CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Zoom Link

Recap

- Some pros of probabilistic approaches for classification are: ______. Some cons are ______.
- High dimensional generative models are based on l___ variables which capture h___ f___ of the outputs.
- Typically we have no idea what these could be, so we s____ them from a n___ distribution and warp them into the distribution required using a n___ n____.
- In high dimensions, most l___ v___ will not lead to t____ s___ e____ with high probability.
- So we learn a second function Q that attempts to produce p(___ | ___).
- The KL divergence between two distributions X and Y is defined as D(X,Y)=_____.

Today

- Generative Machine Learning
- Support Vector Machines

Evaluating Likelihood

$$p(X) = \int p(X \mid z)\, p(z)\, dz, z \sim N(0, I)$$

$$\approx \frac{1}{n} \sum_i p(X \mid z_i)$$

- Unfortunately, in high dimensions, most $p(X \mid z_i)$ will be near zero, so this is going to be VERY inefficient

Second key idea

- What if we had a function $Q(z \mid X)$, that could return a distribution over those z's that are likely to produce X?

- Then maybe we could use $E_{z \sim Q}\, p(X \mid z)$ to get a good approximation to the likelihood

Relationship between $E_{z\sim Q}\, p(X \mid z)$ and $p(X)$

$$D(Q(z \mid X) \,\|\, p(z \mid X))$$

$$= E_{z\sim Q}\left(\log\left(Q(z \mid X)\right) - \log\left(p(z \mid X)\right)\right)$$

$$= E_{z\sim Q}\left(\log\left(Q(z \mid X)\right) - \log\left(p(X \mid z)\right) - \log\left(p(z)\right)\right)$$

$$+\log\left(p(X)\right)$$

So

$$\log\left(p(X)\right) - D(Q(z \mid X) \,\|\, p(z \mid X)) =$$

$$E_{z\sim Q}\left(\log\left(p(X \mid z)\right)\right) - D\left(Q(z \mid X) \,\|\, p(z)\right)$$

Observations

- If we can find a good Q, the LHS $\approx p(X)$
- The RHS can be optimized via SGD! (w/suitable choices)
- The RHS is called an "encoder-decoder" architecture
  - Q is given X and is "encoding" it into z
  - p (through the unknown f introduced before) will take z and "decode" it into X

Optimizing the RHS

- What to choose for $Q(z \mid X)$?
- Suppose we set

$$Q(z \mid X) = N(\mu_\varphi(X), \Sigma_\varphi(X))$$

  - In this case, this will be a single ANN φ that takes X as input and outputs μ and Σ
- With this choice, the second term on the RHS can be computed in closed form

Second term

$$D\left(Q(z \mid X) \,\|\, p(z)\right) =$$

$$\frac{1}{2}\left[tr(\Sigma(X)) + \mu(X)^T \mu(X) - k - \log\left(\det\left(\Sigma(X)\right)\right)\right]$$

