Exam revision

Final exam logistic

- Final exam will be held in person on April 14, at 9AM-12PM Toronto local time in rooms WB 116-119.
- Exam will be 100 points in total and 180 mins long. Students are required to be at the exam location at least 10 mins early, with valid identification. Exam will be administered by FAS.
- You can use one optional A4 **handwritten** aid sheets - double-sided.
- Exam covers all lectures (weeks 1-12), it is closed book/internet.
- A (longer) practice exam has been posted.
- OH will be held as usual next week.

Probabilistic ML Terminology

The final exam will be on the entire course; however, it will be (slightly) more weighted towards post-midterm material. We will go briefly over the main concepts:

- MLE / Exponential families
- Directed Graphical Models
- Decision theory
- Variable elimination
- Message passing
- Hidden Markov Models
- Sampling methods
- Markov chain Monte Carlo
- Variational Inference
- EM algorithm
- Neural networks
- Bayesian regression
- Gaussian processes
- Embeddings/Attention
- Variational autoencoders
- Diffusion models

Definition

Density of a member of the exponential family is of the form:

$$p(x|\eta) = h(x)\exp\{\eta^\top T(x) - A(\eta)\}$$

- $T(x)$: Sufficient statistics
- η: Natural parameter
- $A(\eta)$: Log-partition function
- $h(x)$: carrying measure

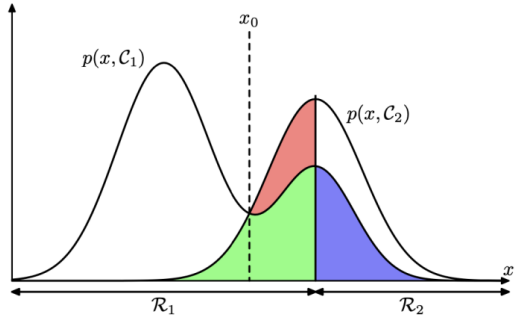
$$\mathbb{E}_{x|\eta}[T(x)] = A'(\eta)$$

Moments of sufficient statistics can be found easily by differentiating the log-partition function!

Examples: Gaussian, Gamma, Exponential, Beta, Dirichlet, Poisson, Geometric... Broad class of distributions!

Week 2: Decision theory, Expected loss

- Minimizing the misclassification rate:



$x \longrightarrow \mathcal{C}_1$ or $\mathcal{C}_2$?

Use $p_\theta(x, \mathcal{C}_1)$

↓

choose maximizer of $p_\theta(x, \cdot)$

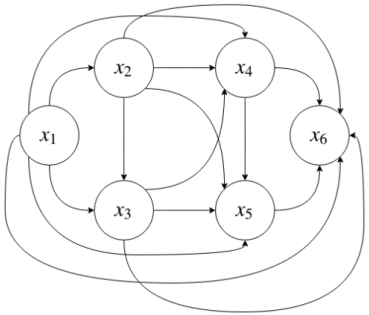
- We use a **loss function** to measure the loss incurred by taking any of the available decisions.
  - e.g. consider a medical diagnosis example; example of a loss matrix:

Truth \ Decision	cancer	normal
cancer	0	1000
normal	1	0

Incorrectly classify as healthy

Incorrectly classify as cancer

Week 2: Directed Acyclic Graphical Models (Bayesian Networks)



- A directed acyclic graphical model (DAGM) implies a factorization of the joint distribution.
- Variables are represented by nodes, and edges represent dependence.

DAGM induces factorization of the joint distribution of $x_1, x_2, \ldots, x_N$:

$$p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i | \text{pa}(x_i))$$

where $\text{pa}(x_i)$ is the set of nodes with edges pointing to x_i.

Pruning algorithm

Allow to read relevant conditional independences from the graph.

Main point

Order in which variables are marginalized affects the computational cost!

Main tool in exact inference is **variable elimination**:

- A simple and general **exact inference** algorithm in any probabilistic graphical model (DAGMs).
- Has computational complexity that depends on the graph structure of the model.
- Sum-product is used to obtain marginals.

Week 3: Complexity of Variable Elimination Ordering

$\sum \prod_i p(x_i \mid \text{parents}(x_i))$

- Different elimination orderings will involve different number of variables appearing inside each sum.

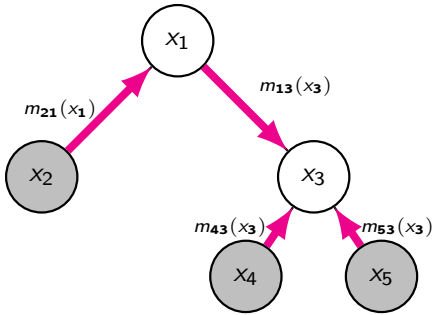
- The complexity of the VE algorithm is

$$\mathcal{O}(mk^{N_{\max}})$$

where

- m is the number of initial factors.
- k is the number of states each random variable takes (assumed to be equal here).
- N_i is the number of random variables inside each sum $\sum_i$.
- $N_{\max} = \max_i N_i$ is the number of variables inside the largest sum.

