Lecture 8

Neural Nets and Information Theory

Last Times:

- Machine Learning
- SGD for minimizing a loss
- regularization
- logistic log-loss
- Thinking in units, autodiff

Today

- BackPropagation
- Neural Nets and Universal Approximation
- KL-Divergence, entropy and cross-entropy
- maximum entropy distributions
- deviance
- AIC

What did we learn about learning?

- x-validation: minimizes loss on training, fits hyperparams on validation
- test risk approximates out-of-sample risk
- regularization or complexity selection helps avoid overfitting
- we have seen the context of supervised learning $p(y|x)$

In unsupervised learning, want $p(x)$. Also need to learn these params using MLE or similar.

Something more: Scoring or Decision Loss

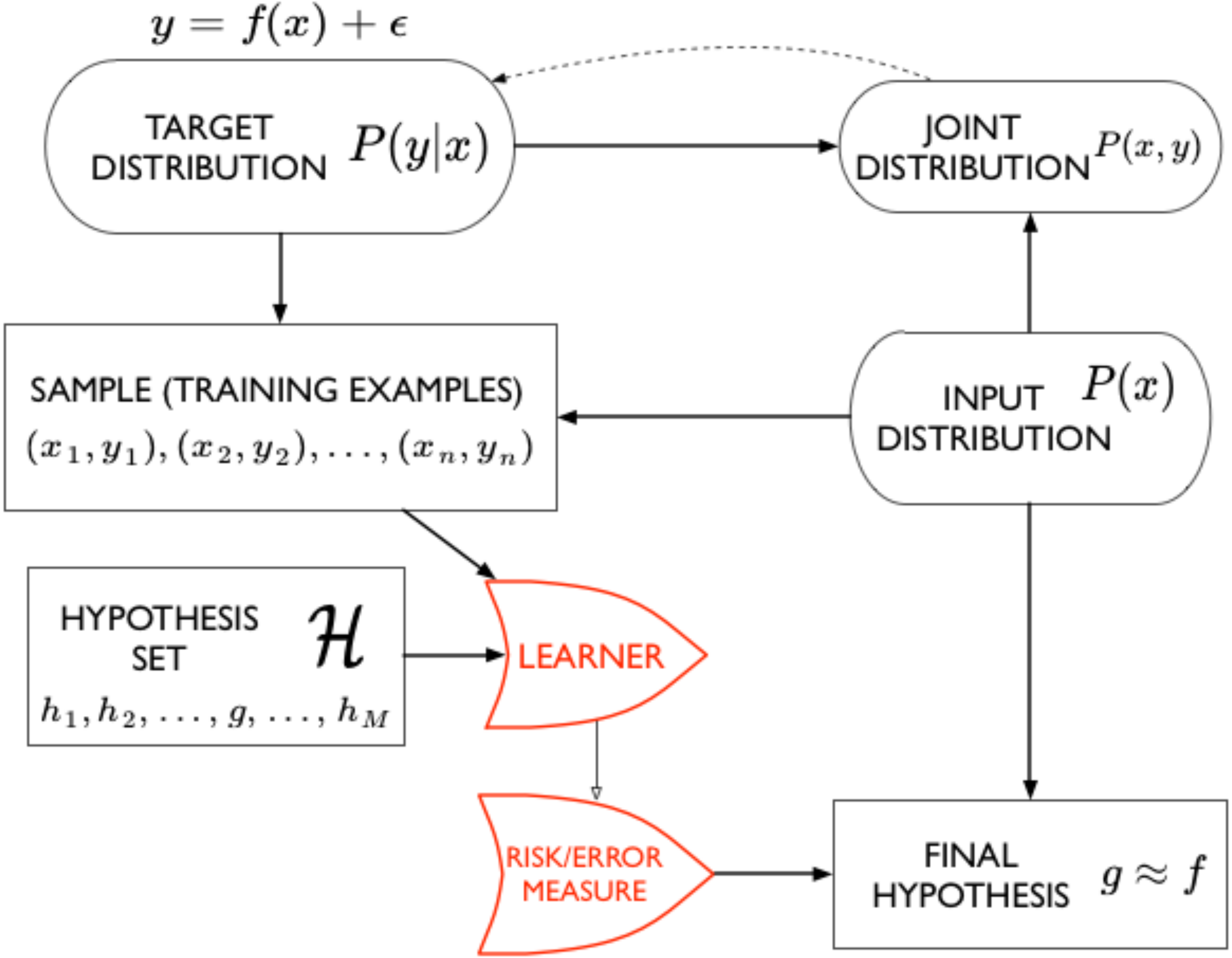
- we train using log-loss for example
- but we can score using a different loss function, example, 1-0 loss (which is not convex)
- this **Decision Loss** depends on the problem at hand
- we will come back to this

Logistic Regression

Define $h(z) = \dfrac{1}{1 + e^{-z}}$.

Then, the conditional probabilities of $y = 1$ or $y = 0$ given a particular sample's features $\mathbf{x}$ are:

$$
\begin{aligned}
P(y = 1|\mathbf{x}) &= h(\mathbf{w} \cdot \mathbf{x}) \\
P(y = 0|\mathbf{x}) &= 1 - h(\mathbf{w} \cdot \mathbf{x}).
\end{aligned}
$$

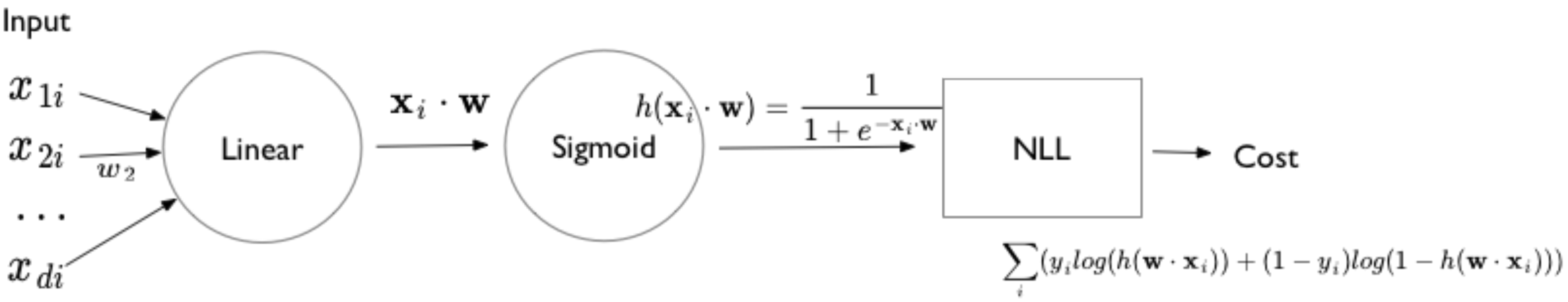


$y = f(x) + \epsilon$
TARGET DISTRIBUTION $P(y|x)$
JOINT DISTRIBUTION $P(x, y)$
SAMPLE (TRAINING EXAMPLES)
$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
INPUT DISTRIBUTION $P(x)$
HYPOTHESIS SET $\mathcal{H}$
$h_1, h_2, \ldots, g, \ldots, h_M$
LEARNER
RISK/ERROR MEASURE
FINAL HYPOTHESIS $g \approx f$

$$P(y|\mathbf{x},\mathbf{w}) = P(\{y_i\}|\{\mathbf{x}_i\},\mathbf{w}) = \prod_{y_i \in \mathcal{D}} P(y_i|\mathbf{x_i},\mathbf{w}) = \prod_{y_i \in \mathcal{D}} h(\mathbf{w}\cdot\mathbf{x_i})^{y_i}(1-h(\mathbf{w}\cdot\mathbf{x_i}))^{(1-y_i)}$$

$$\begin{aligned}
\ell &= log\left(\prod_{y_i \in \mathcal{D}} h(\mathbf{w}\cdot\mathbf{x_i})^{y_i}(1-h(\mathbf{w}\cdot\mathbf{x_i}))^{(1-y_i)}\right) \\
&= \sum_{y_i \in \mathcal{D}} log\left(h(\mathbf{w}\cdot\mathbf{x_i})^{y_i}(1-h(\mathbf{w}\cdot\mathbf{x_i}))^{(1-y_i)}\right) \\
&= \sum_{y_i \in \mathcal{D}} log\, h(\mathbf{w}\cdot\mathbf{x_i})^{y_i} + log\,(1-h(\mathbf{w}\cdot\mathbf{x_i}))^{(1-y_i)} \\
&= \sum_{y_i \in \mathcal{D}} (y_i log(h(\mathbf{w}\cdot\mathbf{x})) + (1-y_i)log(1-h(\mathbf{w}\cdot\mathbf{x})))
\end{aligned}$$

