}{h}\right)$,
  where k is a **kernel function** (not the SVM one!),
  h is the “bandwidth”; for example:
  - Gaussian: $\propto \exp\left(-\|x\|^2/2\right)$;
  - Epanechnikov $\propto \max\{0, 1 - \|x\|^2\}$.

Kernel density estimates: illustration.

Taken from Larry Wasserman's "All of nonparametric statistics":

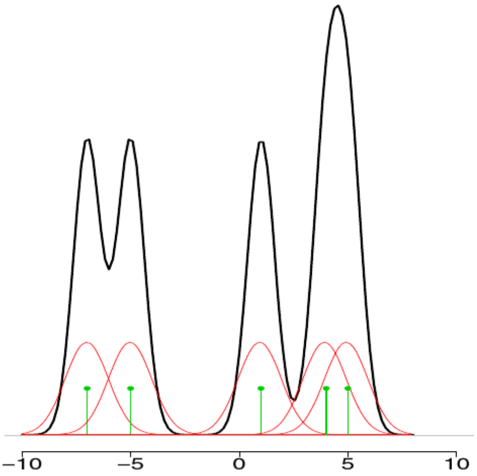


FIGURE 6.4. A kernel density estimator $\widehat{f}_n$. At each point x, $\widehat{f}_n(x)$ is the average of the kernels centered over the data points X_i. The data points are indicated by short vertical bars. The kernels are not drawn to scale.

Kernel density estimates (KDE) vs GMM.

- KDE fits (basically) any density as $n \to \infty$
  (and variance (“bandwidth”) on kernel is tuned).
- GMM fits any density as $k \to \infty$
  (and variance is tuned).
- GMM can succinctly fit some densities
  for which KDE needs many samples.
- KDE is computationally trivial;
  GMM a mess.

Distributions and neural networks.

Distributions and neural networks.

Let's survey our approaches to density estimation.

- Graphical models:
  can be interpretable,
  can encode domain knowledge.
- Kernel density estimation:
  easy to implement, converges to the right thing,
  suffers a curse of dimension.
- **Training:** easy for KDE, messy for graphical models.
  Interpretability: fine for both.
  Sampling: easy for both.
  Probability measurements: easy for KDE, sometimes easy for graphical model.

Distributions and neural networks.

- Neural networks are good at **function approximation**.
- We can use a neural network to approximate a density, **or** we can use it for efficient sampling.
- It seems we can't get both?
  (I guess this is true?)

Modeling densities with neural networks.

- Suppose $(x_i)_{i=1}^n$ drawn from density p.
- Suppose we train a network f to approximate this density (e.g., asking it to approximate the KDE); two issues:
  - How to sample?
    Answer: nasty MCMC methods, for instance Langevin.
  - How to guarantee $\int f = 1$?
    Answer: nasty MCMC again.
    Good luck getting good convergence rates.

Sampling with neural networks: pushforward maps.

Let's discuss **generator networks**;
neural networks used for sampling.

Sampling with neural networks: pushforward maps.

Let's discuss **generator networks**;
neural networks used for sampling.
Basic story:

- Sample x from some efficiently sample-able distribution (uniform, Gaussian).
- Output $g(x)$, where g is a neural network.

Basic properties:

- Clear running time (compare to MCMC!).
- Usual neural network issues (unclear what it's actually doing, unclear how to train).

Pushforward maps?

Recall the definition of a random variable.

- A random variable is a *function* from a sample space to $\mathbb{R}$. E.g., let X denote the sum of two dice. Then $\Pr[X = 3] = (1 + 1)/36$. In general, measure probabilities with inversion: $\Pr[X = 3] = \Pr[X^{-1}(3)] = \Pr[\{(1, 2), (2, 1)\}]$.

A convenient notation is a **pushforward**.

- Let μ denote a single distribution (e.g., one dice).
- Let f be a function over the sample space.
- Let $f\#\mu$ be the distribution that samples $x \sim \mu$ and outputs $f(x)$; equivalently, the probability of a set S is $\Pr_\mu[f^{-1}(S)]$.

Distances over probability spaces?

So we have a way to write down and discuss generator networks:

- $f\#\mu$, where f is a neural network, μ is efficiently sampled.

Distances over probability spaces?

So we have a way to write down and discuss generator networks:

- $f\#\mu$, where f is a neural network, μ is efficiently sampled.

We want $f\#\mu$ to be *close* to some target density p.
How to formalize "close"?

- Standard notion for probability measures: total variation:

$$\frac{1}{2}\int |p(x) - g(x)|\,\mathrm{d}x.$$

  (Picture drawn in class.)
- Problem: very unforgiving for even small errors!

Wasserstein distance.

Easy version of it (suffices for our present purposes): we look at all mappings of one distribution to another. Sometimes this is called "earth mover's distance".

$$W_1(f\#\mu, \nu)$$
$$= \inf \left\{ \int \|f(x) - T(f(x))\| \, \mathrm{d}x \quad : \quad T : \mathbb{R}^d \to \mathbb{R}^d, T\#(f\#\mu) = \nu \right\}.$$

(Pictures drawn in class.)

Summary.

Summary.

- Graphical models and their diagrams.
- HMMs and dependencies amongst latent variables.
- KDE / Parzen windows, comparison to GMMs.
- Densities with neural nets? No thanks.
- Sampling with neural nets? Yes please.