Trace

Dimensionality of z

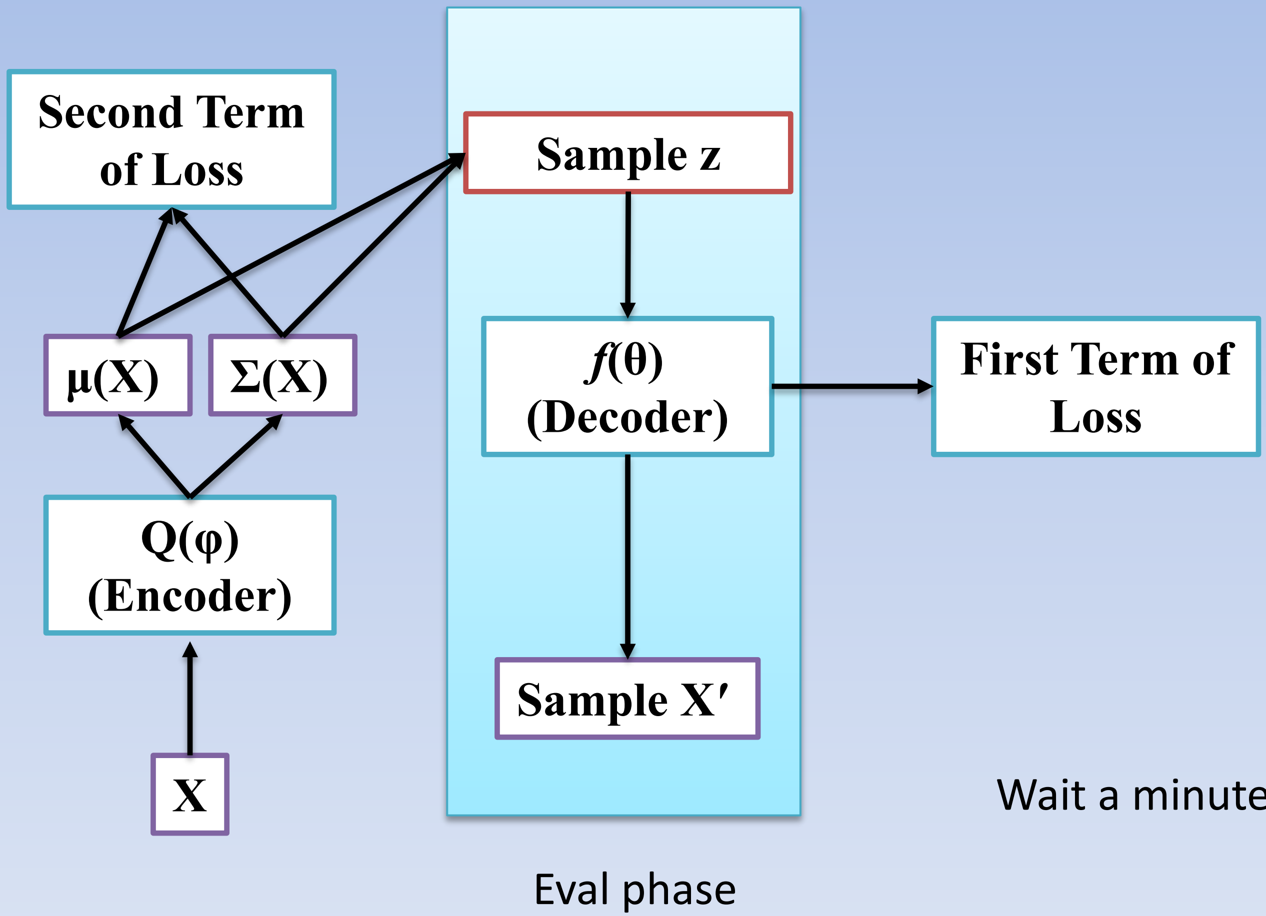
Determinant

First term

$$E_{z\sim Q}\left(\log\left(p(X\mid z)\right)\right)??$$

- Do we need to sample many times? That would be a problem…
- Third clever idea: *One* z sample can be enough!!
  - Why? When we do SGD, every time we run through an example x_i, we will resample z_i, so in the limit of enough epochs the stochastic gradient should converge to the true gradient under expectation!

