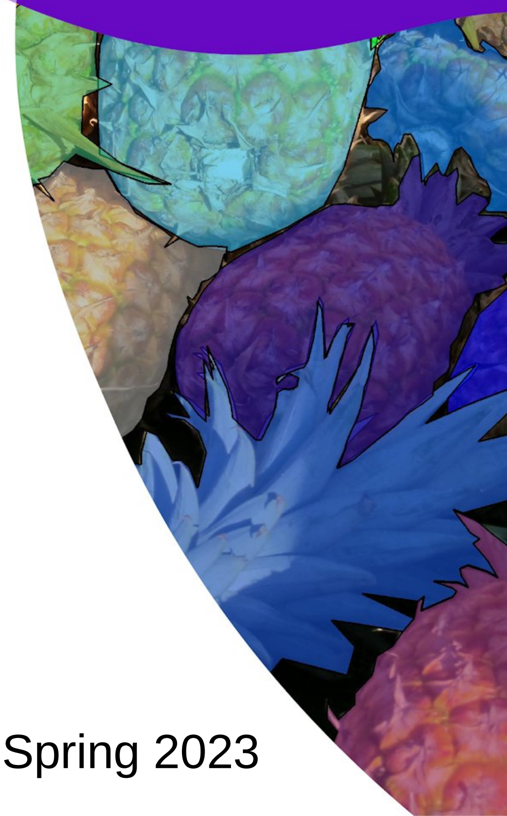
# Energy-Based Models, Variational Methods

Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research
http://yann.lecun.com



Deep Learning, NYU, Spring 2023

FACEBOOK AI

# EBM
# &
# Probabilistic Models

# Refresher on turning energies to probabilities

- **Gibbs distribution (a.k.a. softmax, should be called softargmax)**
- **Discrete / Continuous**

$$P_w(y) = \frac{e^{-\beta F_w(y)}}{\sum_{y'} e^{-\beta F_w(y')}} \qquad P_w(y) = \frac{e^{-\beta F_w(y)}}{\int_{y'} e^{-\beta F_w(y')}}$$

- **Joint distribution**

$$P_w(y,z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'}\int_{z'} e^{-\beta E_w(y',z')}}$$

Partition function

Inverse temperature

- **Conditional distribution**

$$P_w(y,z|x) = \frac{e^{-\beta E_w(x,y,z)}}{\int_{y'}\int_{z'} e^{-\beta E_w(x,y',z')}}$$

- **Marginal distribution**

$$P_w(y|x) = \int_{z'} P_w(y,z'|x) = \frac{\int_{z'} e^{-\beta E_w(x,y,z')}}{\int_{y'}\int_{z'} e^{-\beta E_w(x,y',z')}}$$

# Refresher on turning energies to probabilities

- **Joint distribution**

$$P_w(y, z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'} \int_{z'} e^{-\beta E_w(y',z')}}$$

- **Conditional distribution**

$$P_w(y|z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'} e^{-\beta E_w(y',z)}}$$

- **Marginal distribution**

$$P_w(z) = \frac{\int_{y'} e^{-\beta E_w(y',z)}}{\int_{z'} \int_{y'} e^{-\beta E_w(y',z')}}$$

- **Bayes rules!**

$$P_w(y, z) = P_w(y|z) P_w(z) = P_w(z|y) P_w(y)$$

# Negative log-likelihood loss

$$L(x,y,w) = -\frac{1}{\beta}\log P_w(y|x) = F_w(x,y) + \frac{1}{\beta}\log\left[\int_{y'} e^{-\beta F_w(x,y')}\right]$$

Minus log partition function.

Like a free energy over y.

- **Gradient of log partition function**

$$\frac{\partial\left[-\frac{1}{\beta}\log\left[\int_{y'} e^{-\beta F_w(x,y')}\right]\right]}{\partial w} = \int_{y'} P_w(y'|x)\frac{\partial F_w(x,y')}{\partial w}$$

- **Monte Carlo methods: sample y from P(y|x)**
  - The integral is an expectation of the gradient over the distribution of y
  - Sample y from the distribution and average the corresponding gradients.

# Max Likelihood is (generally) a (bad) Contrastive Method

- **Push down on data points,**
- **Push up on all points**
- **Max likelihood / probabilistic models**

$$P_w(y|x) = \frac{e^{-\beta F_w(x,y)}}{\int_{y'} e^{-\beta F_w(x,y')}}$$

- Loss: $$\mathcal{L}(x,y,w) = F_w(x,y) + \frac{1}{\beta}\log\int_{y'} e^{-\beta F_w(x,y')}$$
- Gradient: $$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \int_{y'} P_w(y'|x)\frac{\partial F_w(x,y')}{\partial w}$$
- 2nd term is intractable: MC/MCMC/HMC/CD: $\hat{y}$ sampled from $P_w(y|x)$

$$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \frac{\partial F_w(x,\hat{y})}{\partial w}$$