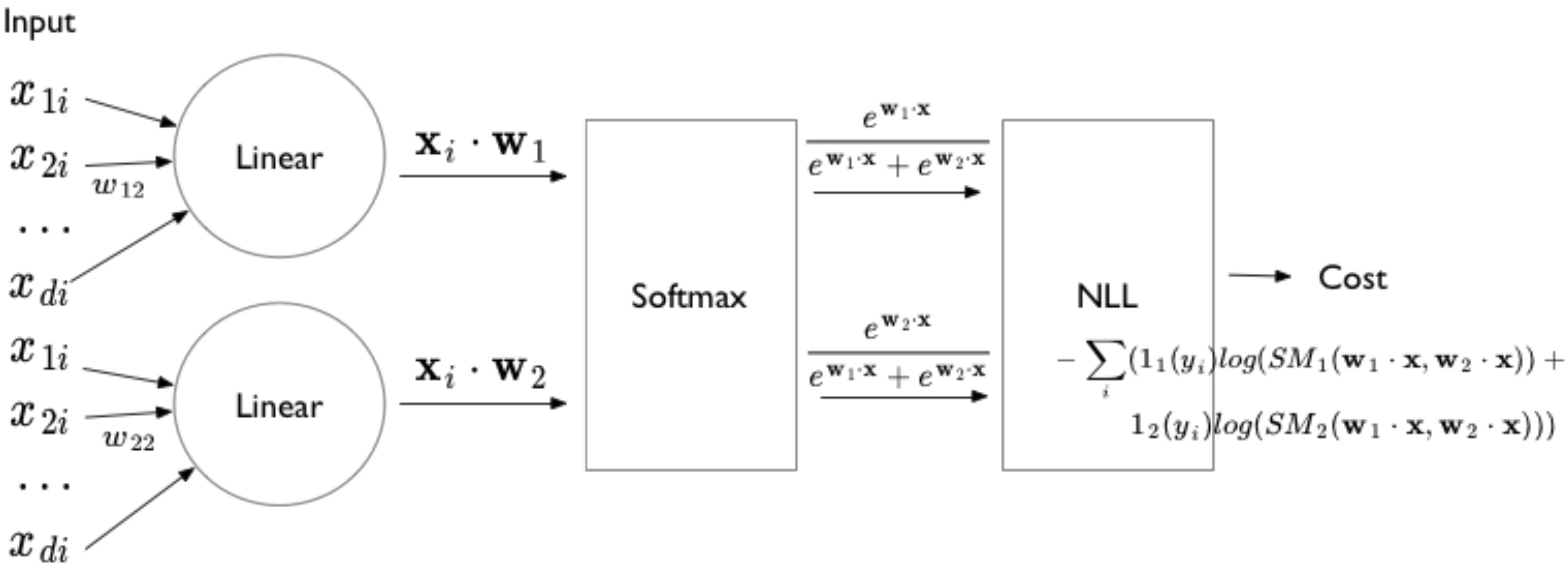
Units based diagram



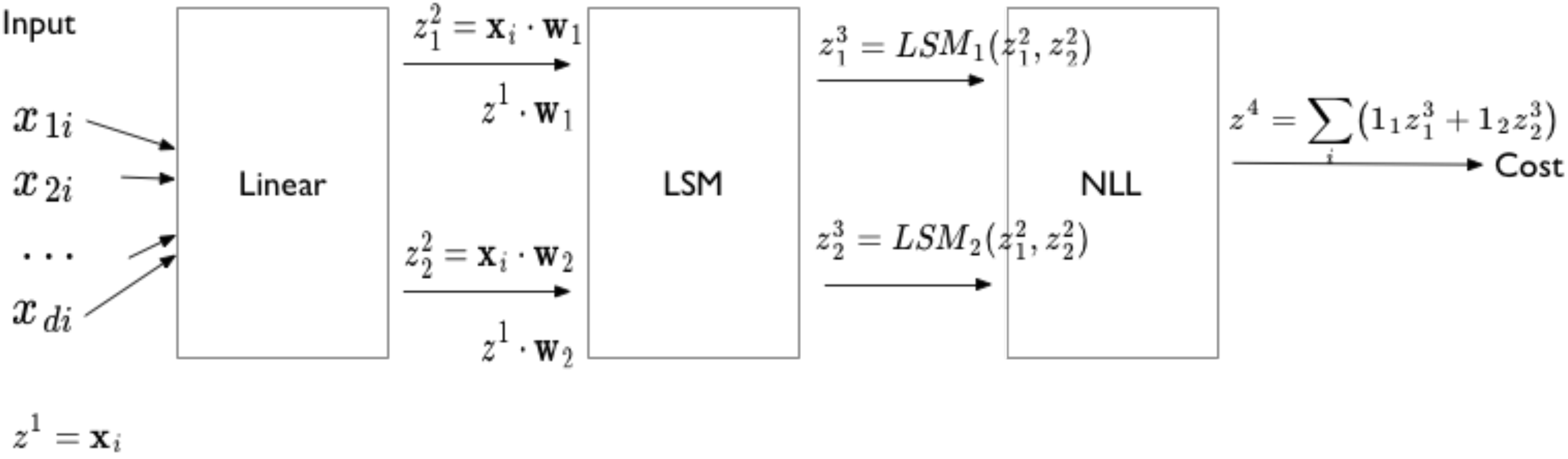
Softmax formulation

- Identify p_i and $1 - p_i$ as two separate probabilities constrained to add to 1. That is $p_{1i} = p_i ; p_{2i} = 1 - p_i$.
- $p_{1i} = \dfrac{e^{\mathbf{w}_1 \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}}}$
- $p_{2i} = \dfrac{e^{\mathbf{w}_2 \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}}}$
- Can translate coefficients by fixed amount ψ without any change

Units diagram for Softmax



Units diagram Again



Equations, layer by layer

$$\mathbf{z}^1 = \mathbf{x}_i$$

$$\mathbf{z}^2 = (z_1^2, z_2^2) = \left(\mathbf{w}_1 \cdot \mathbf{x}_i, \mathbf{w}_2 \cdot \mathbf{x}_i\right) = \left(\mathbf{w}_1 \cdot \mathbf{z}_i^1, \mathbf{w}_2 \cdot \mathbf{z}_i^1\right)$$

$$\mathbf{z}^3 = (z_1^3, z_2^3) = \left(LSM_1(z_1^2, z_2^2), LSM_2(z_1^2, z_2^2)\right)$$

$$z^4 = NLL(\mathbf{z}^3) = NLL(z_1^3, z_2^3) = -\sum_i \left(1_1(y_i)z_1^3(i) + 1_2(y_i)z_1^3(i)\right)$$

Reverse Mode Differentiation

$$Cost = f^{Loss}(\mathbf{f}^3(\mathbf{f}^2(\mathbf{f}^1(\mathbf{x}))))$$

$$\nabla_{\mathbf{x}} Cost = \frac{\partial f^{Loss}}{\partial \mathbf{f}^3} \frac{\partial \mathbf{f}^3}{\partial \mathbf{f}^2} \frac{\partial \mathbf{f}^2}{\partial \mathbf{f}^1} \frac{\partial \mathbf{f}^1}{\partial \mathbf{x}}$$