The Variational Auto-Encoder (VAE) architecture

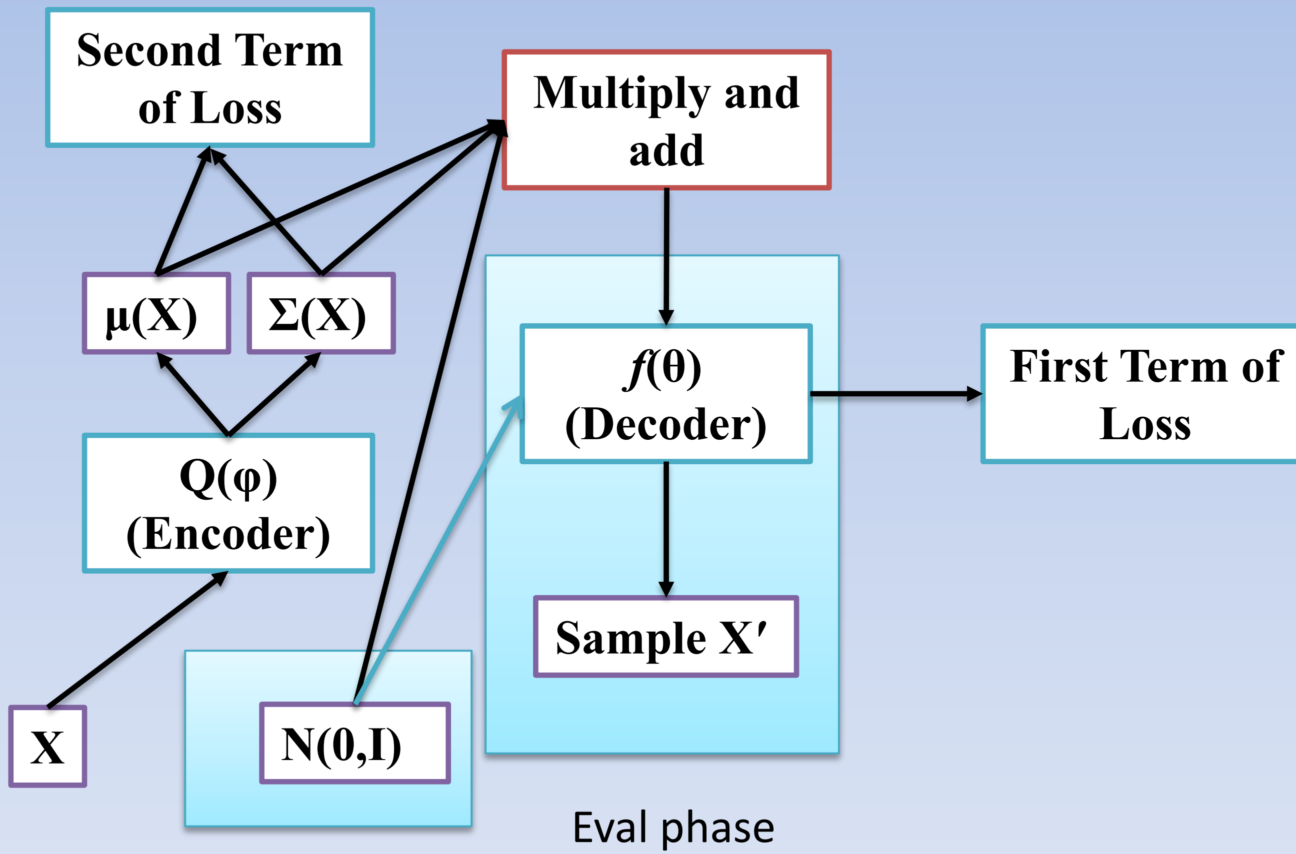


Wait a minute…

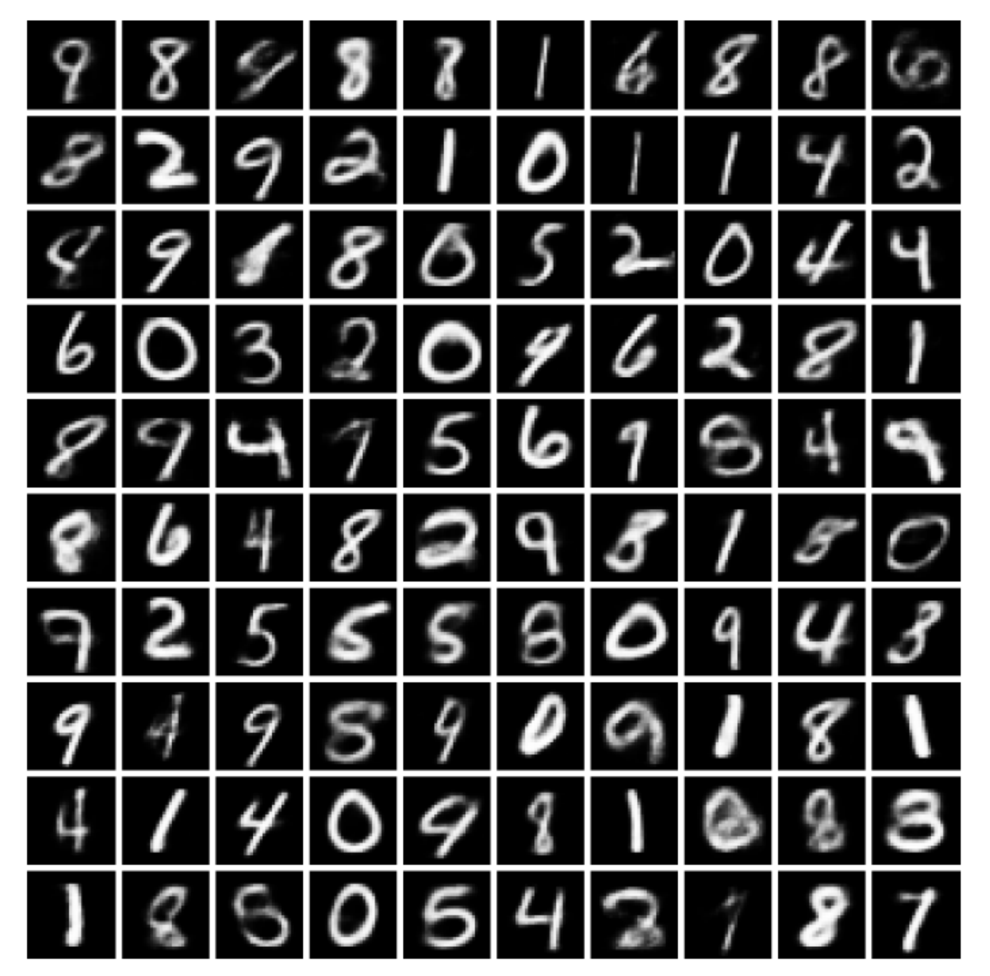
The final clever idea: the "Reparameterization Trick"

