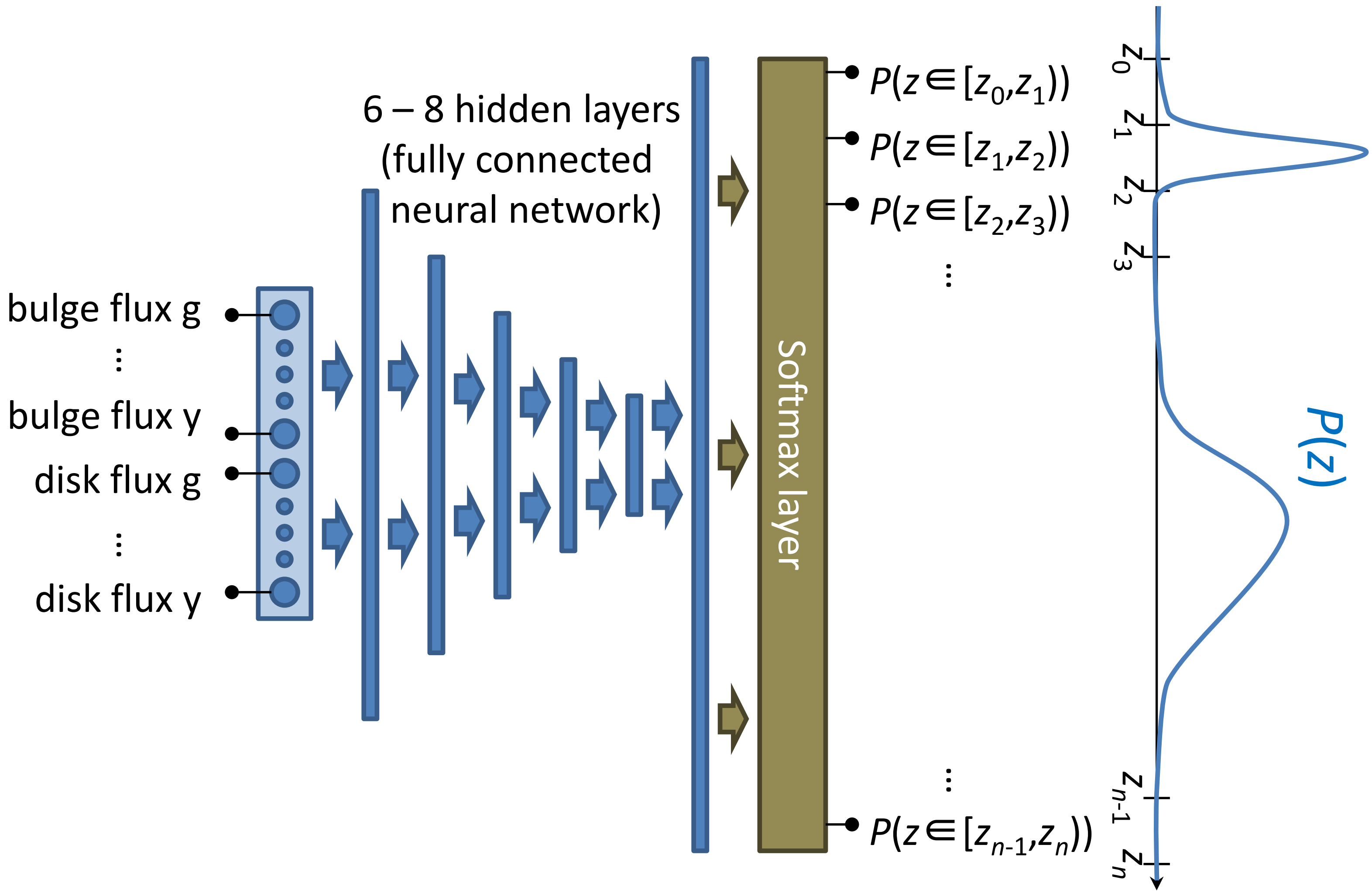
Best point estimate based on Loss & Risk

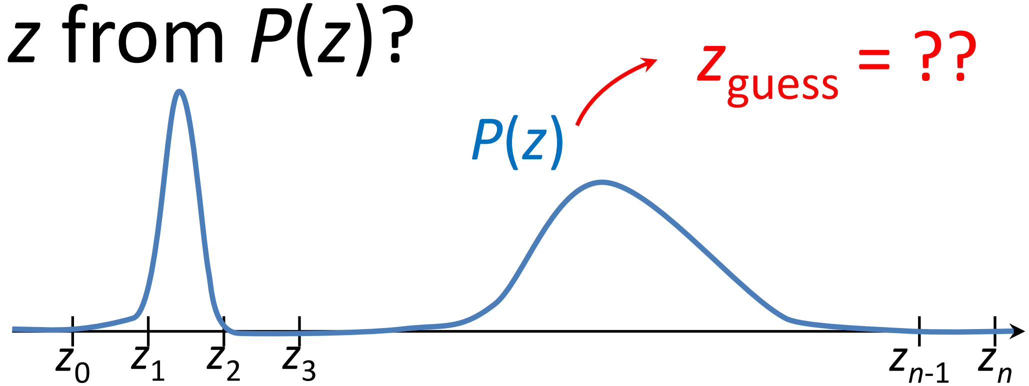
Sogo Mineo,
Masayuki Tanaka

EPHOR: Extended photometric redshift



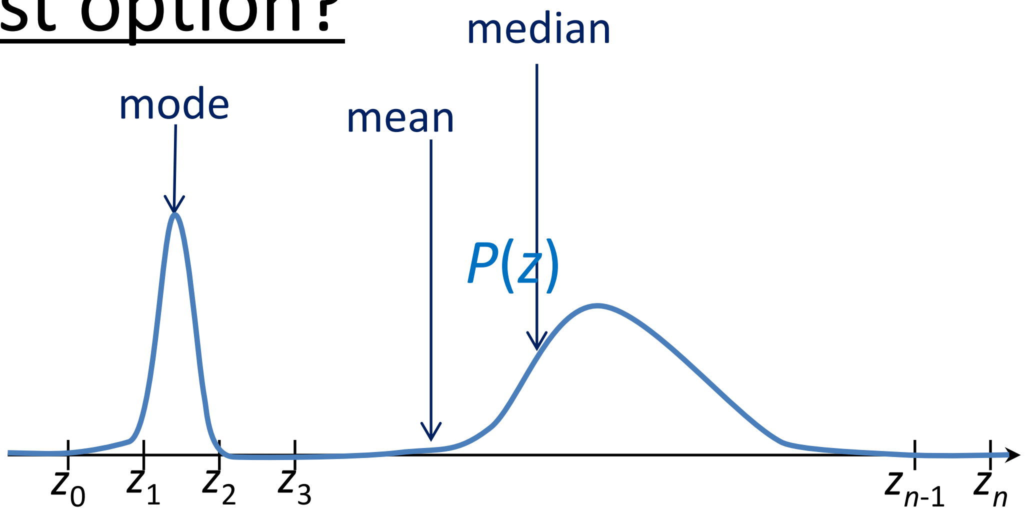
Motivation

- Many photo-z programs output a posterior probability distribution $P(z)$.
- Problem: <u>how to choose a single value</u> the topic for the redshift z from $P(z)$?