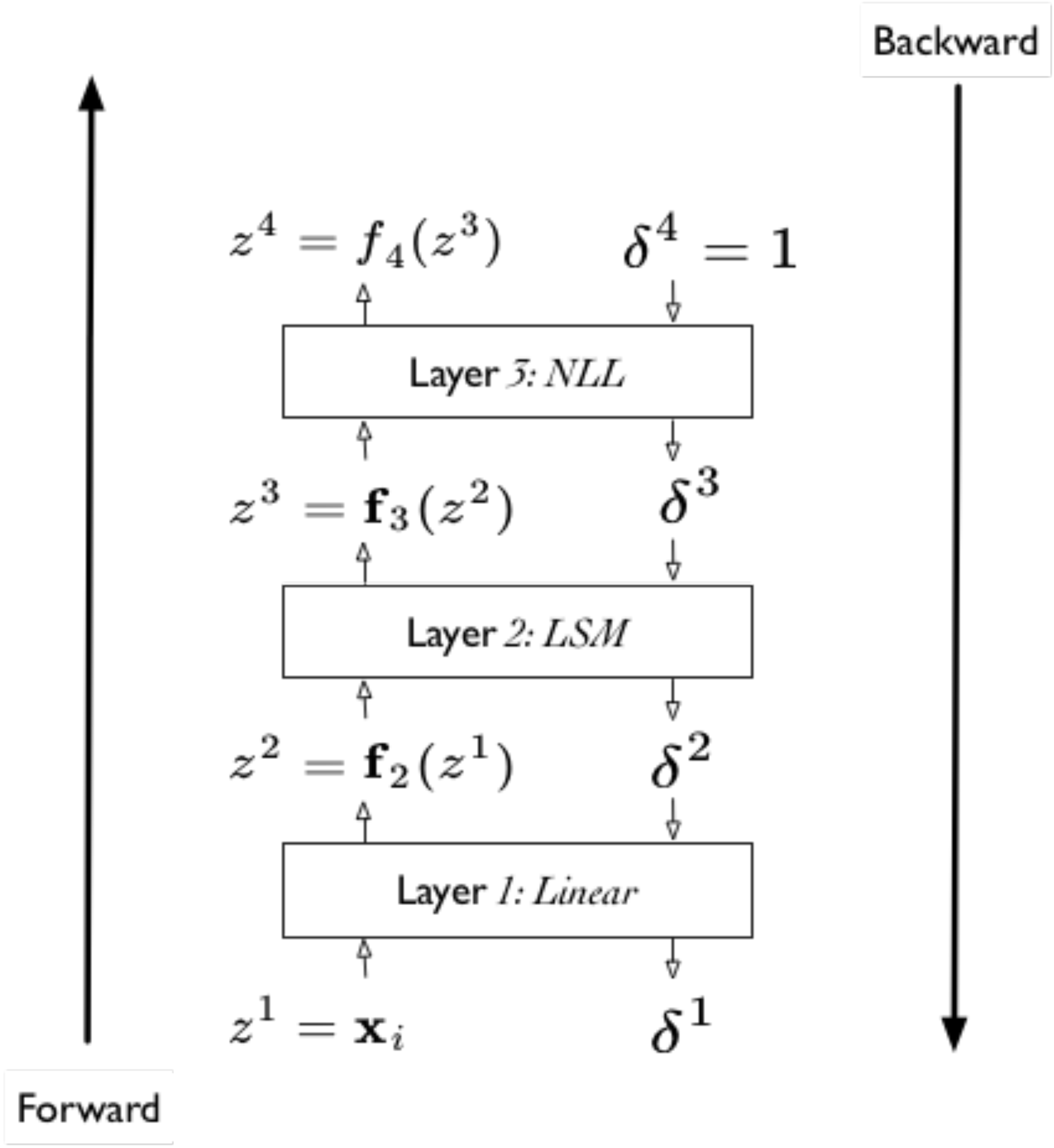
Write as:

$$\nabla_{\mathbf{x}} Cost = (((\frac{\partial f^{Loss}}{\partial \mathbf{f}^3} \frac{\partial \mathbf{f}^3}{\partial \mathbf{f}^2}) \frac{\partial \mathbf{f}^2}{\partial \mathbf{f}^1}) \frac{\partial \mathbf{f}^1}{\partial \mathbf{x}})$$

From Reverse Mode to Back Propagation

- Recursive Structure
- Always a vector times a Jacobian
- We add a "cost layer" to z^4. The derivative of this layer with respect to z^4 will always be 1.
- We then propagate this derivative back.

Layer Cake



Backward
$z^4 = f_4(z^3)$
$\delta^4 = 1$
Layer 3: *NLL*
$z^3 = \mathbf{f}_3(z^2)$
δ^3
Layer 2: *LSM*
$z^2 = \mathbf{f}_2(z^1)$
δ^2
Layer 1: *Linear*
$z^1 = \mathbf{x}_i$
δ^1
Forward

Backpropagation

RULE1: FORWARD (`.forward` in pytorch) $\mathbf{z}^{l+1} = \mathbf{f}^l(\mathbf{z}^l)$

RULE2: BACKWARD (`.backward` in pytorch)

$\delta^l = \frac{\partial C}{\partial \mathbf{z}^l}$ or $\delta^l_u = \frac{\partial C}{\partial z^l_u}$.

$$\delta^l_u = \frac{\partial C}{\partial z^l_u} = \sum_v \frac{\partial C}{\partial z^{l+1}_v} \frac{\partial z^{l+1}_v}{\partial z^l_u} = \sum_v \delta^{l+1}_v \frac{\partial z^{l+1}_v}{\partial z^l_u}$$

In particular:

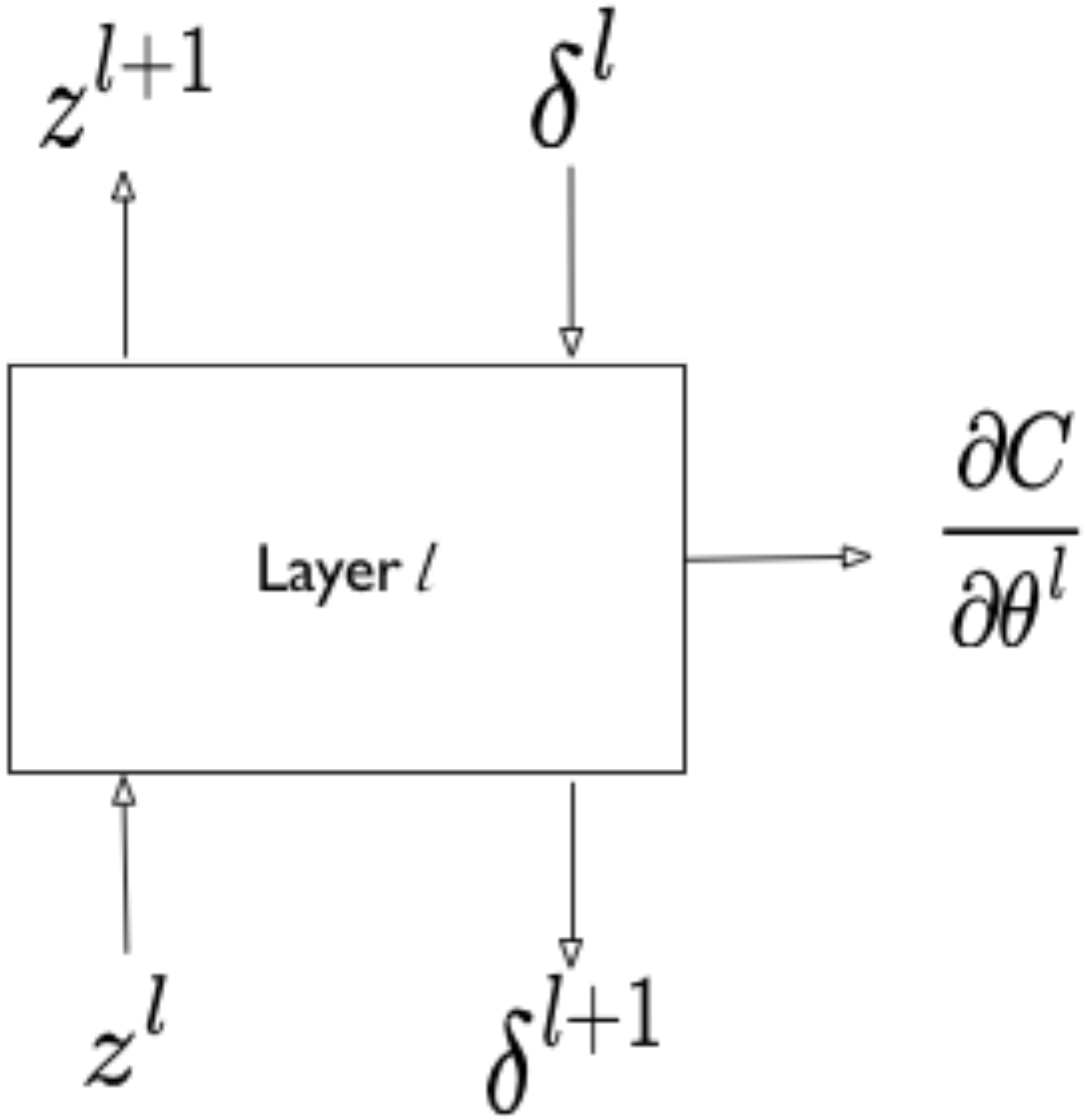
$$\delta_u^3 = \frac{\partial z^4}{\partial z_u^3} = \frac{\partial C}{\partial z_u^3}$$