- We cannot backpropagate the loss through the single sample z!
  - So the first term never affects the encoder, which will never learn good choices for z for each X
- So instead, *we move the sampling to the input layer by sampling* $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{I})$
- We can do this because for a Gaussian $N(\mu(X), \Sigma(X)) = \mu(X) + \Sigma^{1/2}(X)\boldsymbol{\varepsilon}$

The Variational Auto-Encoder (VAE) architecture



Example Output: MNIST



Improvements

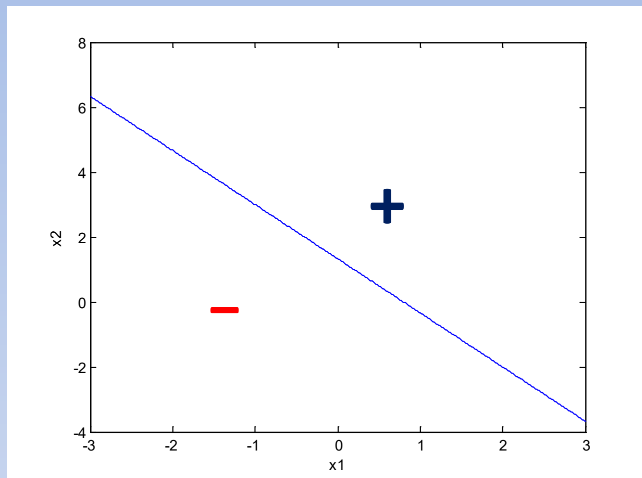
- Many subsequent modifications
  - Conditional VAE, to condition the VAE on known evidence/labels
  - Generative Adversarial Networks (GANs)
    - Combine a generative model with a "discriminator" to enable very high dimensional sampling
    - Many interesting questions emerge, see Robbie Dozier's 2022 MS Thesis
  - Diffusion Models
    - Producing a single Gaussian distribution over $X|z$ in one step can be hard
    - What if we did this in multiple steps, each step a *perturbation* of the previous?

Support Vector Machines

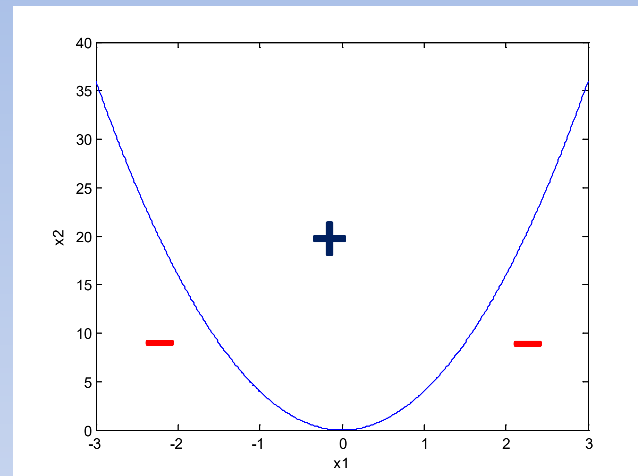
- Combines three fundamental ideas
  - Linear discriminants
  - Margins
  - Kernels

- A theoretically well justified and empirically well-behaved method arising from three fields: ML (Cortes & Vapnik), Statistics (Wahba), Operations Research (Mangasarian)