Choose a single value z_* from $P(z)$

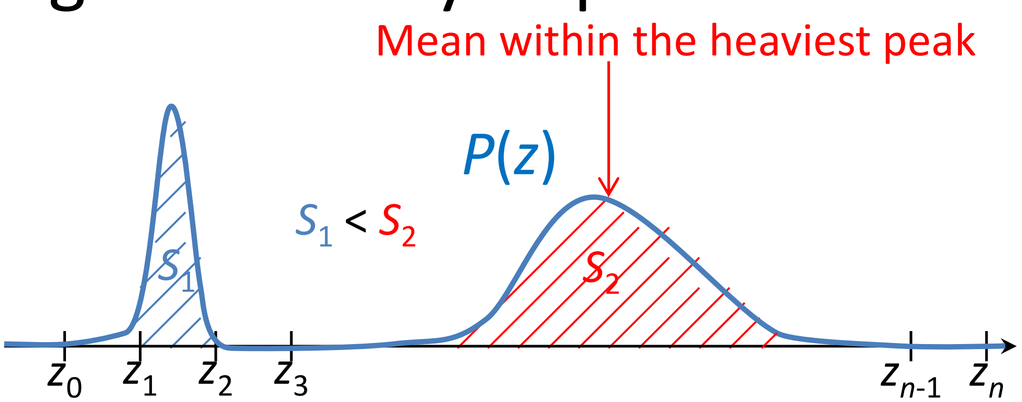
- We want a point estimate z_* from $P(z)$.
- <u>Which is the best option?</u>
  - mean?
  - mode?
  - median?



- ***None*** of them is the best.

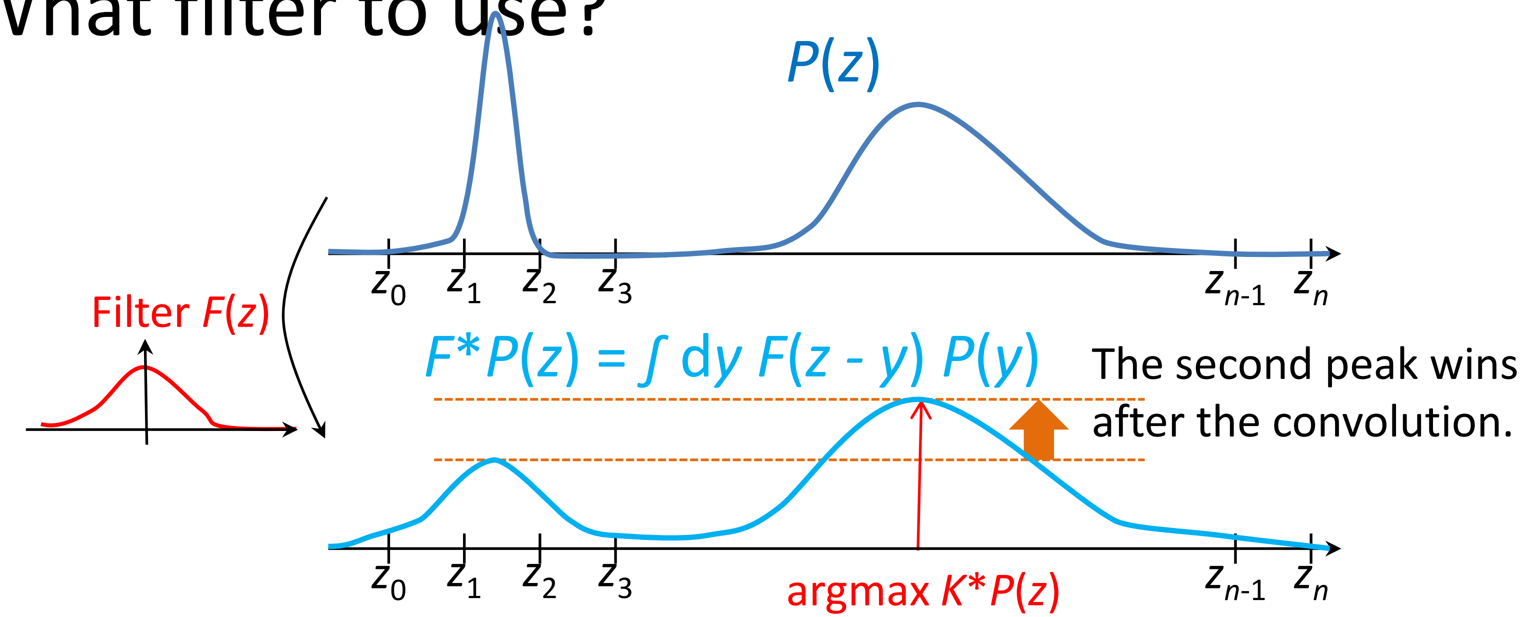
Choose a single value z_* from $P(z)$

- The best choice is the <u>mean</u> within the <u>heaviest peak</u>!
  - To compute it precisely, we would need a clustering algorithm.
    - Too complicated.
- An approximate algorithm easy to perform?



From signal detection

- In signal detection, to detect a peak, they convolve a signal with a filter $F(z)$.
  - Mean of the heaviest peak ≈ argmax $F*P$(z)
- What filter to use?