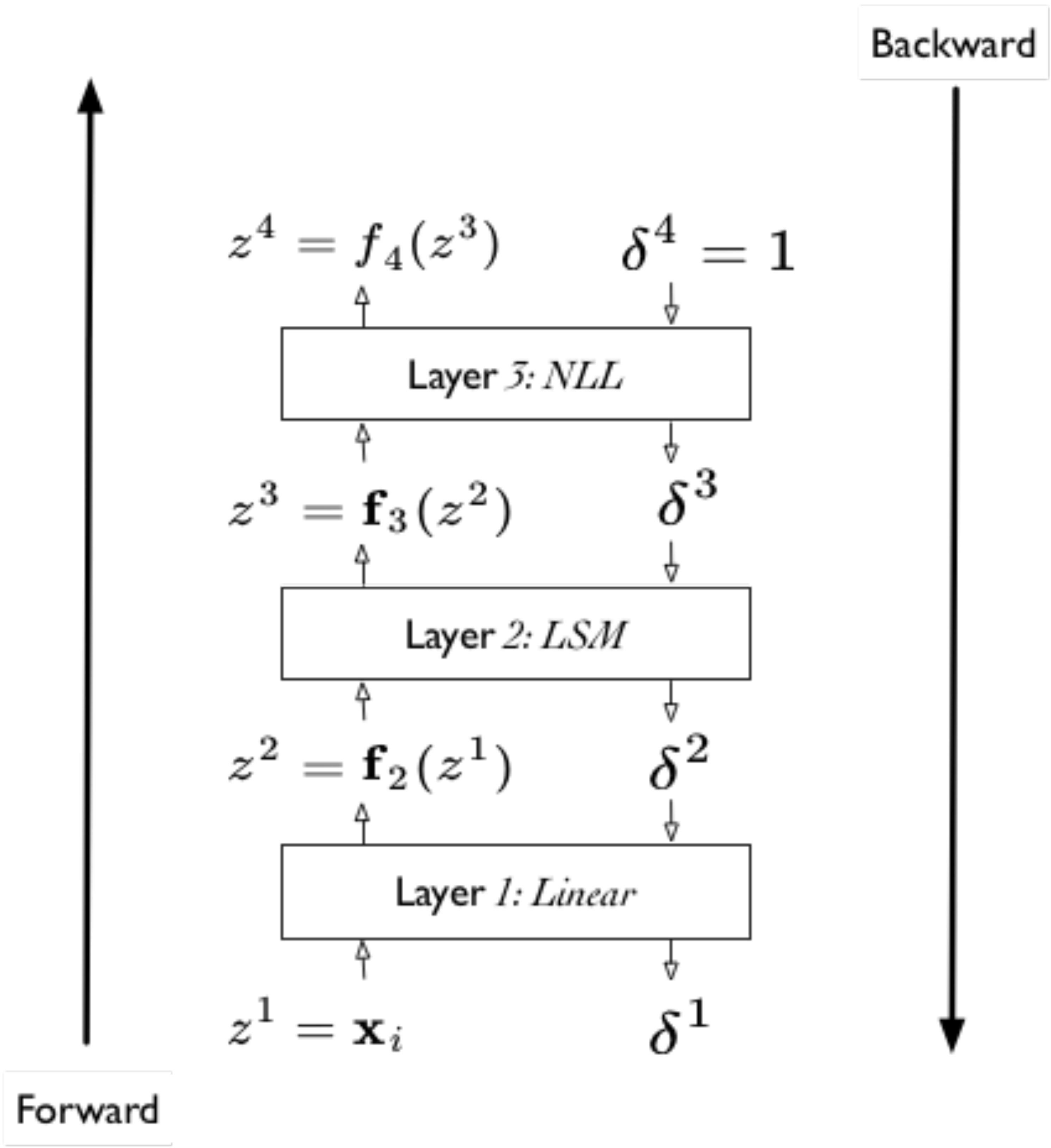
RULE 3: PARAMETERS

$$\frac{\partial C}{\partial \theta^l} = \sum_u \frac{\partial C}{\partial z_u^{l+1}} \frac{\partial z_u^{l+1}}{\partial \theta^l} = \sum_u \delta_u^{l+1} \frac{\partial z_u^{l+1}}{\partial \theta^l}$$

(backward pass is thus also used to fill the `variable.grad` parts of parameters in pytorch)

THATS IT! Write your Own Layer



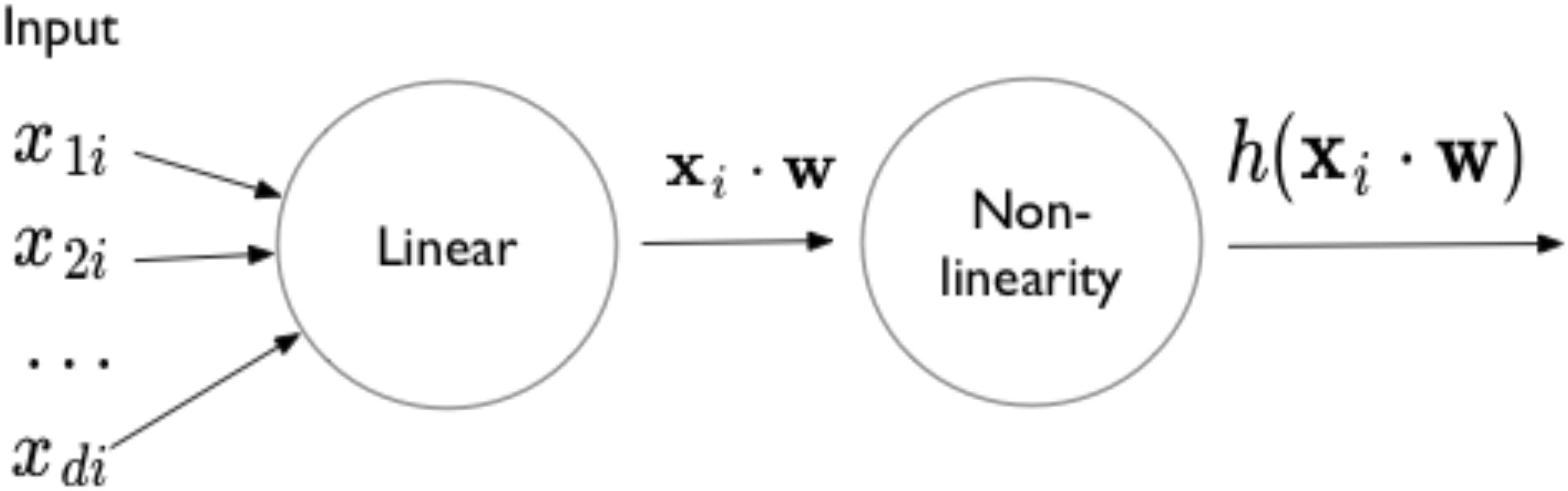

Backward
$z^4 = f_4(z^3)$
$\delta^4 = 1$
Layer 3: NLL
$z^3 = \mathbf{f}_3(z^2)$
δ^3
Layer 2: LSM
$z^2 = \mathbf{f}_2(z^1)$
δ^2
Layer 1: Linear
$z^1 = \mathbf{x}_i$
δ^1
Forward

What it looks like?