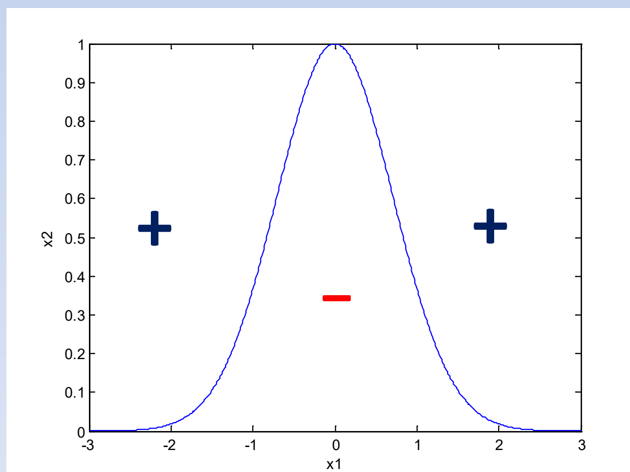
What is a “linear discriminant”?



$sign(5x_1 + 3x_2 - 4)$



$sign(x_2 - 4x_1^2)$



$sign(x_2 - e^{-x_1^2})$

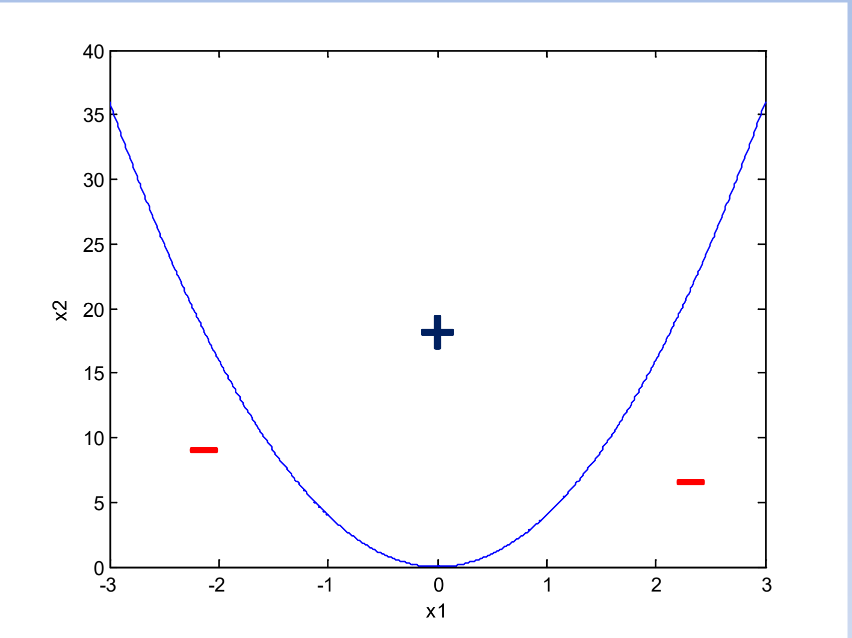
Linear Discriminants

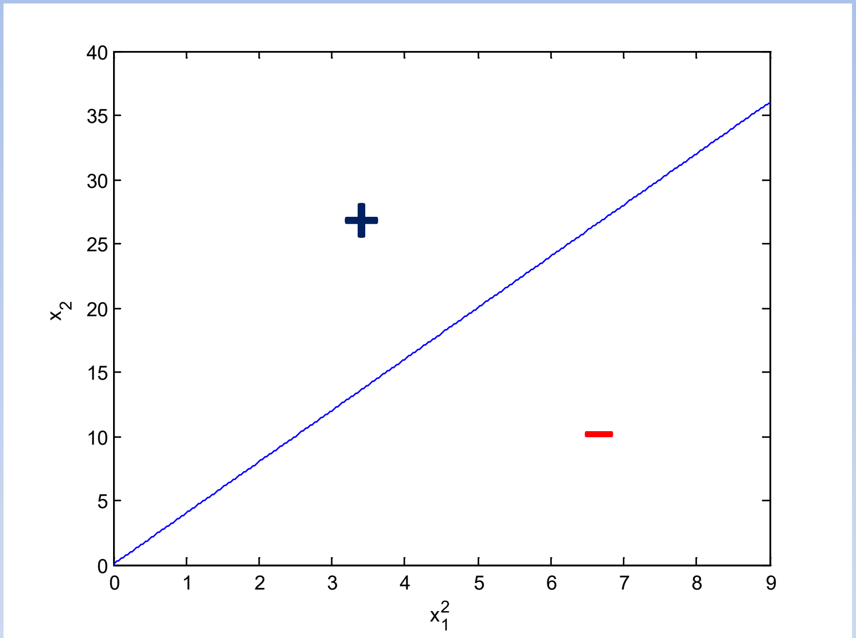
- We generally take "linear" to mean linear in the classifier parameters
  - Linear in $\mathbf{w}$, but not necessarily in $\mathbf{x}$
- A linear discriminant has the general form