There is a canonical way to do variable elimination on trees.



$$
\begin{aligned}
p(x_3|x_E) &= \frac{1}{Z^E} \sum_{x_1} \psi_1(x_1)\psi_2(\bar{x}_2)\psi_3(x_3)\psi_4(\bar{x}_4)\psi_5(\bar{x}_5)\psi_{12}(x_1,\bar{x}_2)\psi_{13}(x_1,x_3)\psi_{34}(x_3,\bar{x}_4)\psi_{35}(x_3,\bar{x}_5) \\
&= \frac{1}{Z^E} \underbrace{\psi_4(\bar{x}_4)\psi_{34}(x_3,\bar{x}_4)}_{m_{43}(x_3)} \underbrace{\psi_5(\bar{x}_5)\psi_{35}(x_3,\bar{x}_5)}_{m_{53}(x_3)} \psi_3(x_3) \sum_{x_1} \psi_1(x_1)\psi_{13}(x_1,x_3) \underbrace{\psi_2(\bar{x}_2)\psi_{12}(x_1,\bar{x}_2)}_{m_{21}(x_1)} \\
&= \frac{1}{Z^E} m_{43}(x_3)m_{53}(x_3)\psi_3(x_3) \underbrace{\sum_{x_1} \psi_1(x_1)\psi_{13}(x_1,x_3)m_{21}(x_1)}_{m_{13}(x_3)} \\
&= \frac{1}{Z^E} \psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3) \quad = \quad \frac{\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3)}{\sum_{x'_3} \psi_3(x'_3)m_{43}(x'_3)m_{53}(x'_3)m_{13}(x'_3)}
\end{aligned}
$$

- The message sent from variable j to $i \in N(j)$ is

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus\{i\}} m_{k\to j}(x_j)$$

  - ⚠If x_j is observed, the message is instead

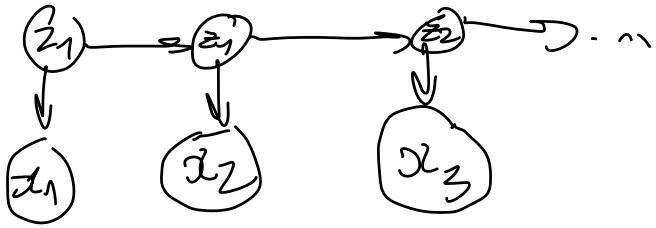
$$m_{j\to i}(x_i) = \psi_j(\bar{x}_j)\psi_{ij}(x_i, \bar{x}_j) \prod_{k\in N(j)\setminus\{i\}} m_{k\to j}(\bar{x}_j)$$

- To compute all marginals, two passes are needed: one from leaves to root, one from root to leaves. Once the message passing stage is complete, compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$$

and normalize.

Week 4: Hidden Markov Models



- Important DAGMs to simplify the joint distribution.
- Posterior inference takes the special form:

$$p(z_t|x_{1:T}) \propto p(z_t, x_{1:t})p(x_{t+1:T}|z_t) \propto \text{(Forward Recursion)(Backward Recursion)}$$

- **Forward-backward algorithm** to compute $p(z_t|x_{1:T})$

Week 4: Estimation method, Simple Monte Carlo

Estimation problem using simple Monte Carlo:

- **Simple Monte Carlo**: Given $\{x^{(r)}\}_{r=1}^{R} \sim p(x)$ we can estimate the expectation $\mathbb{E}_{x\sim p(x)}[\phi(x)]$ using the estimator $\hat{\Phi}$:

$$\Phi := \mathbb{E}_{x\sim p(x)}[\phi(x)] \approx \frac{1}{R}\sum_{r=1}^{R}\phi(x^{(r)}) := \hat{\Phi}$$

- The fact that $\hat{\Phi}$ is a consistent estimator of Φ follows from the Law of Large Numbers (LLN).

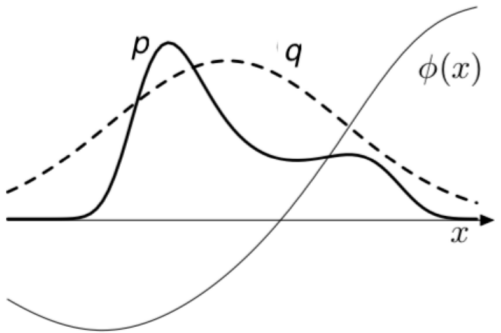
$$+\ \mathrm{Var} = O\left(\frac{1}{R}\right)$$

Assume:

- Target $p(x)$ can be evaluated up to normalizing constant $\tilde{p}(x)$
- There is a simpler density, $q(x)$ from which it is easy to sample from and can evaluate up to normalizing constant $\tilde{q}(x)$

$$\text{Sample:} \quad x^{(r)} \sim q(x) = \tilde{q}(x)/Z_q$$

Importance sampling: estimate the expectation of a function $\phi(x)$.