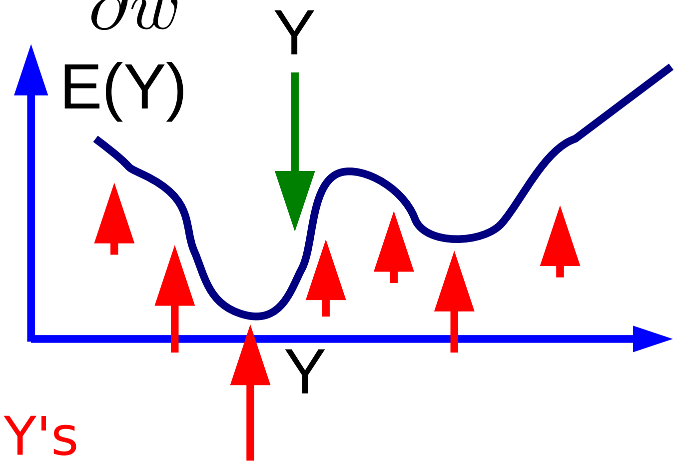
# push down of the energy of data points, push up everywhere else

Gradient of the negative log-likelihood loss for one sample Y:

$$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \int_{y'} P_w(y'|x) \frac{\partial F_w(x,y')}{\partial w}$$

Gradient descent:

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}(x,y,w)}{\partial w}$$



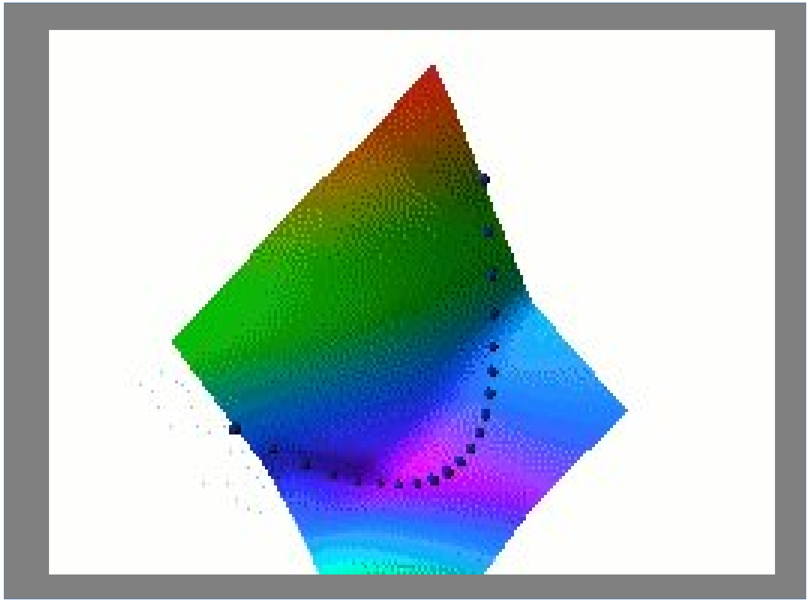

Pushes down on the energy of the samples

Pulls up on the energy of low-energy Y's

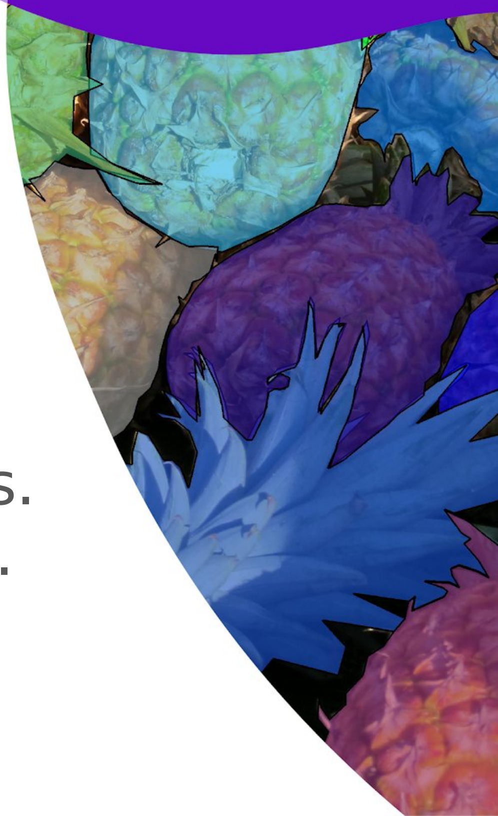
$$w \leftarrow w - \eta \frac{\partial F_w(x,y)}{\partial w} + \eta \int_{y'} P_w(y'|x) \frac{\partial F_w(x,y')}{\partial w}$$

# Problem with Max Likelihood / Probabilistic Methods

- **It wants to make the difference between the energy on the data manifold and the energy just outside of it infinitely large!**
- **It wants to make the data manifold an infinitely deep and infinitely narrow canyon.**
- **The loss must be regularized to keep the energy smooth**
  - e.g. with Bayesian prior or by limiting weight sizes à la Wasserstein GAN.
  - So that gradient-based inference works
  - Equivalent to a Bayesian prior
  - But then, why use a probabilistic model?