$$\mathbf{w}\bullet\varphi(\mathbf{x})+b=0$$

- Here $\boldsymbol{\varphi}$ ("feature map") is any vector function from the domain of $\mathbf{x}$ to R^m
  - $\mathbf{x}$ need not be a number
  - φ could be arbitrary-dimensional

Linear Discriminants





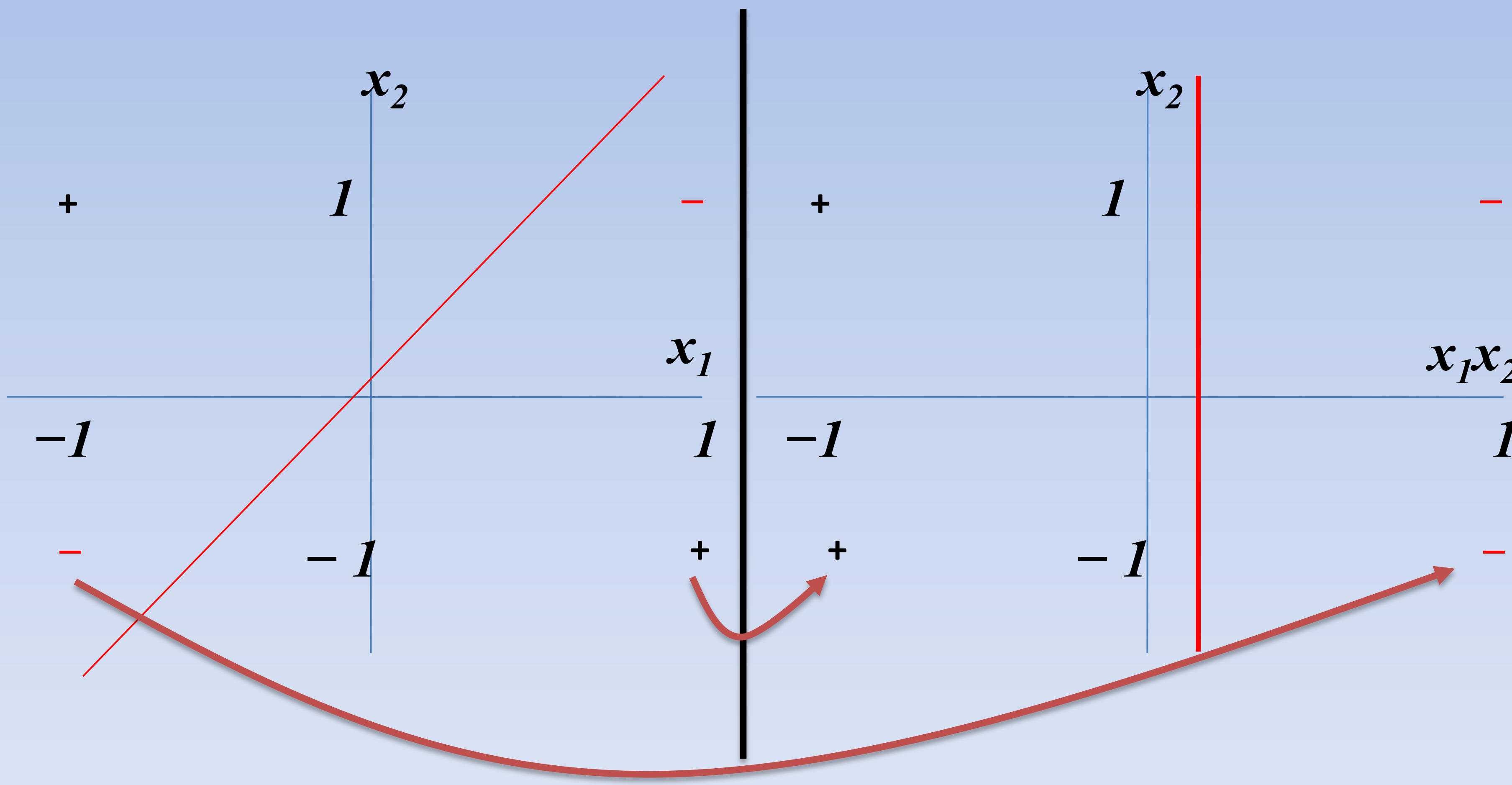
$$sign(x_2 - 4x_1^2)$$