Loss and Risk

- Let us define a certain "loss"
  arising from our guessing a redshift at z_{guess}
  while the true redshift is z.
  - "loss" $L(z_{\text{guess}}, z)$

- Let us define the "risk" of a guess z_{guess}
  as the average loss expected if we adopt the guess.
  - "risk" $R(z_{\text{guess}}) = \int \mathrm{d}z\, L(z_{\text{guess}}, z)\, P(z)$

Loss as a filter

- Risk: $R(z_{\text{guess}}) = \int \mathrm{d}z\, L(z_{\text{guess}}, z)\, P(z)$

Compare!

- Filter convolution: $F{*}P(z) = \int \mathrm{d}y\, F(z - y)\, P(y)$

- The loss $L(z_{\text{guess}}, z)$ plays a role similar to the detection filter $F(z - y)$.
  - A difference is that $L(z_{\text{guess}}, z)$ is *minimal* at $z_{\text{guess}} = z$, while $F(z - y)$ is *maximal* there.

- Best choice of z is the mean of the heaviest peak
  $\approx$ argmax $F{*}P(\mathrm{z}) \approx$ argmin $R(z_{\text{guess}})$.

Loss?

- The best guess z_* proved to be the minimum-risk point:

  $$z_* = \mathrm{argmin}\, R(z_{\mathrm{guess}}),$$

  $$R(z_{\mathrm{guess}}) = \int \mathrm{d}z\, L(z_{\mathrm{guess}}, z)\, P(z).$$

- We have yet to define $L(z_{\mathrm{guess}}, z)$.
  - How should we estimate the loss of the discrepancy between the guessed redshift and the true z?

(1) Square difference as the loss

- If we were to choose square diff. as the loss:
  - $L(z_{\text{guess}}, z) = (z_{\text{guess}} - z)^2$
- Then $z_* = \text{argmin}\, R(z_{\text{guess}}) = \langle z \rangle$.
  - Best if $P(z)$ has only a single peak.
  - Worst otherwise
    - Too heavy penalty for outliers $|z_{\text{guess}} - z| \gg 1$.

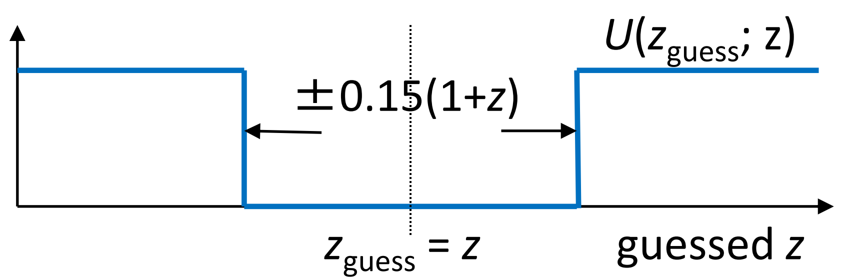
(2) Upside-down top-hat as the loss

- If we were to choose an upside-down top-hat function $U(z_{\text{guess}}, z)$ as the loss:
  "$U(z_{\text{guess}}, z)$ = 1 if $|z_{\text{guess}} - z|$ is very large, otherwise 0"

- Then the risk $R(z_{\text{guess}}) = \int dz\, U(z_{\text{guess}}, z)\, P(z)$
  would be the probability of z_{guess} being an outlier.
  E.g., if we use the definition below,
  $U(z_{\text{guess}}, z)$ agrees with the outlier criteria commonly used.

$$U(z_{\text{guess}}, z) = \begin{cases} 1 & (|z_{\text{guess}} - z| >= w(1+z)) \\ 0 & (\text{otherwise}) \end{cases},$$

$$w = 0.15$$



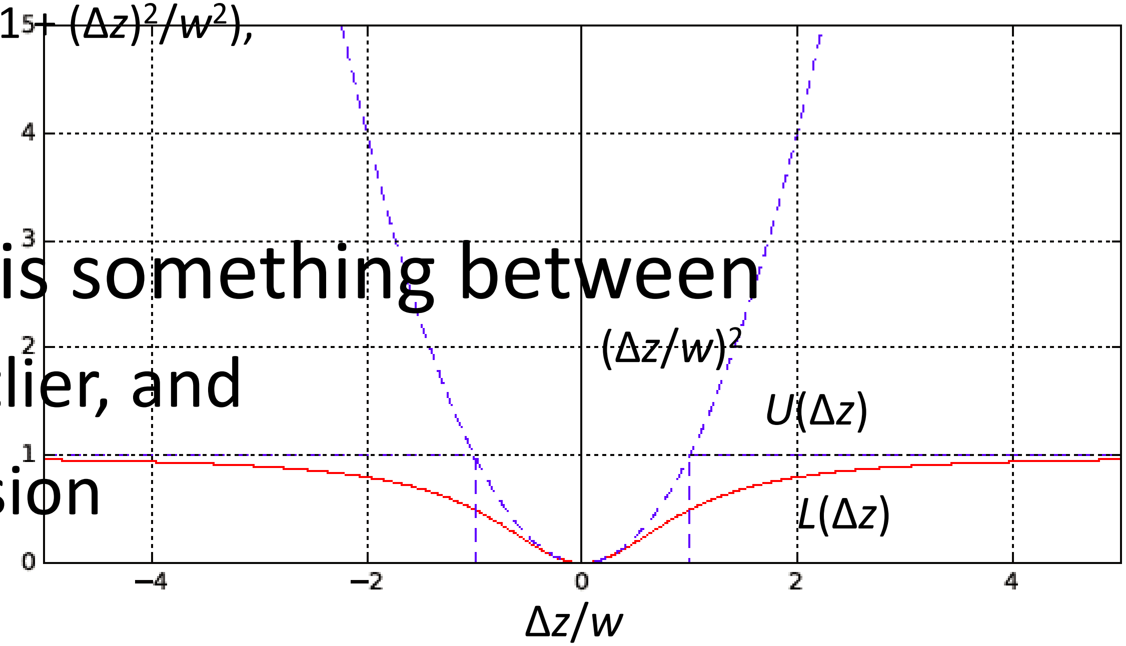
(1+2) Upside-down bell as the loss

- After all, we use an upside-down bell shape
  - that is $\simeq$ U(z_{guess}, z) outside, and
  - that is $\sim$ (z_{guess} - z)^2 inside.