# Regularization through (variational) marginalization.

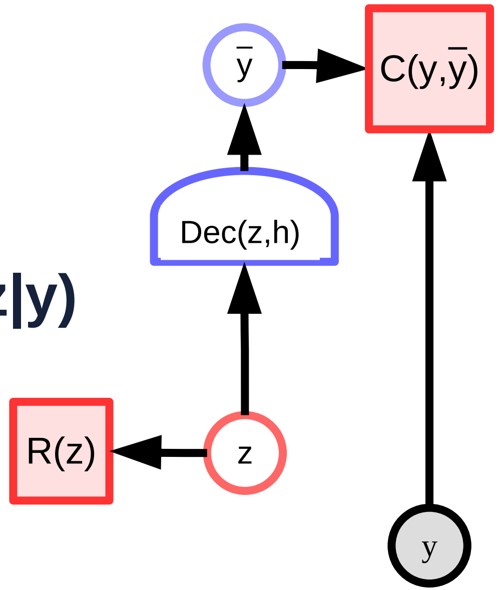


Push down on the energy of training samples.
Minimize the capacity of the latent variables.
Maximize the capacity of the representation.

# Making z a noisy variable to reduce its information content

- **The information content of the latent variable z must be minimized**
- **One (probabilistic) way to do this:**
  - make z "fuzzy" (e.g. stochastic)
  - Z is a sample from a distribution q(z|y)

$$E(y,z) = C(y, Dec(z))$$




- **Minimize the expected value of the energy under q(z|y)**

$$< E(y) > = \int_z q(z|y) E(y,z)$$

- **Minimize the information content of q(z|y) about y**

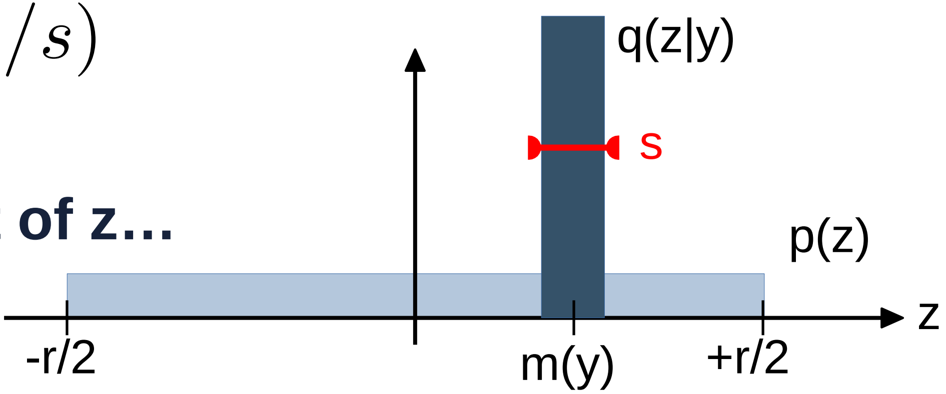
# What information does q(z|y) give us about y?

- **Suppose that all the z come from a distribution p(z)**
  - e.g. p(z) uniform over a hypercube of dimension d:  $[-r/2, +r/2]^d$
- **Suppose that q(z|y) is uniform**
  - over a small hypercube of size s centered on m(y)
  - e.g. q(z|y) uniform over $[m_i(y)-s/2, m_i(y)+s/2]$ in each dimension i.
- **There are $(r/s)^d$ small cubes in the big cube**
  - Hence each small cube gives

$$H(z|y) = \log_2 (r/s)^d = d \log_2 (r/s)$$

bits of information about y.

- **To minimize the information content of z…**
  - I can make the small cube large
  - I can make the large cube small



# What information does q(z|y) give us about y?

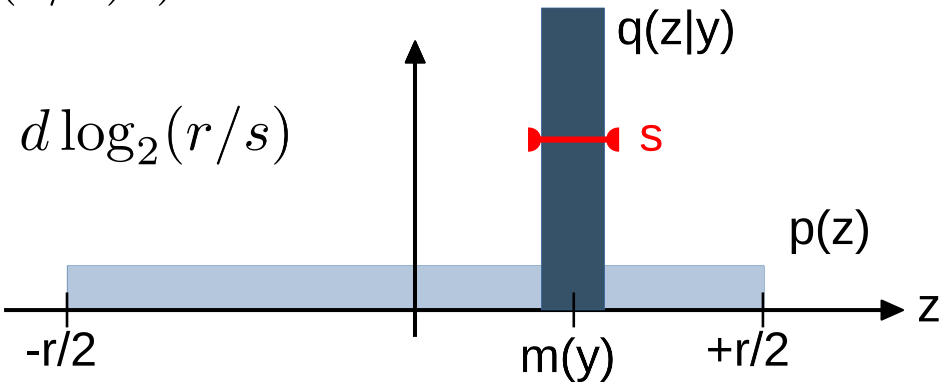
- Suppose that all the z come from a distribution p(z)
- Suppose that each z distributes according to q(z|y)
- The amount of information that q(z|y) gives about p(z) is