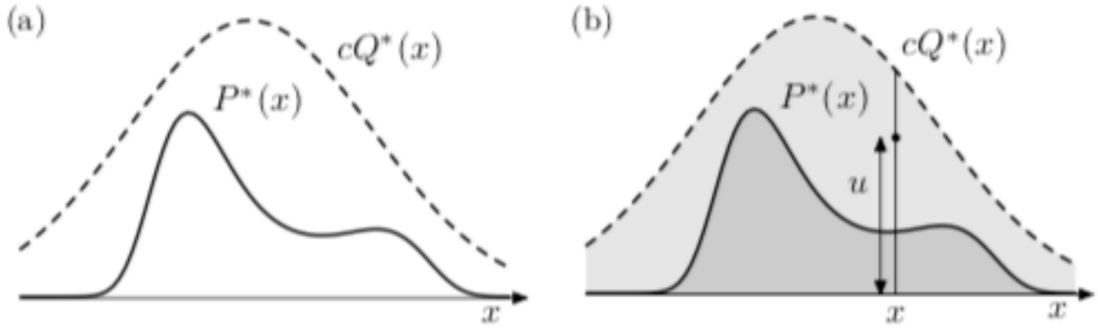
- Introduce weights: $\tilde{w}_r = \frac{\tilde{p}(x^{(r)})}{\tilde{q}(x^{(r)})}$
- The importance weighted estimator of $\mathbb{E}_p\phi(x)$

$$\hat{\Phi}_{iw} = \sum_{r=1}^{R} \phi(x^{(r)}) \cdot w_r$$

where $w_r = \frac{\tilde{w}_r}{\sum_{r=1}^{R} \tilde{w}_r}$

$\tilde{p}(x) \in c\tilde{q}(x)$

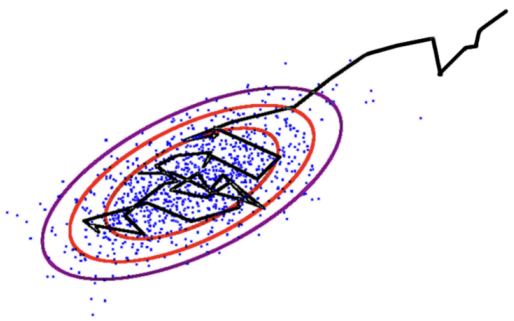
Week 4: Rejection sampling



The procedure is as follows:

1. Generate two random numbers.
   1. x is generated from $q(x)$.
   2. u is generated uniformly from the interval $[0, cq^*(x)]$.
2. Accept or reject the sample x by comparing the value of u with $p^*(x)$
   1. If $u > p^*(x)$, then x is rejected.
   2. Otherwise x is accepted; x is added to our set of samples $\{x^{(r)}\}$.

Week 5: Markov Chain Monte Carlo (MCMC)



- In contrast to rejection sampling, where the accepted points $\{x^{(t)}\}$ are independent, MCMC methods generate a dependent sequence.
- Each sample $x^{(t)}$ has a probability distribution that depends on the previous value, $x^{(t-1)}$.
- MCMC methods need to be run for a time in order to generate samples that are from the target distribution p.

We can still do Monte Carlo estimation for large enough T to estimate the mean of a test function ϕ:

$$\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{T}\sum_{t=1}^{T} f(x^{(t)}).$$

(also a good idea to discard a bunch of initial samples)

Week 5: Metropolis-Hastings algorithm

As before, assume we can evaluate $\tilde{p}(x)$ for any x. Our procedure:

- A tentative new state x' is generated from the proposal density $q(x'|x^{(t)})$. We accept the new state with probability

$$A(x'|x^{(t)}) = \min\left\{1, \frac{\tilde{p}(x')q(x^{(t)}|x')}{\tilde{p}(x^{(t)})q(x'|x^{(t)})}\right\}$$

  - If accepted, set $x^{(t+1)} = x'$. Otherwise, set $x^{(t+1)} = x^{(t)}$.

- Metropolis: Simpler version when $q(x'|x) = q(x|x')$ for all x, x'.

- **Theorem:** This procedure defines a Markov chain with stationary distribution $\pi(x)$ equal to the target distribution $p(x)$.

$\tilde{p}(x_1, \cdots, x_d)$

Week 5: Gibbs Sampling Procedure

Suppose the vector x has been divided into d components

$$x = (x_1, \ldots, x_d).$$

Start with any $x^{(0)} = (x_1^{(0)}, \ldots, x_d^{(0)})$. In the t-th iteration:

- For $j = 1, \ldots, d$:
  - Sample $x_j^{(t)}$ from the conditional distribution given other components:

$$x_j^{(t)} \sim p(x_j | x_{-j}^{(t-1)}) = \frac{p(x_j, x_{-j}^{(t-1)})}{p(x_{-j}^{(t-1)})}$$

    Where $x_{-j}^{(t-1)}$ represents all the components of x except for x_j at their current values:

$$x_{-j}^{(t-1)} = (x_1^{(t)}, x_2^{(t)}, \ldots, x_{j-1}^{(t)}, x_{j+1}^{(t-1)}, \ldots, x_d^{(t-1)})$$

- No accept/reject, only accept.