$L(z_{\text{guess}}, z) = L(\Delta z) = 1 - 1/(1 + (\Delta z)^2/w^2)$,

$\Delta z = (z_{\text{guess}} - z)/(1 + z)$,

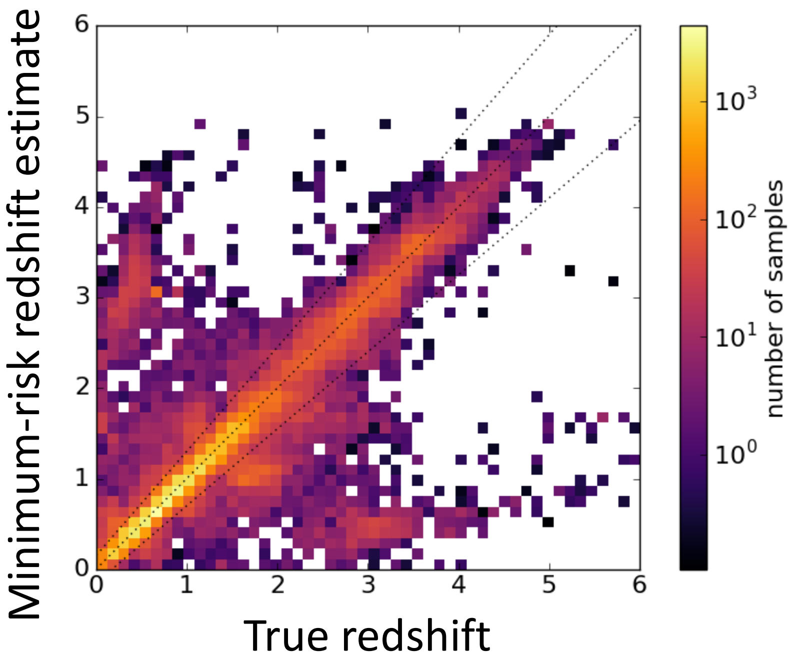
$w = 0.15$

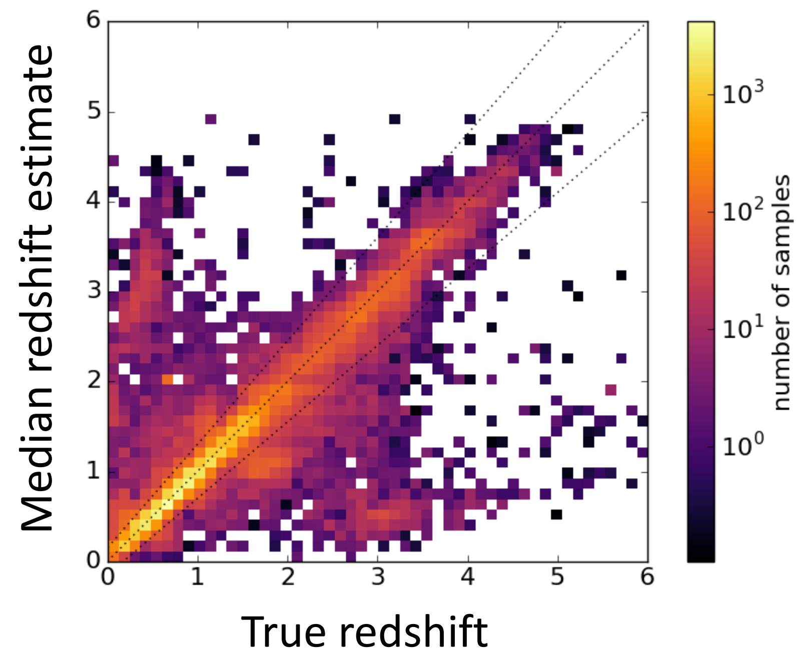


⇨ The risk $R(z_{\text{guess}})$ is something between
  - Risk of being outlier, and
  - Expected dispersion

z_* (minimum-risk z) ⇔ z_{median}

	z_*		z_{median}
Outlier fraction:	0.108(2)	(-3%)	0.111(2)
Bias [mean(Δz)][*1]:	0.0090(9)	(-40%)	0.016(1)
Dispersion [RMS(Δz)] [*1]:	0.171(8)	(-10%)	0.190(9)





(*1): $\Delta z = (z_{\mathrm{guess}} - z_{\mathrm{true}}) / (1 + z_{\mathrm{true}})$

Risk as a selection criterium

- It is natural to use the risk $R(z_*)$ in the selection of good samples:
  - "Select the sample if ***R* < threshold**."

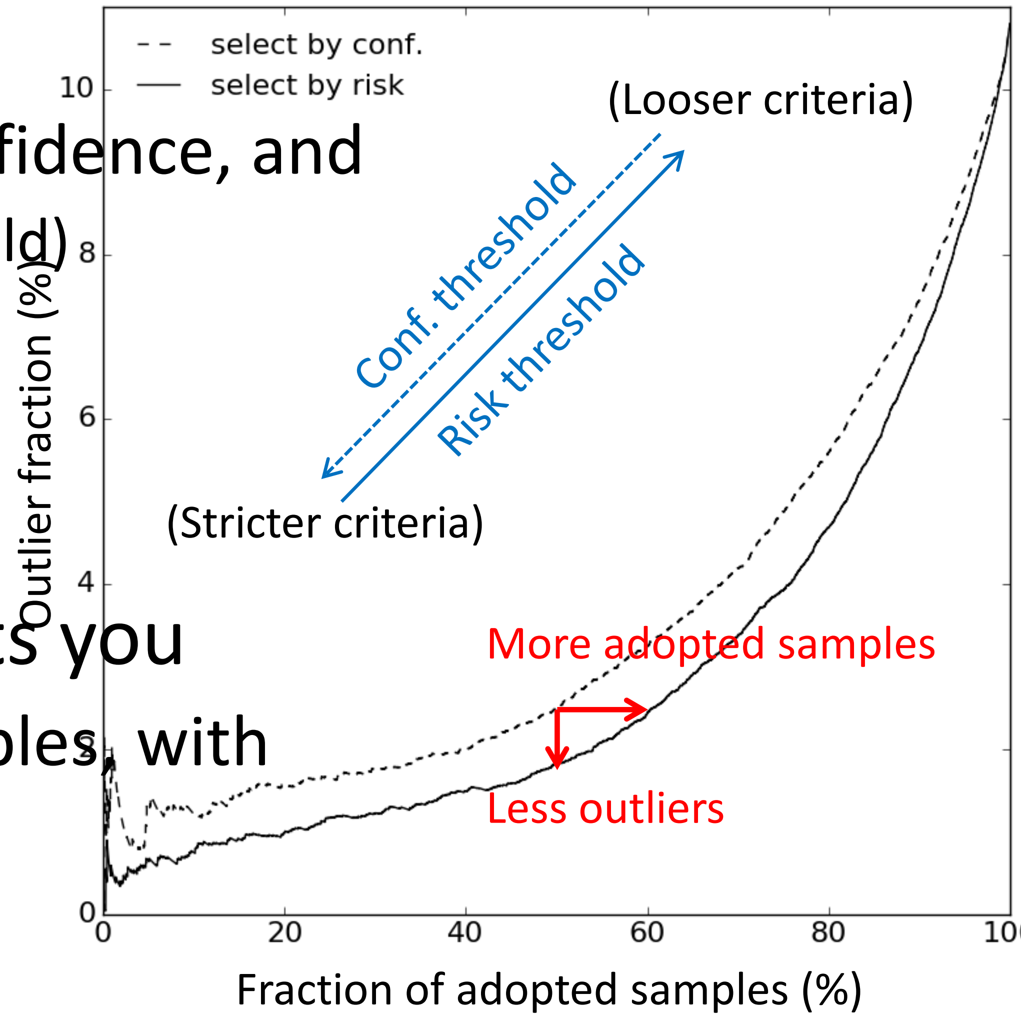
- The risk $R(z_*)$ can take the place of "confidence" used commonly:

$$\mathrm{confidence}(z_{\mathrm{guess}}) = \int \mathrm{d}z \Pi(z_{\mathrm{guess}}, z) P(z),$$

$$\Pi(z_{\mathrm{guess}}, z) = \begin{cases} 1 & (|z_{\mathrm{guess}} - z| \leq 0.03(1+z)) \\ 0 & (\mathrm{otherwise}) \end{cases}$$

Risk ⇔ Confidence

- Compare samples
  - Selected by the confidence, and
    (confidence > threshold)
  - Selected by the risk
    (Risk < threshold)



- Selection by risk gets you
  - More adopted samples, with
  - Less outliers.

Summary

- I propose a minimum-risk choice for the point estimate of redshift.
  - Included in the catalogs of HSC public/internal releases as "photoz_best"
- I propose that the risk be used as a substitute for "confidence".
  - Risk is included in the catalogs as "photoz_risk_best"

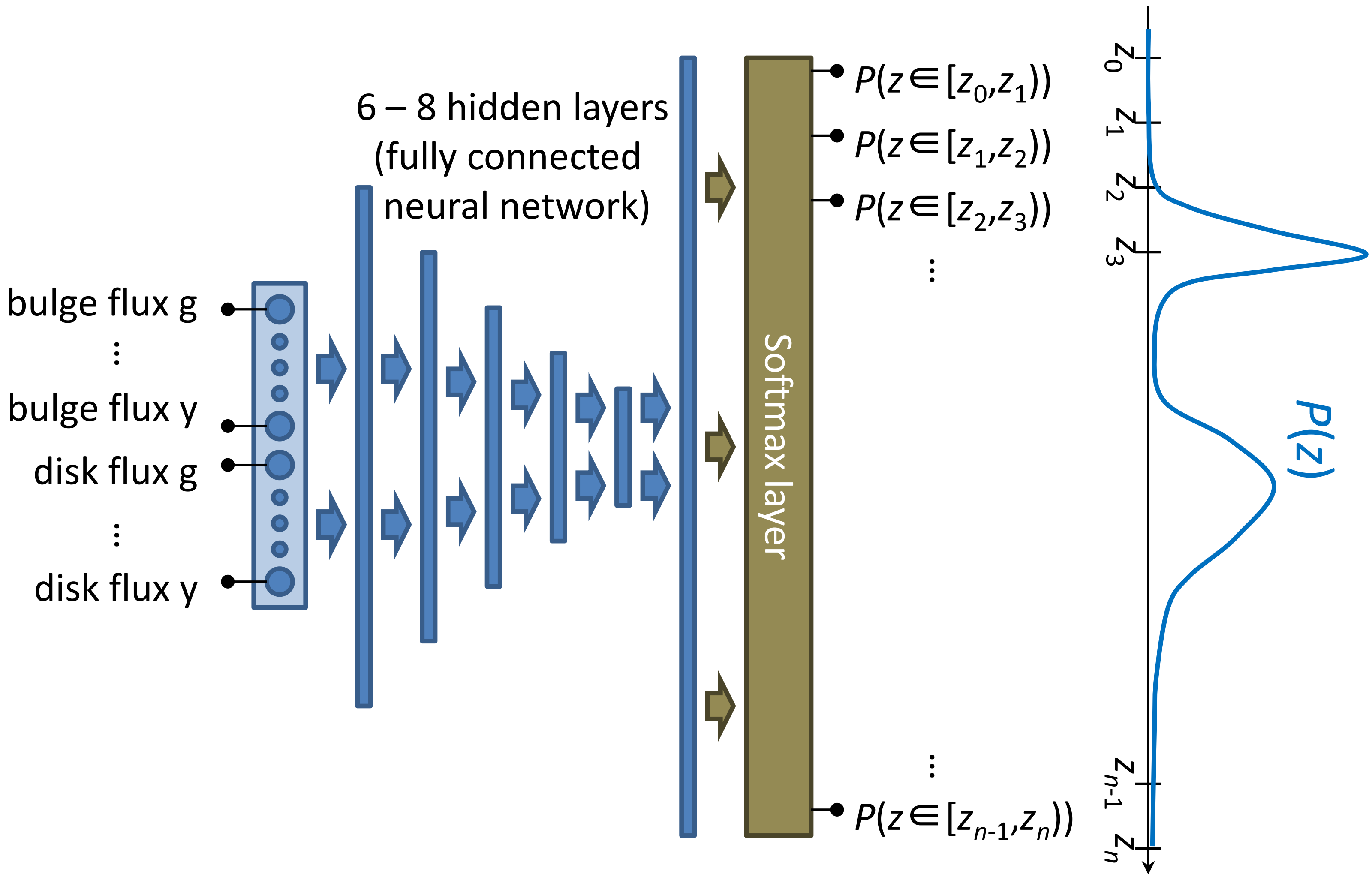
(BACKUP)