$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2(q(z|y)/p(z))$$

- Example: uniform case: $p(z) = (1/r)^d$ , $q(z|y) = (1/s)^d$

$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2((1/s)^d/(1/r)^d)$$

$$KL(q(z|y), p(z)) = \log_2(r/s)^d \int_z q(z|y) = d \log_2(r/s)$$



# General case: minimize expected energy & information of z on y

- **Minimize the expected energy**

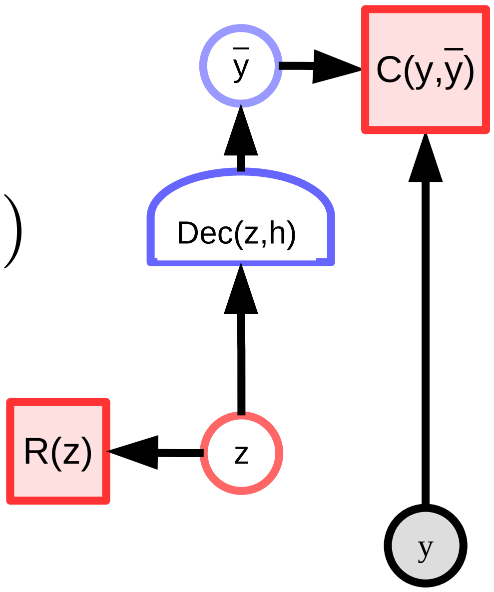
$$< E(y) >_q = \int_z q(z|y) E(y,z)$$

$$E(y,z) = C(y, Dec(z))$$

- **Minimize the relative entropy**
  - Between q(z|y) and a prior distribution p(z).

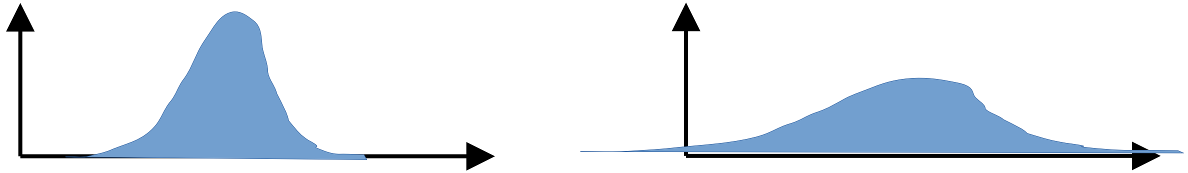
$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2(q(z|y)/p(z))$$

  - This is the number of bits one sample from q(z|y) will give us about p(z)



# Marginalization as Regularization through Maximum Entropy

- **Find a distribution q(z|y) that minimizes the expected energy while having maximum entropy**
  - high entropy distribution == small information content from a sample



  - Pick a family of distributions q(z|y) (e.g. Gaussians) and find the one that minimizes the **variational free energy**:

$$\tilde{F}_q(y) = \int_z q(z|y)E(y,z) + \frac{1}{\beta}\int_z q(z|y)\log_2(q(z|y)/p(z))$$

  - The trade-off between energy and entropy is controlled by the beta parameter.

# Gaussian case

- **Both p(z) and q(z|y) are Gaussians**

$$p(z) = N(0, r) \qquad\qquad q(z|y) = N(m(y), s(y))$$

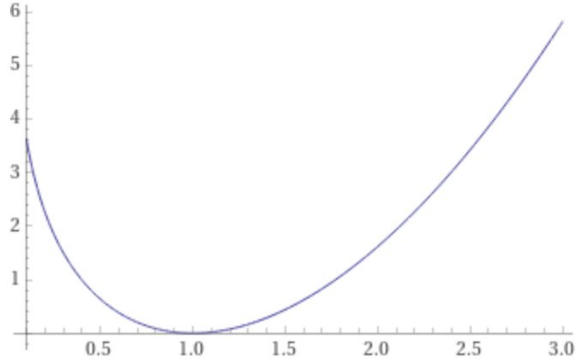
$$KL(q,p) = \log(r/s(y)) + \frac{m(y)^2 + s(y)^2}{2r^2} - \frac{1}{2}$$

(this is in nats, not bits. Divide by log(2) to get it in bits).

- **Assume r=1:**

$$KL(q,p) = \frac{1}{2}(m(y)^2 + s(y)^2 - \log_2(s(y)^2) - 1)$$

  - This has a minimum at s=1



# Marginalization as Regularization through Maximum Entropy

Y. LeCun

$$\tilde{F}_q(y) = \int_z q(z|y)E(y,z) + \frac{1}{\beta}\int_z q(z|y)\log(q(z|y)/p(z))$$

- If the family q(z|y) is flexible enough, the q*(z|y) that minimizes the variational free energy is the Gibbs distribution:

$$q^*(z|y) = \frac{e^{-\beta E(y,z)}}{\int_{z'} e^{-\beta E(y,z')}}$$

- With this q*(z|y), the variational free energy becomes the free energy:

$$\tilde{F}_{q^*}(y) = F(y) = -\frac{1}{\beta}\log\int_z e^{-\beta E(y,z)}$$

- The Gibbs distribution on z is the one best trade-off between minimizing the expected energy and maximizing its entropy