The conditional distribution does not depend on the normalizing constant.

The HMC algorithm (run until it mixes):

- Current position: $(x^{(t-1)}, v^{(t-1)})$
- Sample momentum: $v^{(t)} \sim \mathcal{N}(0, I)$.
- Start at $(x, v) = (x^{(t-1)}, v^{(t)})$ and run Leapfrog integrator for L steps and reach (x', v'). (a point with approximately the same total energy)
- Accept new state $(x', -v')$ with probability:

$$\min\left\{1, \frac{\exp(H(x^{(t-1)}, v^{(t-1)}))}{\exp(H(x', v'))}\right\}$$

→ Metropolis algorithm

$q(x' | x^{(t-1)}) = q(x^{(t-1)} | x')$

- Low energy points are favored.

Week 6: Variational Inference and KL divergence

We will measure the difference between q and p using the **Kullback-Leibler divergence**

$$KL(q(z)||p(z)) = \int q(z) \log \frac{q(z)}{p(z)} dz \quad \text{or} \quad = \sum_z q(z) \log \frac{q(z)}{p(z)}$$

Properties of the KL Divergence

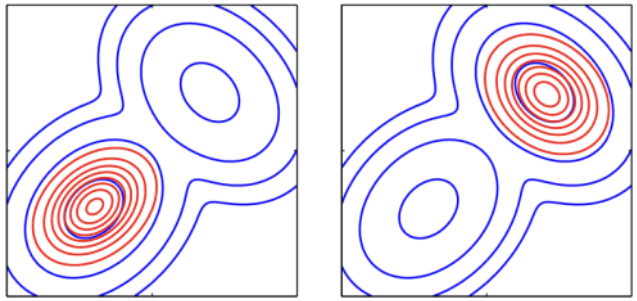
- $KL(q||p) \geqslant 0$
- $KL(q||p) = 0 \Leftrightarrow q = p$
- $KL(q||p) \neq KL(p||q)$
- KL divergence is not a metric, since it is not symmetric

Week 6: Information (I-)Projection

I-projection: $q^* = \arg\min_{q \in \mathcal{Q}} KL(q||p) = \mathbb{E}_{x \sim q(x)} \log \frac{q(x)}{p(x)}$.

- $p \approx q \implies KL(q||p)$ small
- I-projection underestimates support, and does not yield the correct moments.
- $KL(q||p)$ penalizes q having mass where p has none.

$p(x)$ is mixture of two 2D Gaussians and $\mathcal{Q}$ is the set of all 2D Gaussian distributions (with arbitrary covariance matrices)



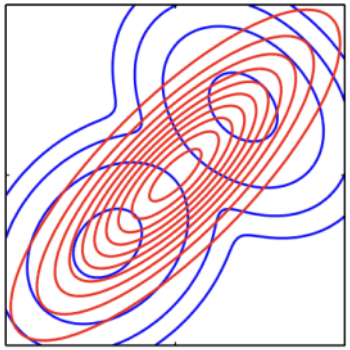
p=Blue, q^*=Red (two equivalently good solutions!)

Week 6: Moment (M-)projection

M-projection: $q^* = \arg\min_{q \in \mathcal{Q}} KL(p||q) = \mathbb{E}_{x \sim p(x)} \log \frac{p(x)}{q(x)}$.

- $p \approx q \implies KL(p||q)$ small
- $KL(p||q)$ penalizes q missing mass where p has some.
- M-projection yields a distribution $q(x)$ with the correct mean and covariance if $\mathcal{Q}$ is the family of Gaussians.

$p(x)$ is mixture of two 2D Gaussians and $\mathcal{Q}$ is the set of all 2D Gaussian distributions (with arbitrary covariance matrices)



p=Blue, q^*=Red

ELBO is a lower bound on the (log) evidence. Maximizing the ELBO is the same as minimizing $KL(q_\phi(z)||p(z|x))$.

$$KL(q_\phi(z)||p(z|x)) = \mathbb{E}_{z\sim q_\phi} \log \frac{q_\phi(z)}{p(z|x)} = \mathbb{E}_{z\sim q_\phi}\left[\log\left(q_\phi(z)\cdot\frac{p(x)}{p(z,x)}\right)\right]$$
$$= \mathbb{E}_{z\sim q_\phi}\left[\log\frac{q_\phi(z)}{p(z,x)}\right] + \mathbb{E}_{z\sim q_\phi}\log p(x) := -\mathcal{L}(\phi) + \log p(x).$$

Where $\mathcal{L}(\phi)$ is the **ELBO**: $\mathcal{L}(\phi) = \mathbb{E}_{z\sim q_\phi}[\log p(z,x) - \log q_\phi(z)]$.

- Because $KL(q_\phi(z)||p(z|x)) \geqslant 0$,
$$\mathcal{L}(\phi) \leqslant \log p(x)$$
- maximizing the ELBO $\implies$ minimizing $KL(q_\phi(z)||p(z|x))$.

- Full dataset $\{\mathbf{X}, \mathbf{Z}\}$ but we only observe $\{\mathbf{X}\}$. The model for (x, z) is tractable.