$$\varphi(\mathbf{x}) = (x_1^2, x_2)$$

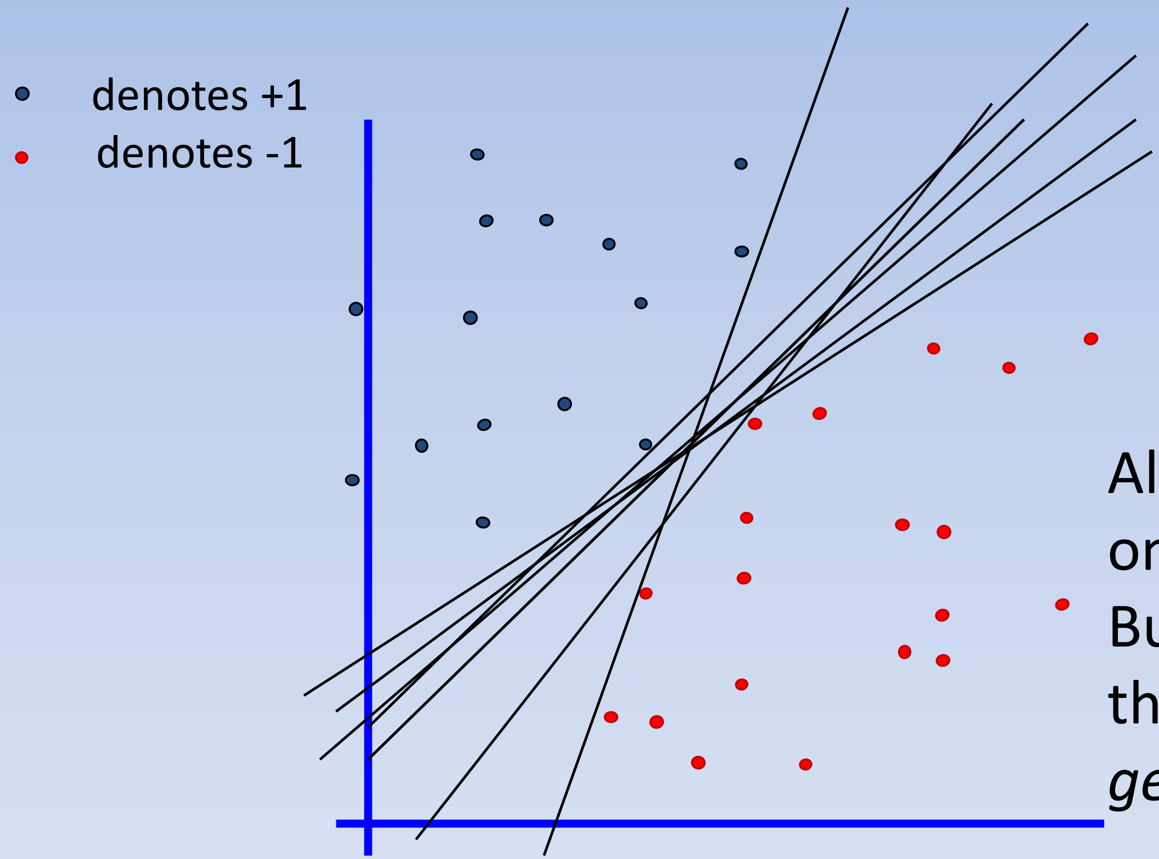
$$sign(\varphi_2(\mathbf{x}) - 4\varphi_1(\mathbf{x}))$$

φ maps features to an m-dimensional vector space

XOR and the Linear Discriminant



Find the Classifier



All are equally good on the training sample. But is there any one that we expect to *generalize* best?