Extended Photometric Redshift

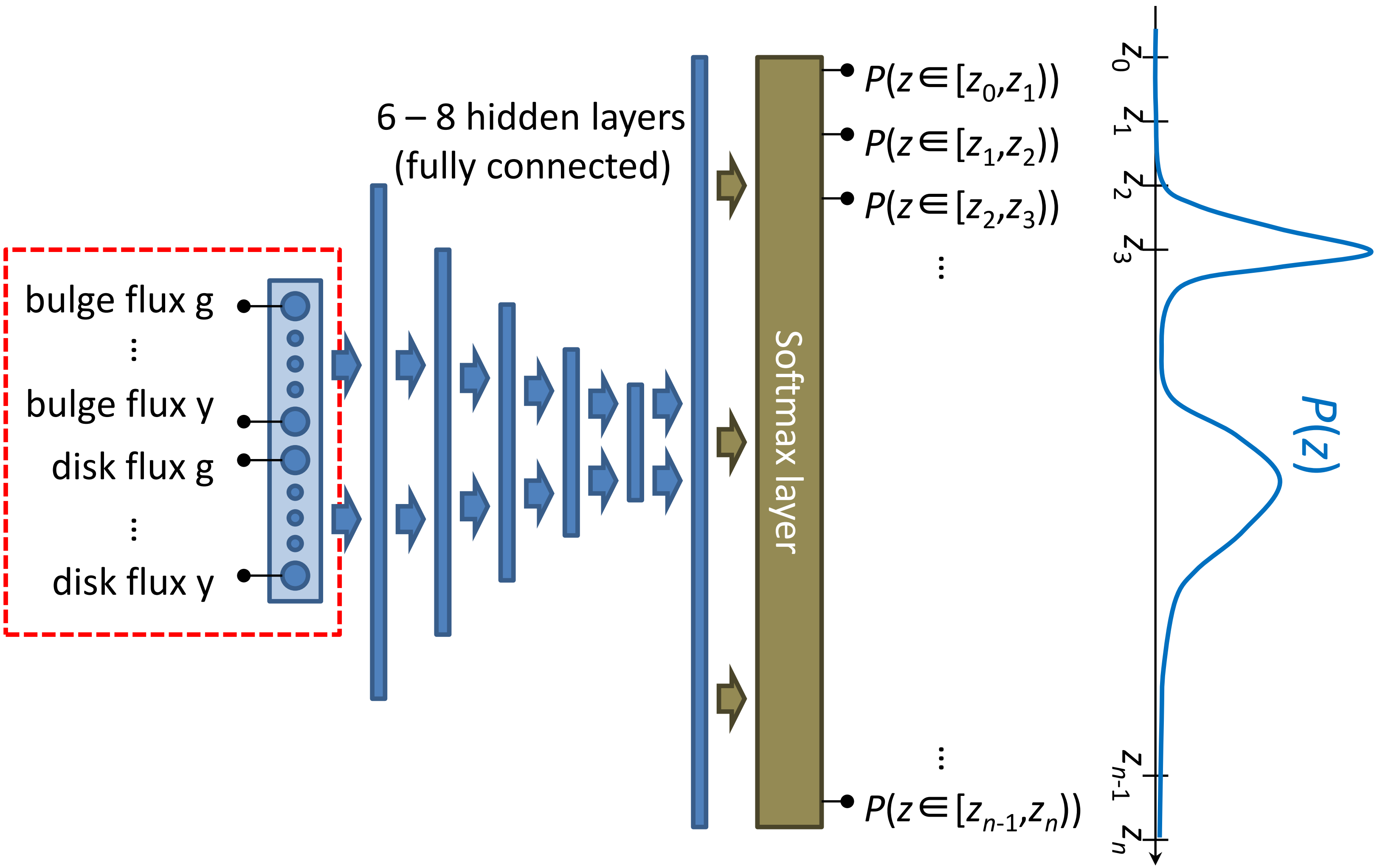
— A neural network to perform photo-z

EPHOR

EPHOR: Extended photometric redshift

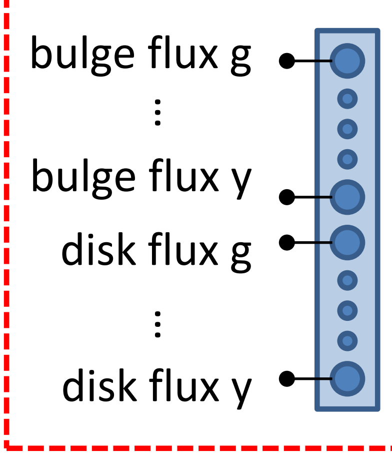


Input



6 – 8 hidden layers
(fully connected)
bulge flux g
⋮
bulge flux y
disk flux g
⋮
disk flux y
Softmax layer
$P(z \in [z_0,z_1))$
$P(z \in [z_1,z_2))$
$P(z \in [z_2,z_3))$
⋮
⋮
$P(z \in [z_{n-1},z_n))$
z_0
z_1
z_2
z_3
z_{n-1}
z_n
$P(z)$