See https://github.com/joelgrus/joelnet

Look at the video. A full deep learning library in 35 minutes!

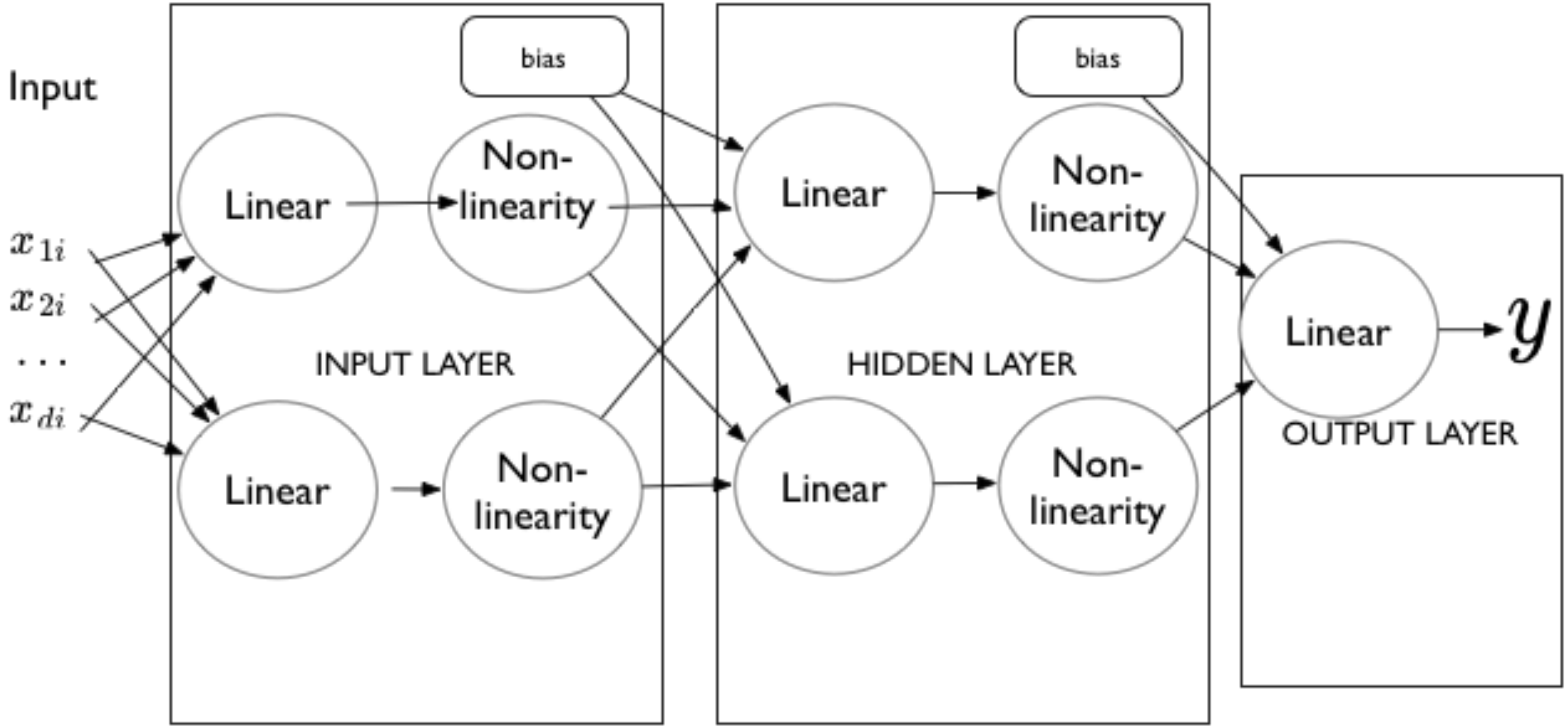
Neural Nets: The perceptron



Just combine perceptrons

- both deep and wide
- this buys us complex nonlinearity
- both for regression and classification
- key technical advance: BackPropagation with
- autodiff
- key technical advance: gpu

Combine Perceptrons



Universal Approximation

- any one hidden layer net can approximate any continuous function with finite support, with appropriate choice of nonlinearity
- under appropriate conditions, all of sigmoid, tanh, RELU can work
- but may need lots of units
- and will learn the function it thinks the data has, not what you think

KL-Divergence

$$D_{KL}(p, q) = E_p[log(p) - log(q)] = E_p[log(p/q)]$$
$$= \sum_i p_i log(\frac{p_i}{q_i}) \ or \int dPlog(\frac{p}{q})$$

$$D_{KL}(p, p) = 0$$

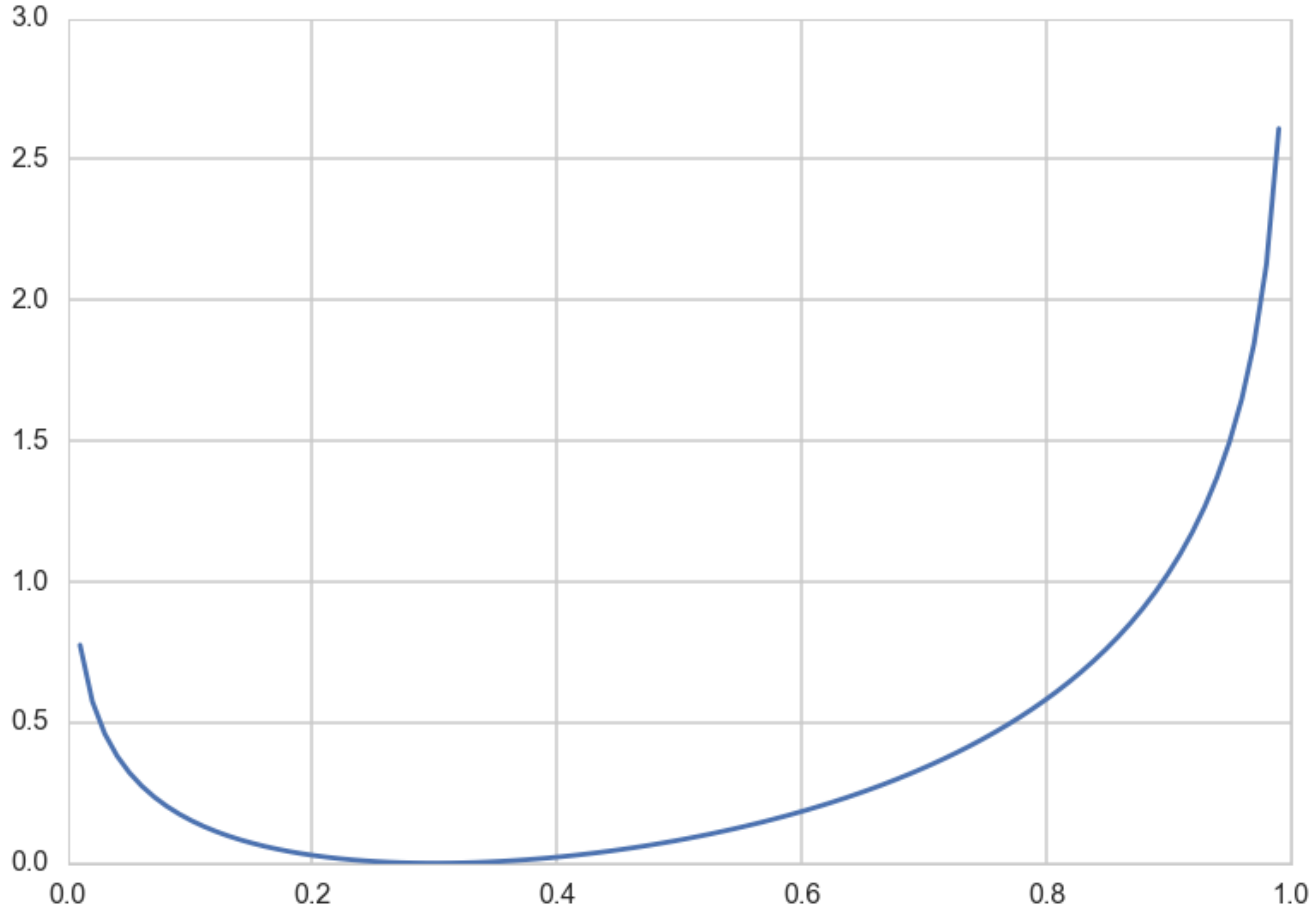
KL divergence measures distance/dissimilarity of the two distributions p(x) and q(x).