# Variational Inference

- **Variational approximation of marginalization over z**

$$F(y) = -\frac{1}{\beta}\log\int_z q(z|y)\frac{e^{-\beta E(y,z)}}{q(z|y)}$$

**Jensen's inequality: -log(average()) < average(-log())**

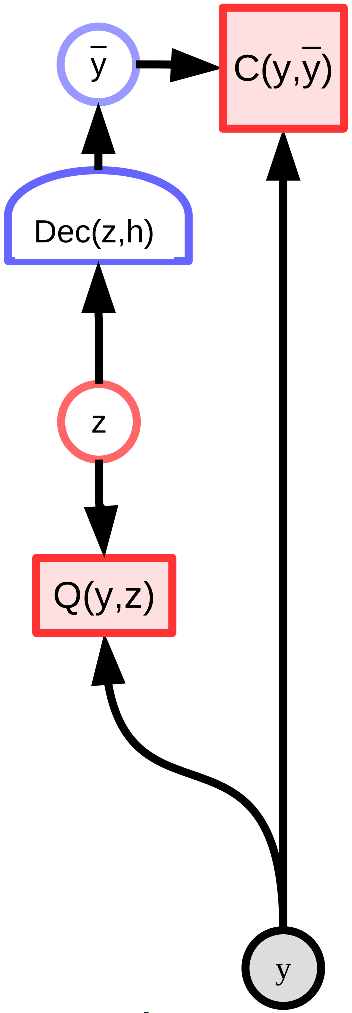
$$F(y) \leq \tilde{F}(y) = \int_z q(z|y)\left[-\frac{1}{\beta}\log\frac{e^{-\beta E(y,z)}}{q(z|y)}\right]$$

- **Variational free energy:**

$$\tilde{F}(y) = \int_z q(z|y)E(y,z) + \frac{1}{\beta}\int_z q(z|y)\log(q(z|y))$$

$$F = \langle E \rangle - TS$$

$\leftarrow$ **Helmholtz free energy in thermodynamics:**
**for a given average energy <E>, a system minimizes it free energy by maximizing its entropy S. The trade-off depends on the temperature T**



# Variational Auto-Encoder

- **If Q(y,z) is quadratic, q(z|y) is Gaussian.**

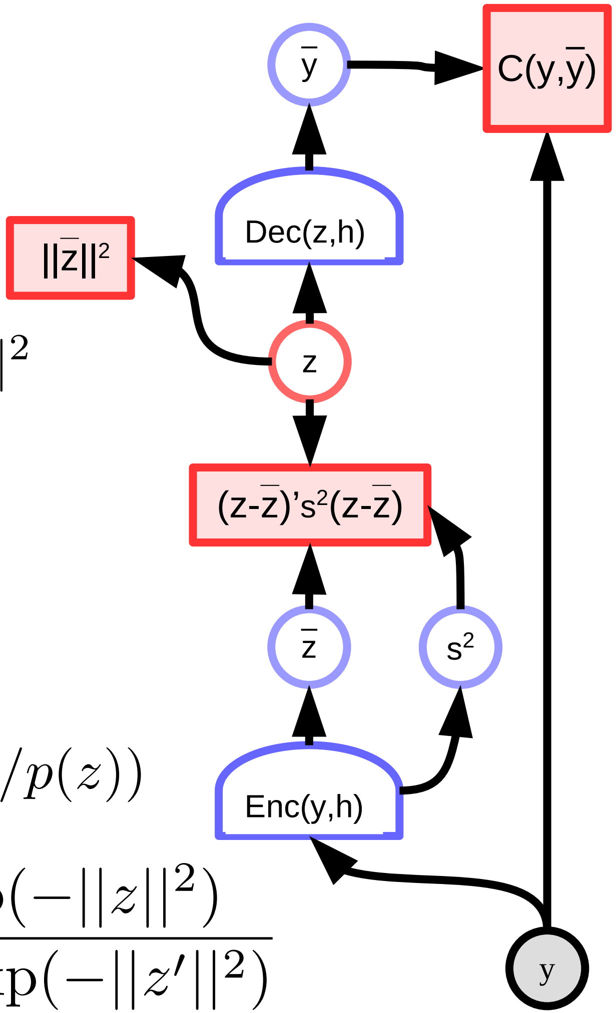
$$Q_{w_e}(y,z) = (z - \mathrm{Enc}_{w_e}(y))^T s^2 (z - \mathrm{Enc}_{w_e}(y)) + \gamma ||\mathrm{Enc}_{w_e}(y)||^2$$

$$q_{w_e}(z|y) = \frac{e^{-\beta Q_{w_e}(y,z)}}{\int_{z'} e^{-\beta Q_{w_e}(y,z')}}$$ **(Gaussian)**

$$\tilde{F}_w(y) = \int_z q_{w_e}(z|y) E_{w_d}(y,z) + \frac{1}{\beta} \int_z q_{w_e}(z|y) \log(q_{w_e}(z|y)/p(z))$$