- Our knowledge about the latent variables is given only by the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta)$.

- Because we cannot use the complete data log-likelihood, we can consider expected complete-data log-likelihood:

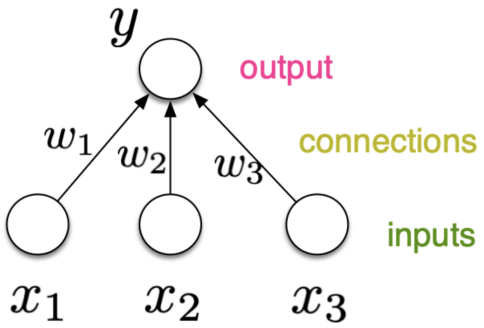
$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \log p(\mathbf{X}, \mathbf{Z}|\theta)$$

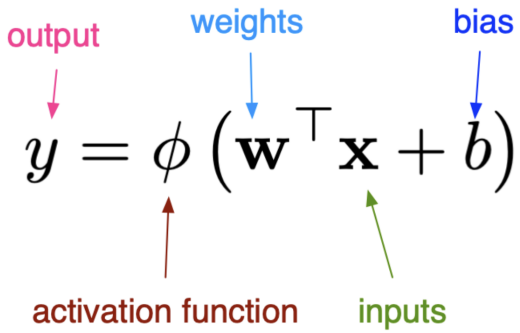
- In the E-step, we use the current parameters θ^{old} to compute the posterior over the latent variables $p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$.

- In the M-step, we find the revised parameter estimate θ new by maximizing the expected complete log-likelihood:

$$\theta^{\text{new}} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}})$$

For neural nets, we use a simple model for neuron, or **unit**:



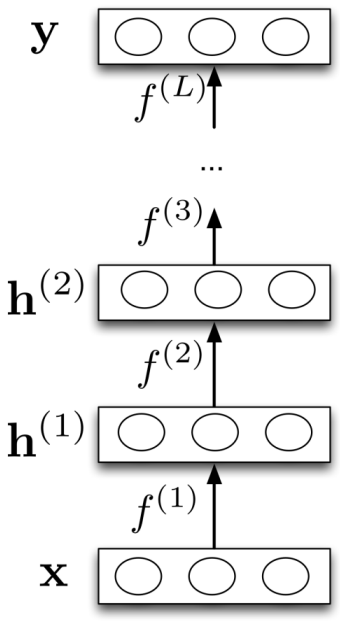


- Same as logistic regression: $y = \sigma(\mathbf{w}^\top \mathbf{x} + b)$
- By throwing together lots of these simple neuron-like processing units, we can do some powerful computations!

Week 9: Computation in Each Layer

Each layer computes a function.

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}) = \phi(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\mathbf{h}^{(2)} = f^{(2)}(\mathbf{h}^{(1)}) = \phi(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$
$$\vdots$$
$$\mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)})$$



The network computes a composition of functions.

$$\mathbf{y} = f^{(L)} \circ \cdots \circ f^{(1)}(\mathbf{x}).$$

The last layer depends on the task.

- Regression: $\mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)}) = (\mathbf{w}^{(L)})^\top \mathbf{h}^{(L-1)} + b^{(L)}$
- Classification: $\mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)}) = \sigma((\mathbf{w}^{(L)})^\top \mathbf{h}^{(L-1)} + b^{(L)})$

↳ softmax

Week 9: Backpropagation Algorithm

Learning parameters in NN is typically done with SGD, where the gradient is computed using backpropagation.

Let $v_1, \ldots, v_N$ be an ordering of the computation graph where parents come before children. v_N denotes the variable for which we try to compute gradients ($\mathcal{L}$, $\mathcal{L}_{reg}$ etc).

- forward pass:

$$\text{For } i = 1, \ldots, N,$$
$$\text{Compute } v_i \text{ as a function of Parents}(v_i).$$

- backward pass (denote $\bar{v}_i = \frac{\partial v_N}{\partial v_i}$): start setting $\bar{v}_N = 1$

$$\text{Then for } i = N-1, \ldots, 1: \qquad \bar{v}_i = \sum_{j \in \text{Children}(v_i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}$$

- **Prior distribution:** $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{S})$
- **Model (Likelihood):** $y \mid \mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{w}^\top \psi(\mathbf{x}), \sigma^2)$
- Assuming fixed/known $\mathbf{S}$ and σ^2.
- **Deriving the posterior distribution:**

$$\begin{aligned}
\log p(\mathbf{w} \mid \mathcal{D}) &= \log p(\mathbf{w}) + \log p(\mathcal{D} \mid \mathbf{w}) + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\top \mathbf{S}^{-1}\mathbf{w} - \frac{1}{2\sigma^2}\|\boldsymbol{\Psi}\mathbf{w} - \mathbf{y}\|^2 + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\top \mathbf{S}^{-1}\mathbf{w} - \frac{1}{2\sigma^2}\left(\mathbf{w}^\top \boldsymbol{\Psi}^\top \boldsymbol{\Psi}\mathbf{w} - 2\mathbf{y}^\top \boldsymbol{\Psi}\mathbf{w} + \mathbf{y}^\top \mathbf{y}\right) + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\top \left(\sigma^{-2}\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \mathbf{S}^{-1}\right)\mathbf{w} + \frac{1}{\sigma^2}\mathbf{y}^\top \boldsymbol{\Psi}\mathbf{w} + \text{const}
\end{aligned}$$

complete the square

Thus $\mathbf{w} \mid \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Psi}^\top \mathbf{y}, \quad \boldsymbol{\Sigma} = \left(\sigma^{-2}\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \mathbf{S}^{-1}\right)^{-1}$$