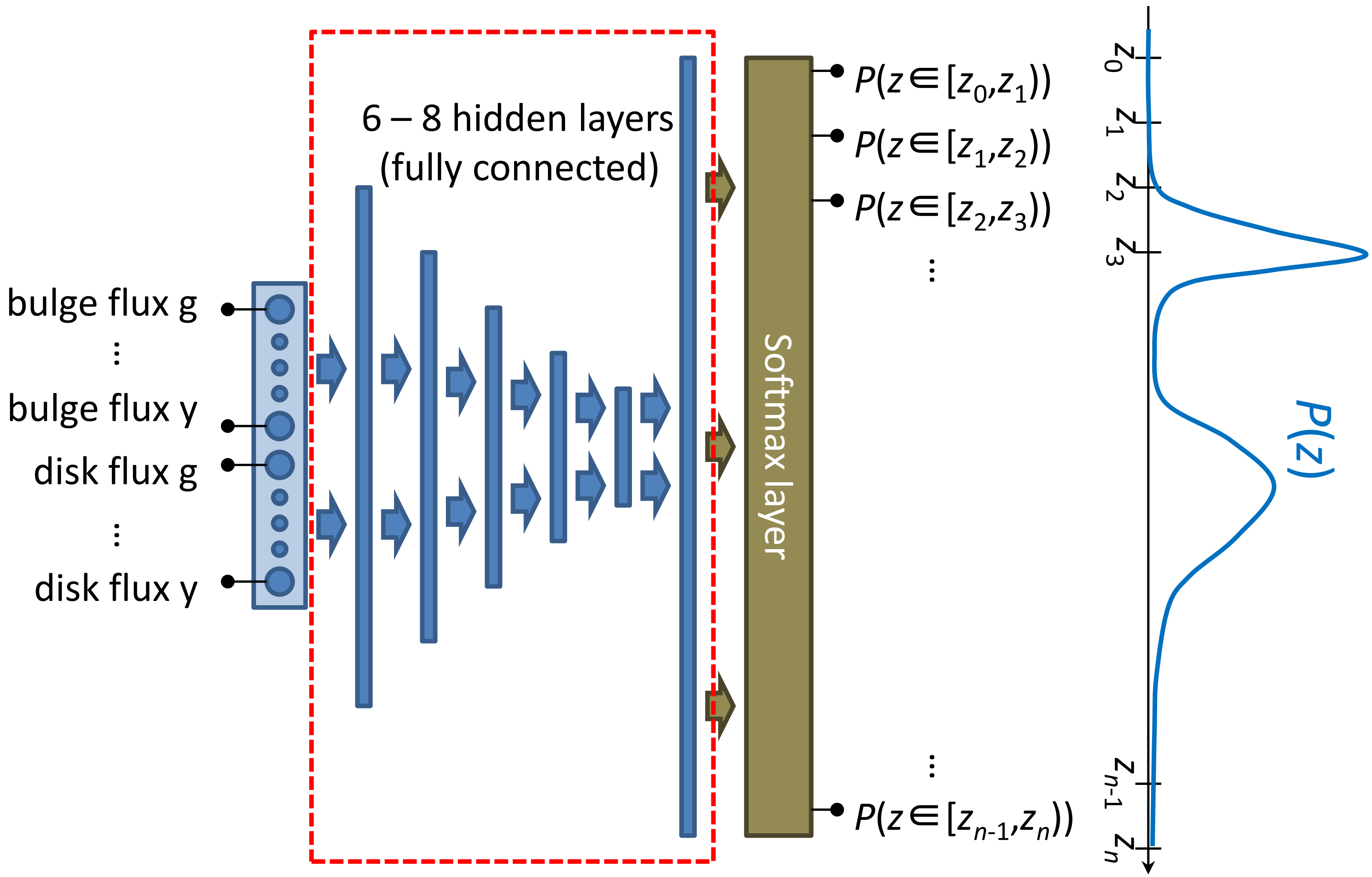
Input



- 2 components for each band are input.
  - Bulge flux & disk flux
- 5 bands for each object.
  - g, r, i, z, y
- 10 inputs in total.

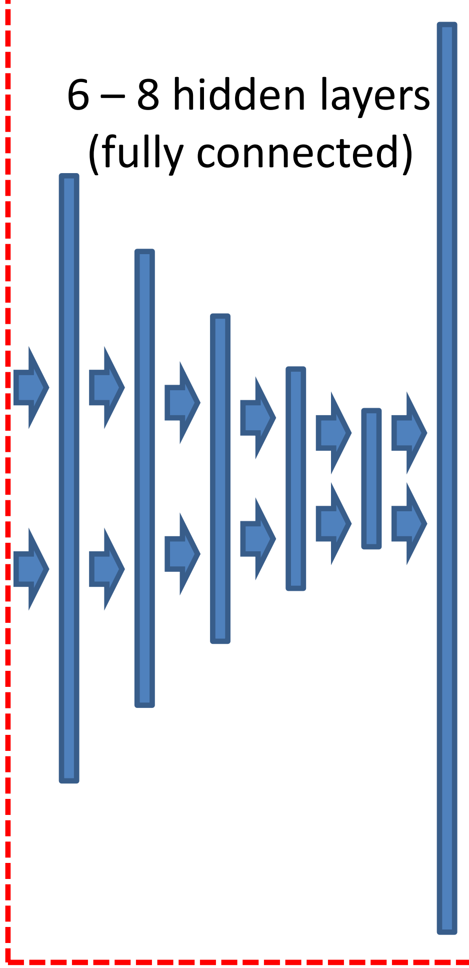
- Apparent object size is useful, but is not used
  - For fear of systematic correlation with shear measurement.

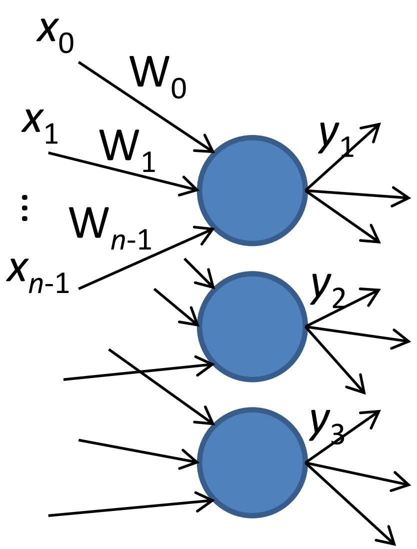
Hidden layers

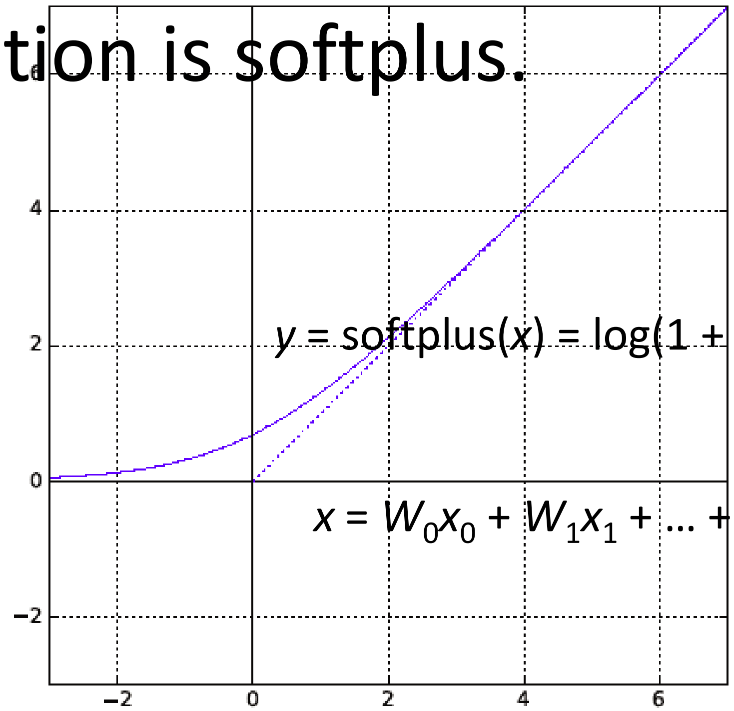


Hidden layers

- Hidden layers are fully connected.
  - All neurons in a layer are connected to all neurons in the next layer.
- Activation function is softplus.