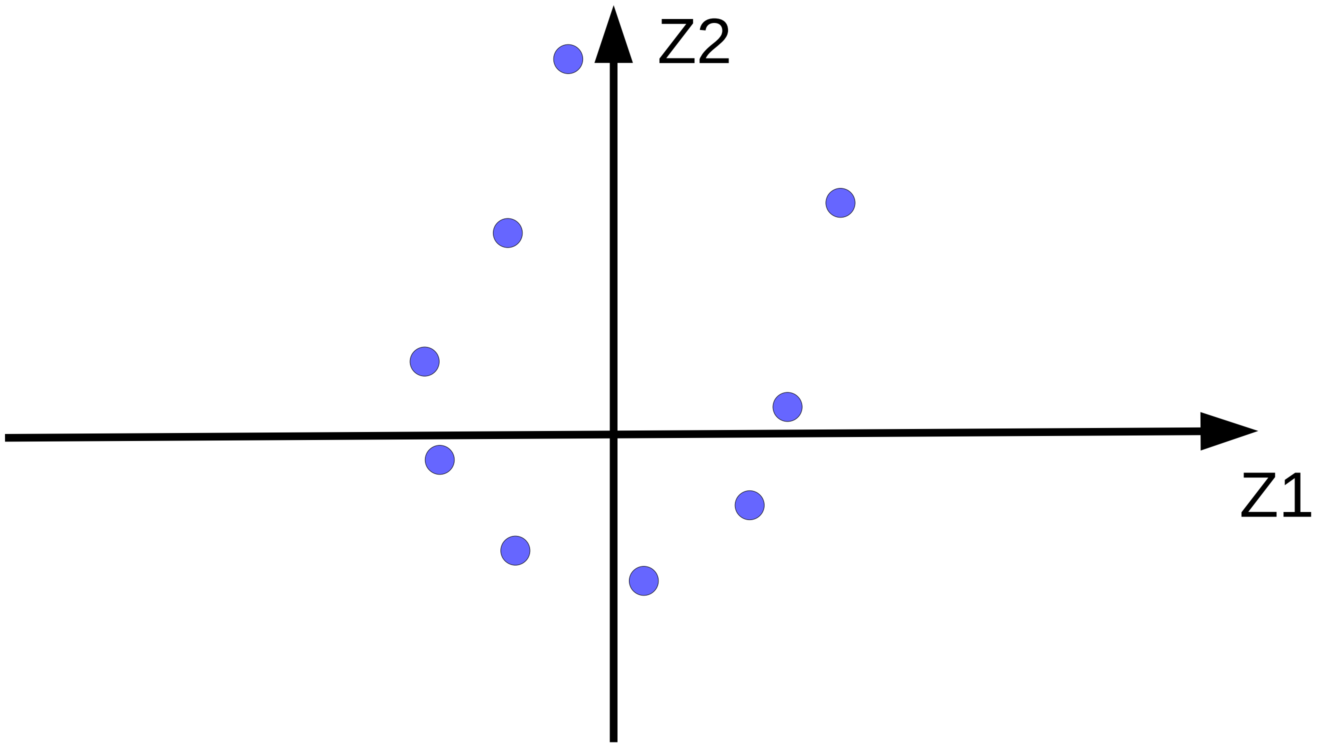
- **Loss** $\mathcal{L}(y,w) = \tilde{F}_w(y)$