Linear model: $y \mid \mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2) \quad f(\mathbf{x}) = \mathbf{w}^\top \psi(\mathbf{x})$

- Given N samples, we have: $\mathbf{y} \mid \mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$
- Since $\mathbf{f}$ is a Gaussian process ($\mathbf{w}$ is Gaussian), we have $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$

$$K_{ij} = \frac{1}{a} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \psi(\mathbf{x}^{(i)})^\top \mathbf{S} \psi(\mathbf{x}^{(j)})$$

- Therefore the marginal of $\mathbf{y}$ is given by

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}) \qquad \mathbf{C} = \mathbf{K} + \sigma^2 I$$

where

$$C(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sigma^2 \delta_{ij}$$

$\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$.

- Let's define $\mathbf{y}_N = (y^{(1)}, y^{(2)}, \ldots, y^{(N)})^\top$.
- We have the marginal of $\mathbf{y}_N$ given by

$$\mathbf{y}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_N) \qquad \mathbf{C}_N = \mathbf{K}_N + \sigma^2 I$$

  where $C_N(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sigma^2 \delta_{ij}$.

- This reflects the two Gaussian sources of randomness.
- **Goal in regression:** We want to predict for a new input $\mathbf{x}^{(N+1)}$. We need

$$p(y^{(N+1)} \mid \mathbf{y}_N)$$

- Note that $\mathbf{x}^{(i)}$'s are treated as constants.

- We have

$$\mathbf{y}_{N+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_{N+1}) \qquad \mathbf{C}_{N+1} = \mathbf{K}_{N+1} + \sigma^2 I$$

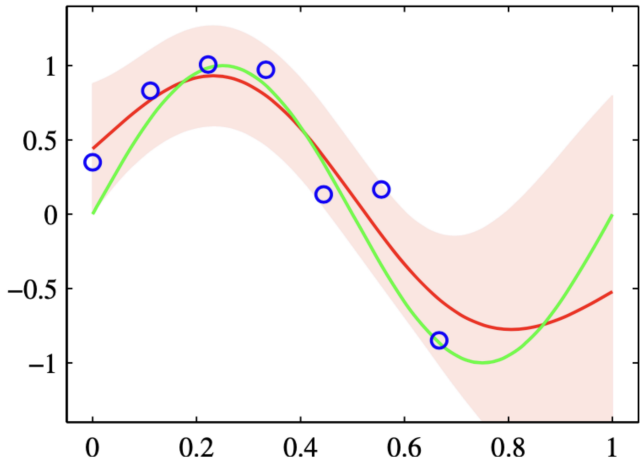
where

$$C_{N+1}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sigma^2 \delta_{ij}$$

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{bmatrix}.$$

  - Here, $c = k(\mathbf{x}^{(N+1)}, \mathbf{x}^{(N+1)}) + \sigma^2$
  - $\mathbf{k}$ is a vector with entries $k_i = k(\mathbf{x}^{(i)}, \mathbf{x}^{(N+1)})$
- Since $\mathbf{y}_{N+1}$ is multivariate Gaussian, $y^{(N+1)} \mid \mathbf{y}_N$ is also Gaussian with mean and covariance

$$m(\mathbf{x}^{(N+1)}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}_N \qquad \sigma^2(\mathbf{x}^{(N+1)}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}$$



- The green curve: the true sinusoid from which the data points, shown in blue, are obtained with additional of Gaussian noise.
- The red line: mean of the Gaussian process predictive distribution.
- The shaded region: plus and minus two standard deviations.

We start with a fairly strong assumption:

"Words that have similar meanings will occur in similar contexts"

Based on that we define a **context** of size **k** of token $x_{i,j}$ as a set of tokens:

$$\text{context}(x_{i,j}) = \{x_{i,j-k}, x_{i,j-(k-1)}, \ldots, x_{i,j-1}, x_{i,j+1}, \ldots, x_{i,j+k}\}$$ CBoW

Then given a set of datapoints $x_{i=1:N,j=1:M_i}$ and a vocabulary $V_{r=1:M}$ we define an unsupervised learning task of predicting what words occur in the context of each word in the vocabulary. More formally, given a sequence of training words $x_1, \ldots, x_T$ we want to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in\text{context}(t)} \log p(w_j|w_t)$$

[1] This is also called a **skip-gram**

The basic formulation of $p(x|w)$ uses the softmax function:

$$\underset{\hat{\theta}}{p}(x|w) = \frac{\exp((u(w))^T(v(x)))}{\sum_{r=1}^{M}\exp((u(w))^T(v(V_r)))}$$

where $u(w)$ is the "word" and $v(w)$ is the "context" representation of word w i.e the bBOW.