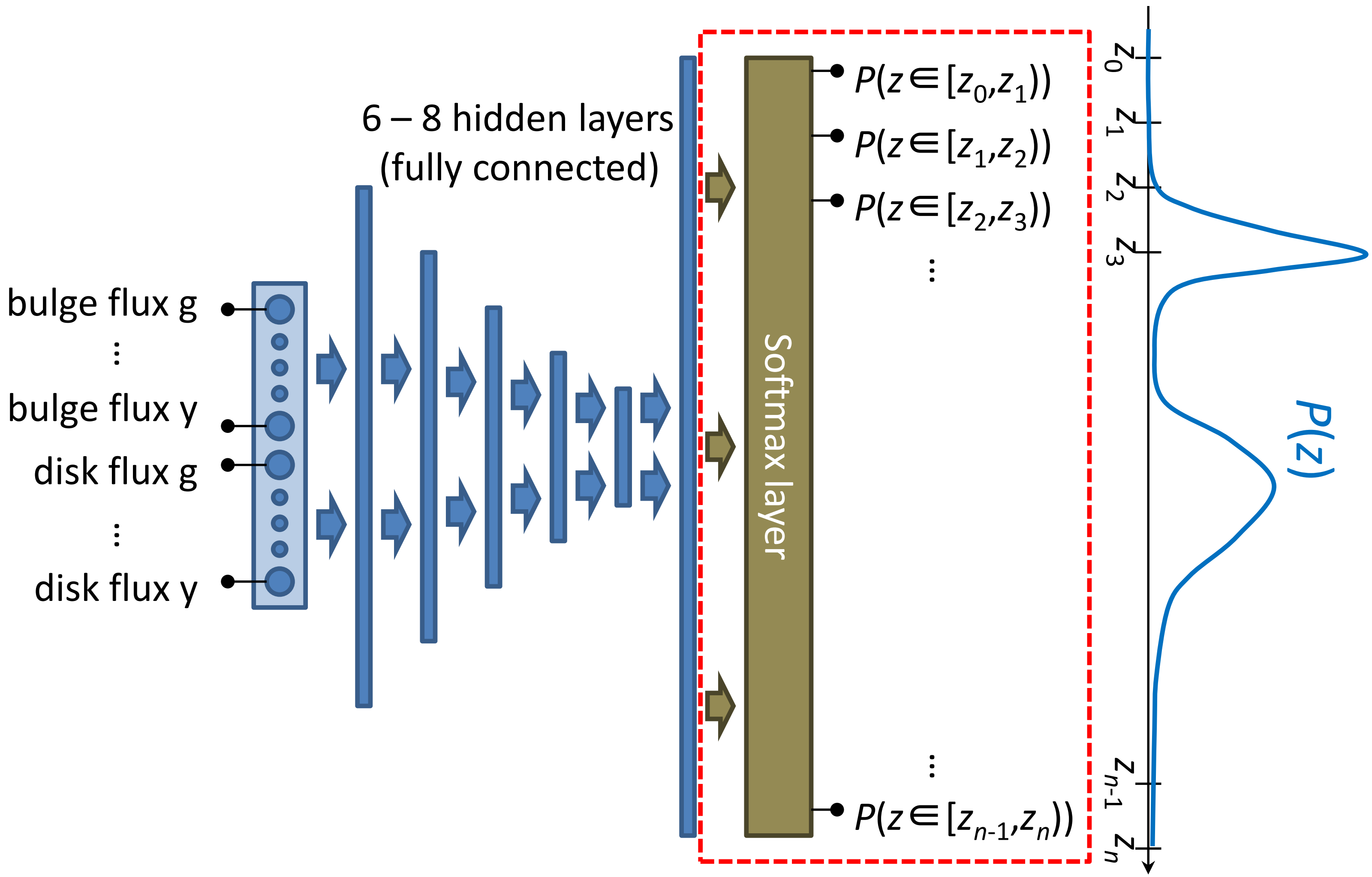




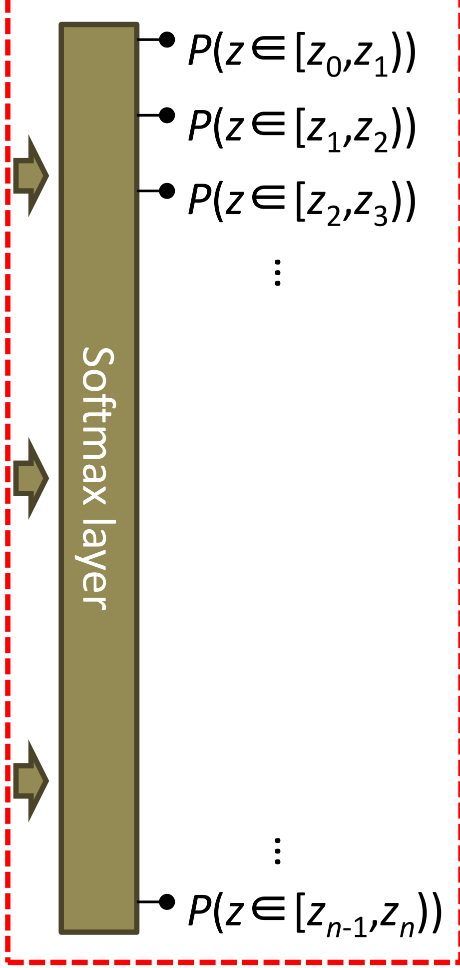
$$y = \text{softplus}(x) = \log(1 + \exp(x))$$

$$x = W_0 x_0 + W_1 x_1 + \ldots + W_{n-1} x_{n-1}$$

Output layer



Output layer



- Output layer is softmax:

  $y_i = \exp(x_i) \,/\, \Sigma_j \exp(x_j)$

  – to normalize the sum $\Sigma y_i = 1$.

- Use the cross entropy as the objective function in training:

  $H = -\Sigma\, y'_i \log y_i$ (y': supervisory value)

  – so that y_i will truly be the probability:

  $y_i = P(z \in [z_{i-1}, z_i))$.