KL example

Bernoulli Distribution p with $p = 0.3$.

Try toapproximate by q. What parameter?

```
def kld(p,q):
    return p*np.log(p/q) + (1-p)*np.log((1-p)/(1-q))
```



KL-Divergence is always non-negative

Jensen's inequality: given a convex function $f(x)$:

$$E[f(X)] \geq f(E[X])$$

$$\implies D_{KL}(p,q) \geq 0 \text{ (0 iff } q = p \,\forall x\text{).}$$

$$D_{KL}(p,q) = E_p[log(p/q)] = E_p[-log(q/p)] \geq -\log(E_p[q/p]) = -\log(\int dQ) = 0$$

PROBLEM: we dont know distribution p. If we did, why do inference?

SOLUTION: Use the empirical distribution

That is, approximate population expectations by sample averages.

So, $E_p[f] \simeq \frac{1}{N} \sum_{iin \mathcal{D}_{train}} f(x_i)$. Go back and see Logistic regression!

Maximum Likelihood justification

$$D_{KL}(p,q) = E_p[log(p/q)] = \frac{1}{N}\sum_i (log(p_i) - log(q_i))$$

Minimizing KL-divergence $\implies$ maximizing $\sum_i log(q_i)$

Which is exactly the log likelihood! MLE!

Information and Uncertainty

- coin at 50% odds has maximal uncertainty
- reflects my lack of knowledge of the physics
- many ways for 50% heads.
- an election with $p = 0.99$ has a lot of Information

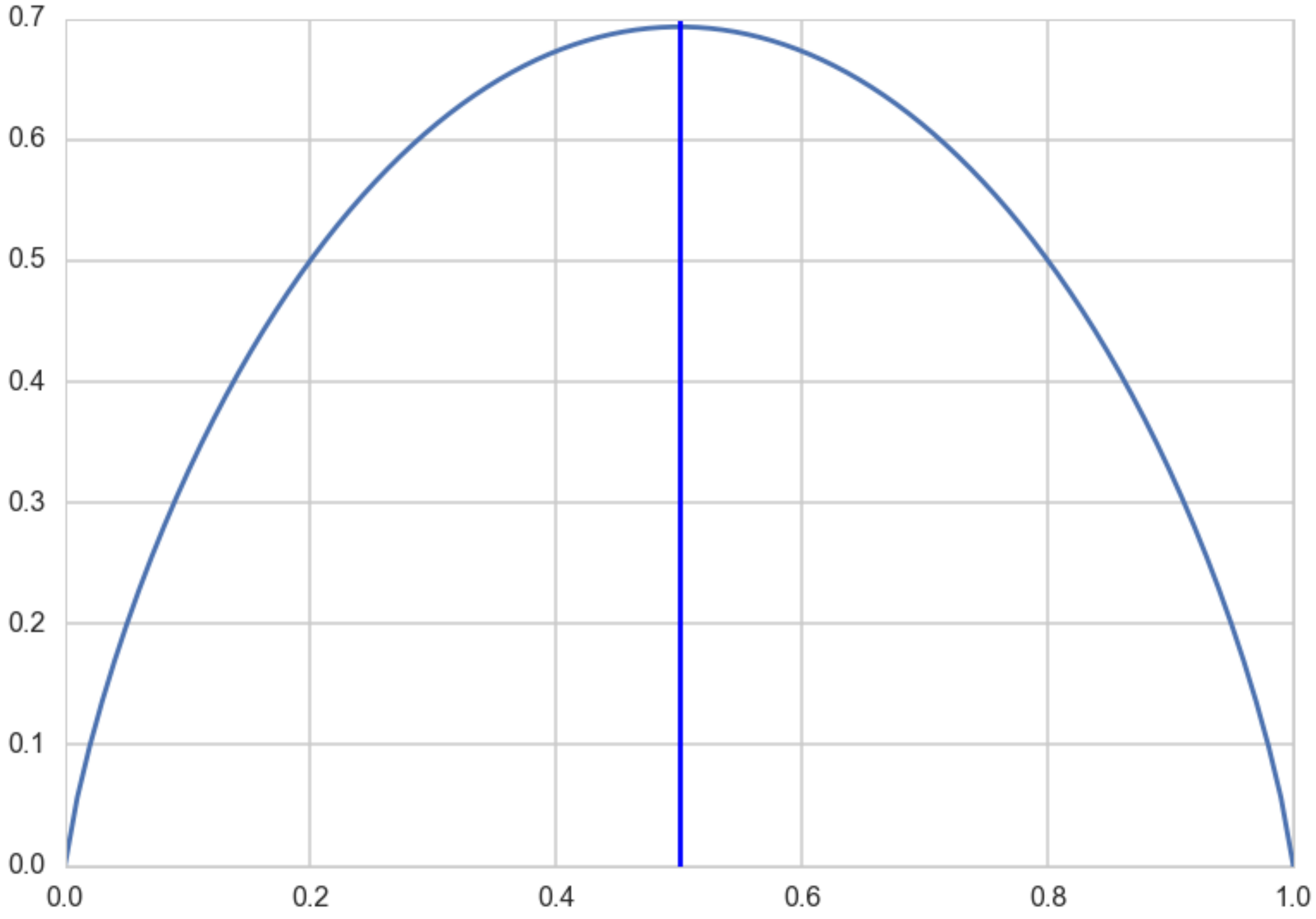
information is the reduction in uncertainty from learning an outcome

Information Entropy, a measure of uncertainty

Desiderata:
- must be continuous so that there are no jumps
- must be additive across events or states, and must increase as the number of events/states increases

$$H(p) = -E_p[log(p)] = -\int p(x)log(p(x))dx \;\; OR \; -\sum_i p_i log(p_i)$$

Entropy for coin fairness



$$H(p) = -E_p[log(p)] = -p * log(p) - (1-p) * log(1-p)$$

```
def h(p):
    if p==1.:
        ent = 0
    elif p==0.:
        ent = 0
    else:
        ent = - (p*math.log(p) + (1-p)* math.log(1-p))
```

Thermodynamic notion of Entropy

$$P(n_1, n_2, \ldots, n_M) = \frac{N!}{\prod_i n_i!} \prod_i (\frac{1}{M})^{n_i}$$

Multiplicity: $W = \frac{N!}{\prod_i n_i!}$

Entropy $H = \frac{1}{N} log(W)$ which is:

$\frac{1}{N} log(P(n_i, n_2, \ldots, n_M))$ **sans constant**

Using Stirling's approximation $log(N!) \sim Nlog(N) - N$ as $N \to \infty$ and where fractions n_i/N are held fixed:

$$H = -\sum_i p_i log(p_i)$$

A particular arrangement $\{n_i\} = (m_1, n_2, n_3, \ldots, n_M)$ is a **microstate** and the overall distribution of $\{p_i\}$, is a **macrostate**.