$$p(z) = \frac{\exp(-||z||^2)}{\int_{z'} \exp(-||z'||^2)}$$



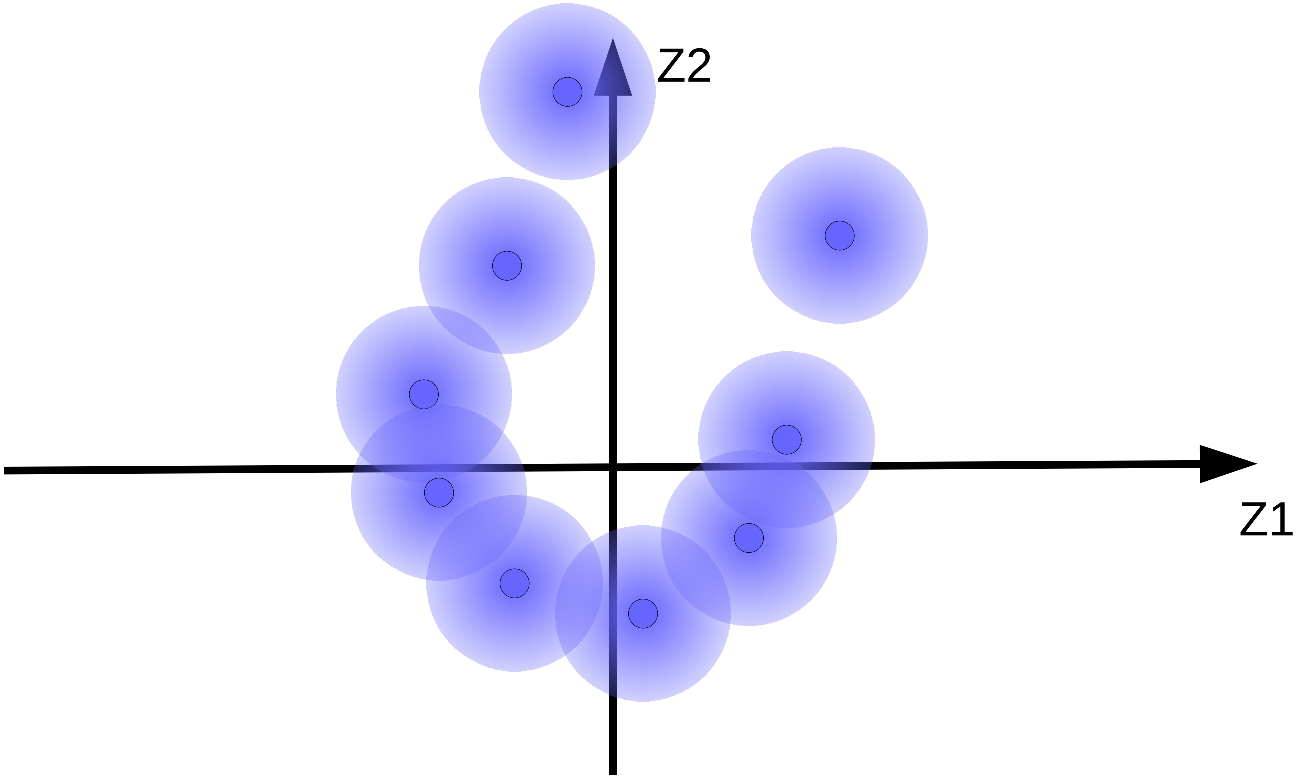
# Variational Auto-Encoder

- **Code vectors for training samples**



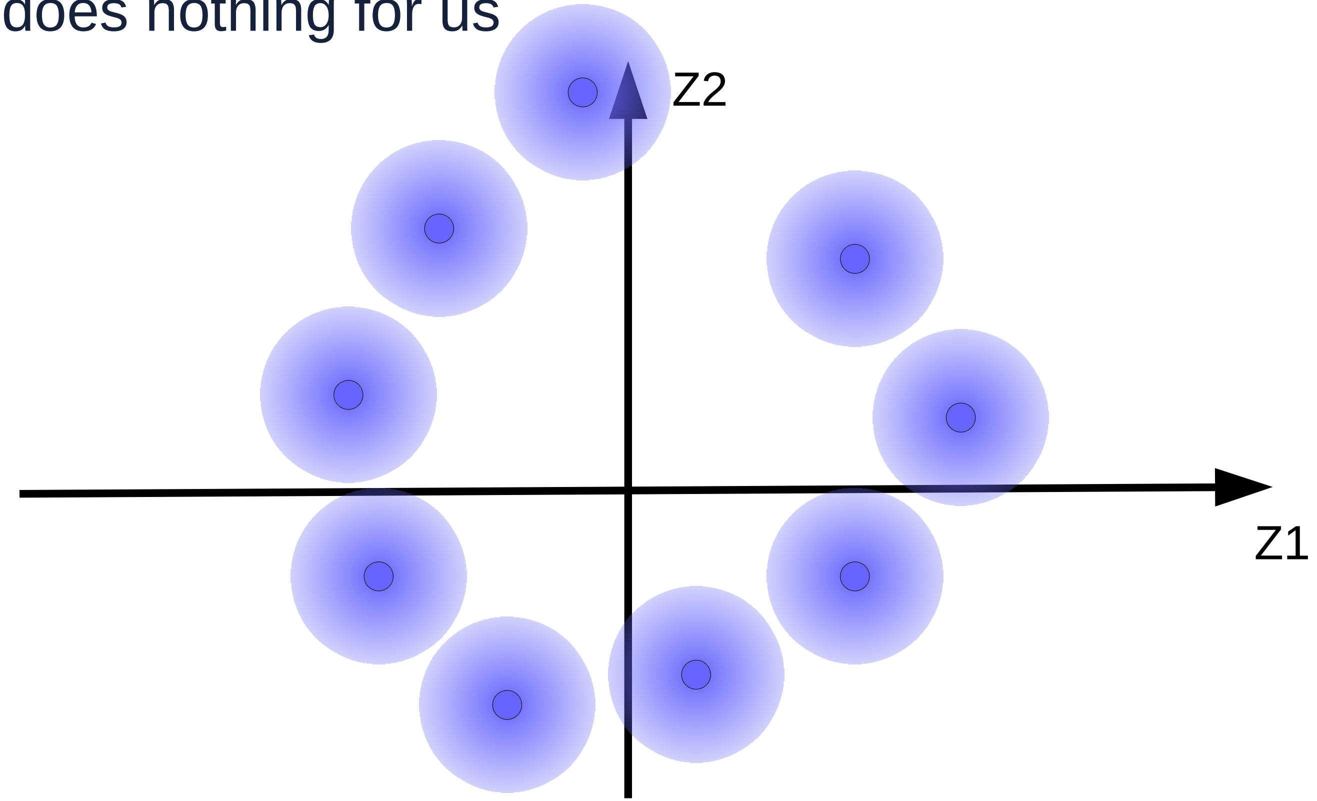
# Variational Auto-Encoder

- **Code vectors for training sample with Gaussian noise**
  - Some fuzzy balls overlap, causing bad reconstructions