- In our particular case we will take u and v to be simple linear projections of the **one-hot** (binary BoW) encoding of the word, and context respectively.

$$u(w) = U \cdot bBoW(w), \qquad v(w) = V \cdot bBoW(w)$$

The matrices U, V will be of size $e \times M$ where e is the embedding dimension, which is a hyperparameter you chose.

We can also learn which of the input words are the most important!

We will begin by creating 3 separate embeddings from each of our inputs, by simply multiplying them by (learned) matrices:

$$q = W^Q x \qquad \text{(query)}$$
$$k = W^K x \qquad \text{(key)}$$
$$v = W^V x \qquad \text{(value)}$$

We then define the **Attention Layer** as:

$$\text{Attn}(q, k, v) = \sum_{i=1}^{m} \alpha_i(q, k_i) v_i$$

where α is the scoring function.

$$\text{Attn}(q, k, v) = \sum_{i=1}^{m} \alpha_i(q, k_i) v_i$$

The most common choice of the attention function is called the **dot product attention**. We obtain the scores by a normalized dot product of the k and q vectors.

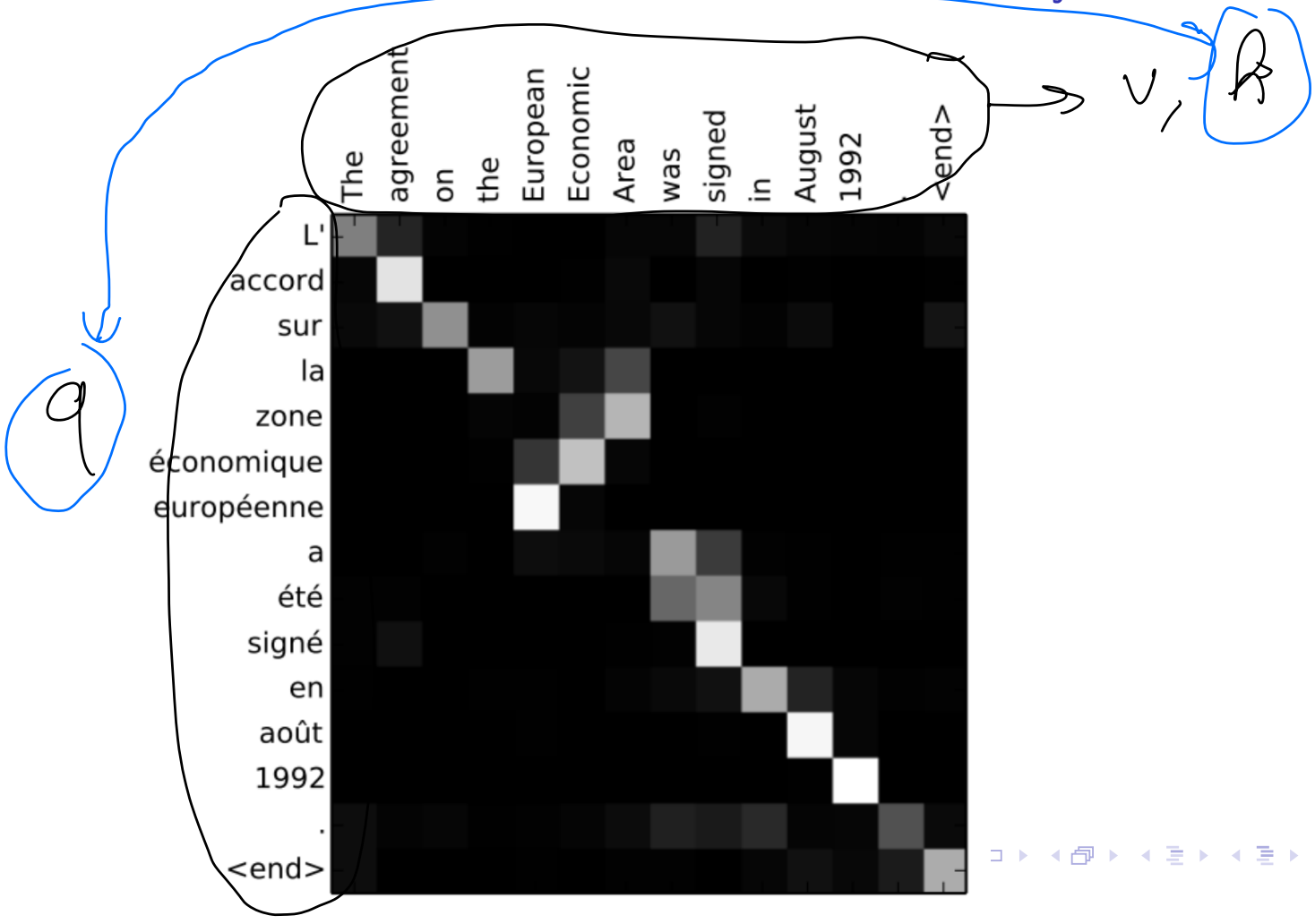
$$b(q, k) = \frac{q^T k}{\sqrt{d}}$$

where d is a normalizing constant, usually the dimensionality of the vectors.