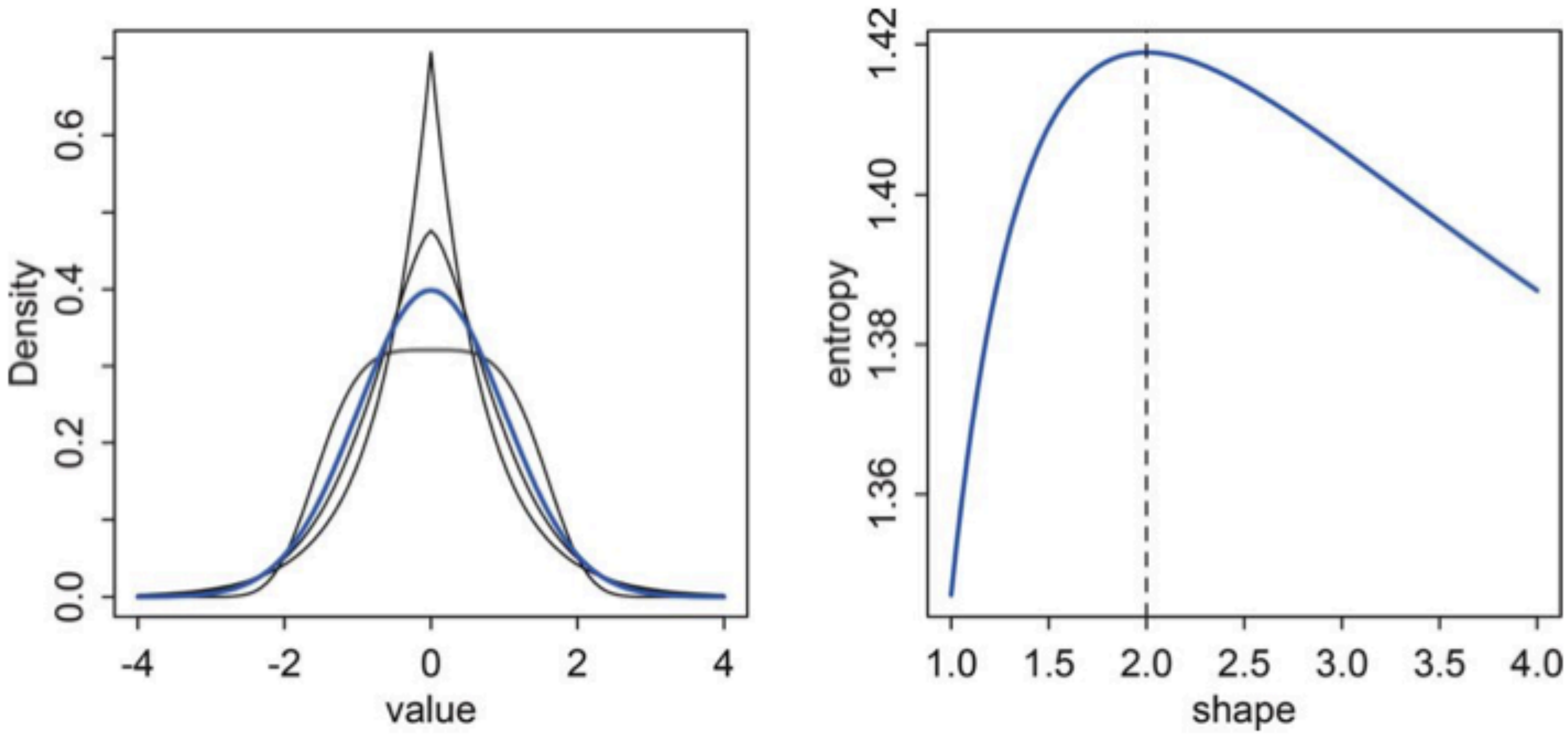
Maximize with Largrange multipliers: $p_j = 1/M$ all equal.

Maximum Entropy (MAXENT)

- finding distributions consistent with constraints and the current state of our information
- what would be the least surprising distribution?
- The one with the least additional assumptions?

The distribution that can happen in the most ways is the one with the highest entropy

Normal as MAXENT



For a gaussian

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

$$H(p) = E_p[log(p)] = E_p[-\frac{1}{2}log(2\pi\sigma^2) - (x-\mu)^2/2\sigma^2]$$

$$= -\frac{1}{2}log(2\pi\sigma^2) - \frac{1}{2\sigma^2}E_p[(x-\mu)^2] = -\frac{1}{2}log(2\pi\sigma^2) - \frac{1}{2} = \frac{1}{2}log(2\pi e\sigma^2)$$

Cross Entropy

$$H(p,q) = -E_p[log(q)]$$

Then one can write:

$$D_{KL}(p,q) = H(p,q) - H(p)$$

KL-Divergence is additional entropy introduced by using q instead of p.

We saw this for Logistic regression

- $H(p, q)$ and $D_{KL}(p, q)$ are not symmetric.
- if you use a unusual , low entropy distribution to approximate a usual one, you will be more surprised than if you used a high entropy, many choices one to approximate an unusual one.

Corollary: if we use a high entropy distribution to aproximate the true one, we will incur lesser error.

Back to the gaussian

Consider $D_{KL}(q,p) = E_q[log(q/p)] = H(q,p) - H(q) >= 0$

$$H(q,p) = E_q[log(p)] = -\frac{1}{2}log(2\pi\sigma^2) - \frac{1}{2\sigma^2}E_q[(x-\mu)^2]$$

$E_q[(x-\mu)^2]$ is CONSTRAINED to be σ^2.

$$H(q,p) = -\frac{1}{2}log(2\pi\sigma^2) - \frac{1}{2} = -\frac{1}{2}log(2\pi e\sigma^2) = H(p) >= H(q)!!!$$

Importance of MAXENT

- most common distributions used as likelihoods (and priors) are in the exponential family, MAXENT subject to different constraints.
- gamma: MAXENT all distributions with the same mean and same average logarithm.
- exponential: MAXENT all non-negative continuous distributions with the same average inter-event displacement

Importance of MAXENT

- Information entropy ennumerates the number of ways a distribution can arise, after having fixed some assumptions.
- choosing a maxent distribution as a likelihood means that once the constraints has been met, no additional assumptions.

The most conservative distribution we could choose consistent with our constraints!