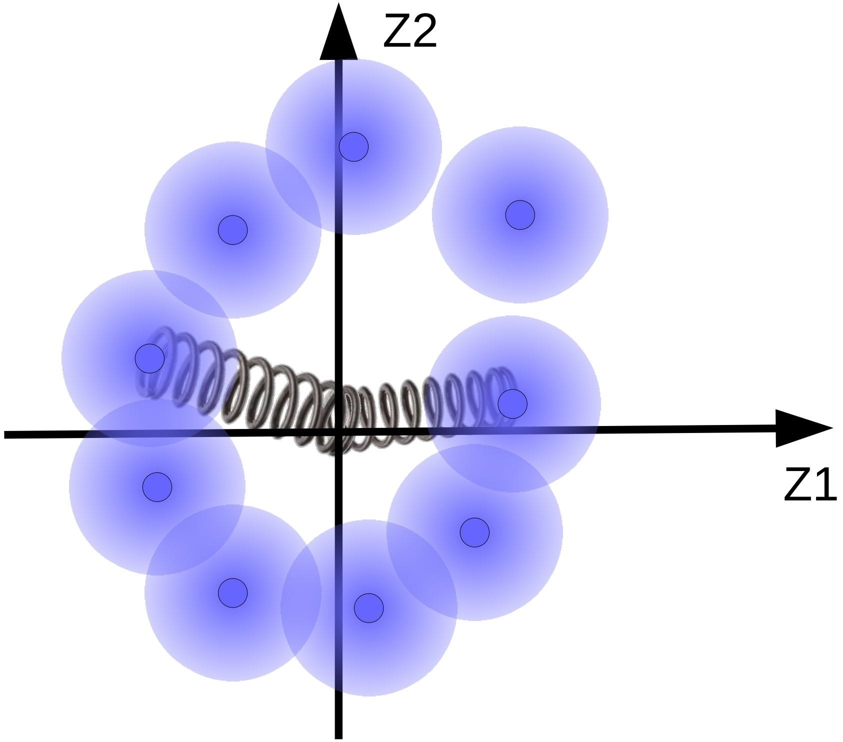
# Variational Auto-Encoder

- **The code vectors want to move away from each other to minimize reconstruction error**
  - But that does nothing for us

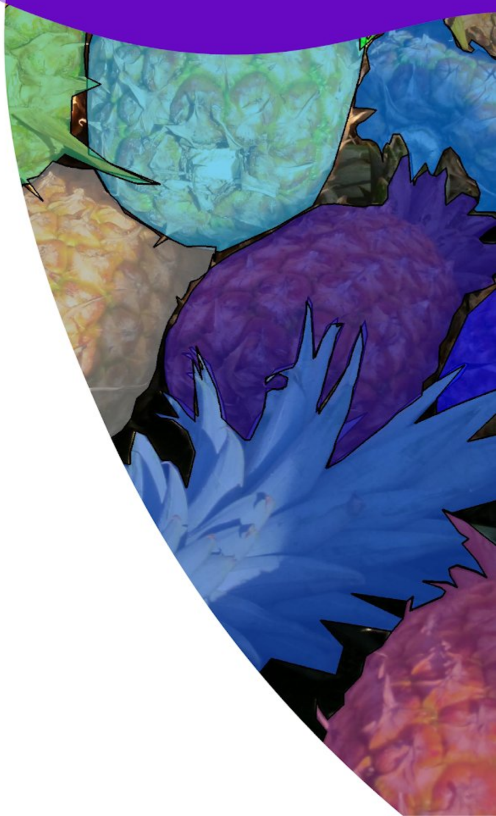


# Variational Auto-Encoder

- **Attach the balls to the center with a sping, so they don't fly away**
  - Minimize the square distances of the balls to the origin
- **Center the balls around the origin**
  - Make the center of mass zero
- **Make the sizes of the balls close to 1 in each dimension**
  - Through a so-called KL term



# Backprop as Lagrangian Optimization



Optimization under constraints

# Reformulating Deep Learning

- **Loss**

$$L(x, y, w) = C(z_K, y) \ \text{such that}\ z_{k+1} = g_k(z_k, w_k), \quad z_0 = x$$

- **Lagrangian for optimization under constraints**

$$L(x, y, z, \lambda, w) = C(z_K, y) + \sum_{k=0}^{K-1} \lambda_{k+1}^T (z_{k+1} - g_k(z_k, w_k))$$

- **Optimality conditions:**

$$\frac{\partial L(x, y, z, \lambda, w)}{\partial z_k} = 0, \quad \frac{\partial L(x, y, z, \lambda, w)}{\partial \lambda_{k+1}} = 0, \quad \frac{\partial L(x, y, z, \lambda, w)}{\partial w_k} = 0$$

# Reformulating Deep Learning

$$\frac{\partial L(x,y,z,\lambda,w)}{\partial \lambda_{k+1}} = z_{k+1} - g_k(z_k, w_k) = 0 \Longrightarrow z_{k+1} = g_k(z_k, w_k)$$

$$\frac{\partial L(x,y,z,\lambda,w)}{\partial z_k} = \lambda_k^T - \lambda_{k+1}^T \frac{\partial g_k(z_k, w_k)}{\partial z_k} = 0 \Longrightarrow$$

- **Backprop!**
  - Lambda is the gradient

$$\lambda_k = \frac{\partial g_{k-1}(z_{k-1}, w_{k-1})}{\partial z_k}^T \lambda_{k+1}$$

$$\frac{\partial L(x,y,z,\lambda,w)}{\partial w_k} = \lambda_{k+1}^T \frac{\partial g_k(z_k, w_k)}{\partial w_k}$$