We then set our attention weights α_i to be the softmax of all the scores:

$$\alpha_i(q, k_i) = \frac{\exp(b(q, k_i))}{\sum_{j=1}^{m} \exp(b(q, k_j))}$$

Week 11: Attention is all you need

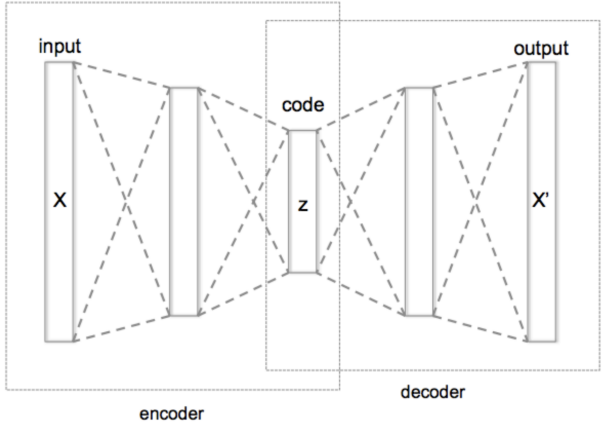


Autoencoders reconstruct their input via an encoder and a decoder.

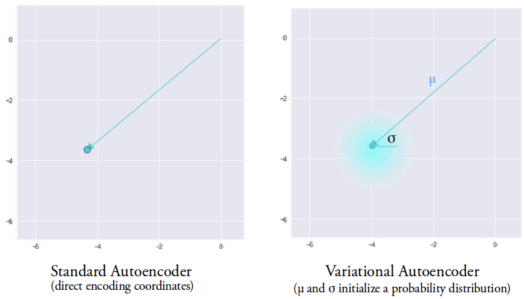
- **Encoder**: $g(x) = z \in F, \quad x \in X$
- **Decoder**: $f(z) = \tilde{x} \in X$
- where X is the data space, and F is the feature (latent) space.
- z is the code, compressed representation of the input, x. It is important that this code is a bottleneck, i.e. that

$$\dim F \ll \dim X$$

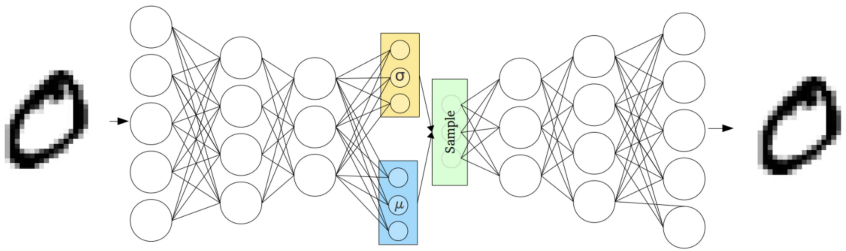
- Goal: $\tilde{x} = f(g(x)) \approx x$.



- The mean μ controls where encoding of input is centered while the standard deviation controls how much can the encoding vary.



- Encodings are generated at random from the "circle", the decoder learns that all nearby points refer to the same input.



Week 12: VAE objective

- Recall the idea behind Variational Inference:

$$\begin{aligned}\mathcal{L}(\theta, \phi|x) = \text{ELBO} &= \log p_\theta(x) - \text{KL}(q_\phi(z|x)||p_\theta(z|x)) \\ &= \mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(x|z)\right] - \text{KL}(q_\phi(z|x)||p(z))\end{aligned}$$

  which is the (negative) loss function we use when training VAEs.

- First term in blue is the expected log-likelihood and the second is the divergence of q_ϕ from the prior.

- The encoder and decoder in a VAE become:
  - **Encoder**: $q_\phi(z|x)$, where
$$q_\phi(z|x) = \mathcal{N}(\mu(x), \Sigma_\phi(x)),$$
    where $\mu_\phi(x)$ and diagonal $\Sigma_\phi(x)$ are computed by a deep NN
  - **Decoder**: $p(x|z) \sim \text{ExpFam}(x|d_\theta(z))$, where $d_\theta(z)$ is typically a deep neural network

High level summary

We focused in this course on probabilistic models that scale to big examples.

Some remarks:

- Graphical representations of models allow for structured and scalable aproaches.
- Conditional independence is interpretable and provides computational advantages.
- Incorporating latent variables increases flexibility with tiny computational cost.
- Variational inference and other techniques based on EFs have been useful.
- In modern ML algorithms models and computation cannot be separated.
- Many new approaches in ML creatively recycle old methods.

Study for the final!

- Review lectures.
- Understand derivations (ask, check textbooks).
- Solve the practice final.
- Fill out course evaluations!
- Good luck!