Model Comparison: Likelihood Ratio

$H(p)$ cancels out!!

$$D_{KL}(p,q) - D_{KL}(p,r) = H(p,q) - H(p,r) = E_p[log(r) - log(q)] = E_p[log(\frac{r}{q})]$$

In the sample approximation we have:

$$D_{KL}(p,q) - D_{KL}(p,r) = \frac{1}{N}\sum_i log(\frac{r_i}{q_i}) = \frac{1}{N}log(\frac{\prod_i r_i}{\prod_i q_i}) = \frac{1}{N}log(\frac{\mathcal{L}_r}{\mathcal{L}_q})$$

Model Comparison: Deviance

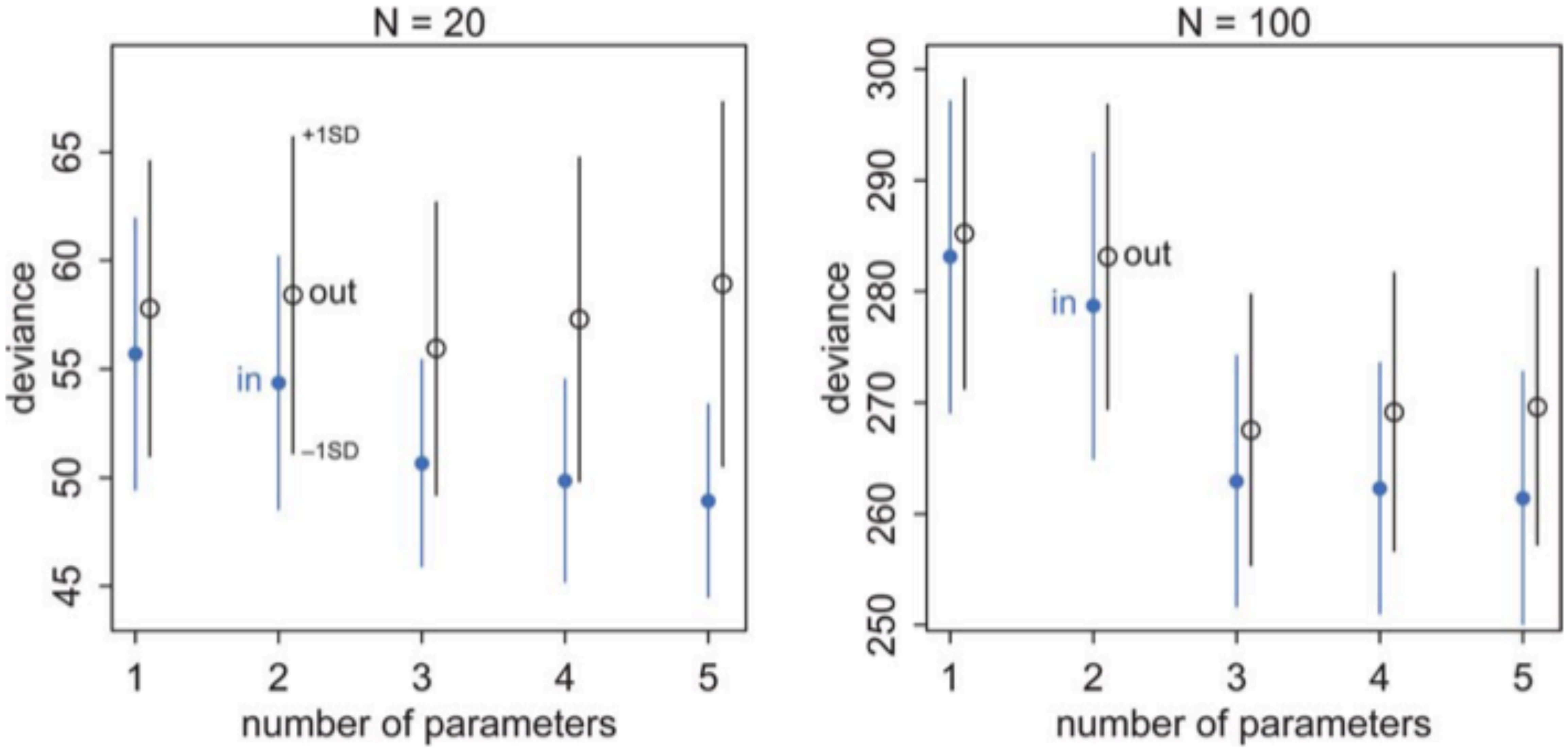
You only need the sample averages of the logarithm of r and q:

$$D_{KL}(p,q) - D_{KL}(p,r) = \langle log(r) \rangle - \langle log(q) \rangle$$

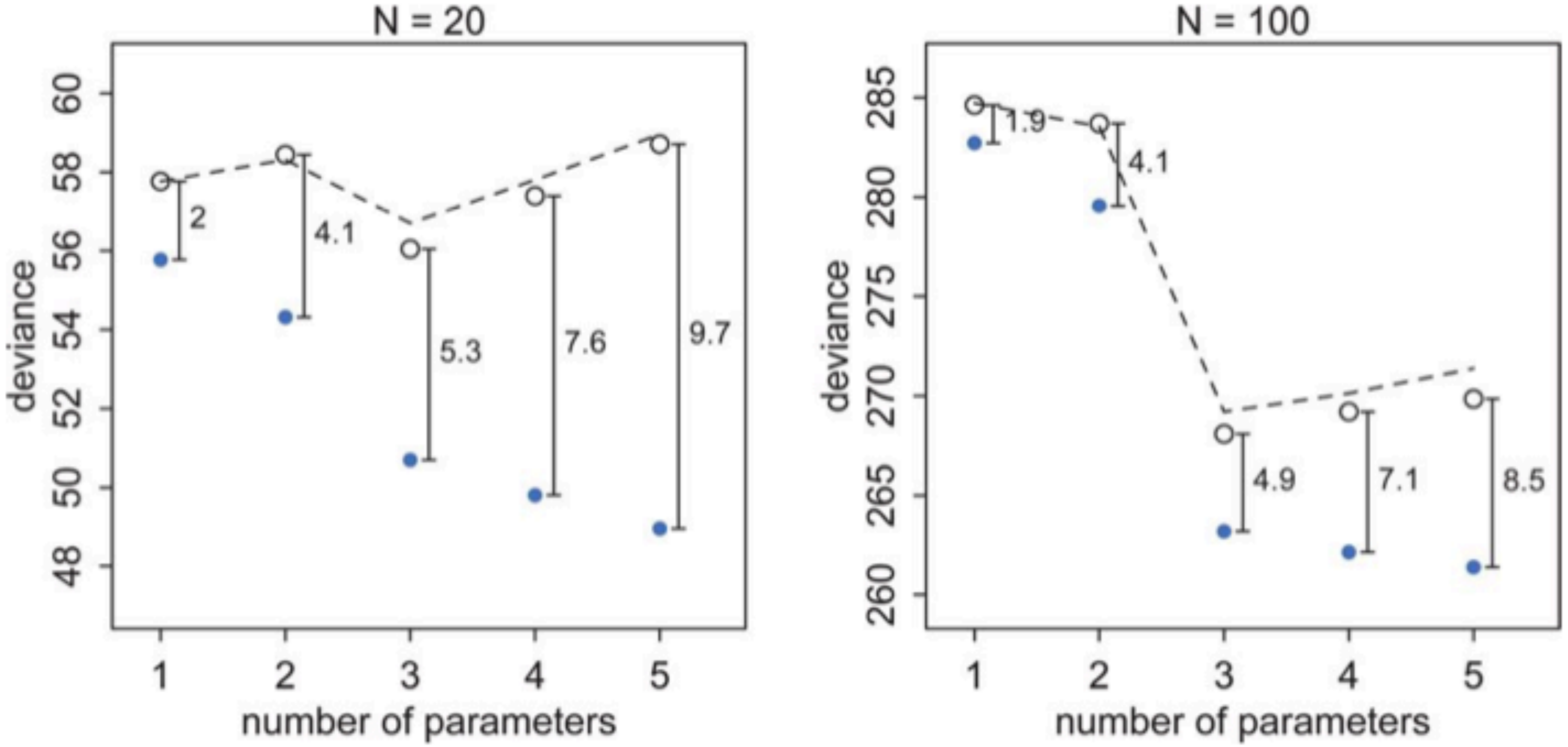
Define the deviance: $D(q) = -2 \sum_i log(q_i)$, a risk (e.g., -$2 \times \ell$, although the distribution need not be a likelihood)...

$$D_{KL}(p,q) - D_{KL}(p,r) = \frac{2}{N}(D(q) - D(r))$$

Train to Test



AIC



The test set deviances are $2 * p$ above the training set ones.

Akake **Information Criterion**:

AIC **estimates out-of-sample deviance**

$$AIC = D_{train} + 2p$$

- Assumption: likelihood is approximately multivariate gaussian.
- penalized log-likelihood or risk if we choose to identify our distribution with the likelihood: REGULARIZATION
- high p increases the out-of-sample deviance, less desirable.