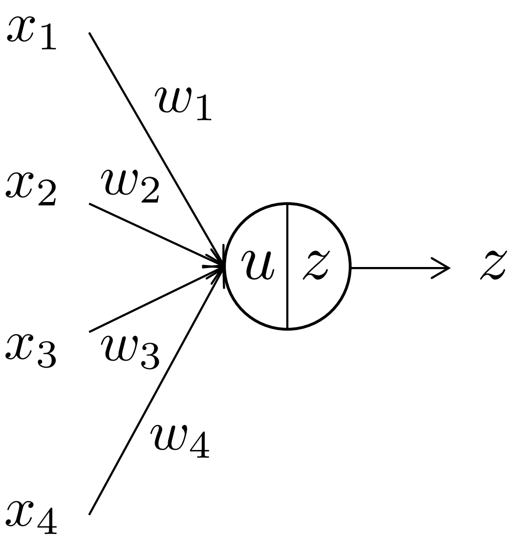
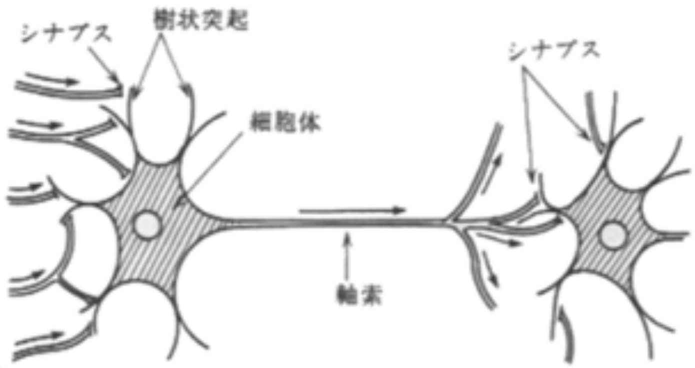
BASIC DESIGN OF NEURAL NETWORKS

Unit (or neuron or cell)

- A simplified model of a (biological) neuron





$$u = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

$$z = f(u)$$

Activation function

- *f* is called activation function

$$z = \frac{1}{1+e^{-u}}$$ Logistic (sigmoid) function

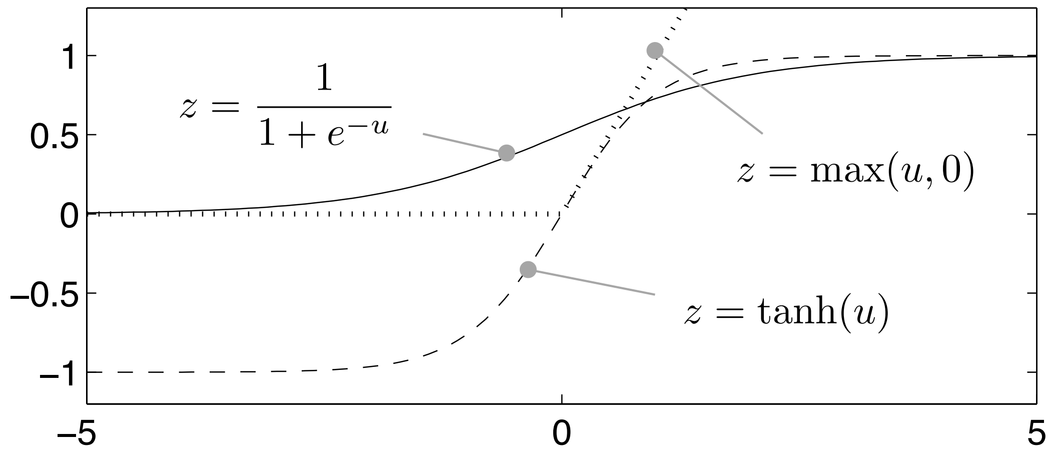
$$z = \tanh(u)$$ Hyperbolic tangent

Classical

$$z = \max(u, 0)$$ ReLU: Rectified Linear Unit ← Current standard

$$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$z = f(u)$$



Other activation functions

- So many proposals, but ReLU is still the standard

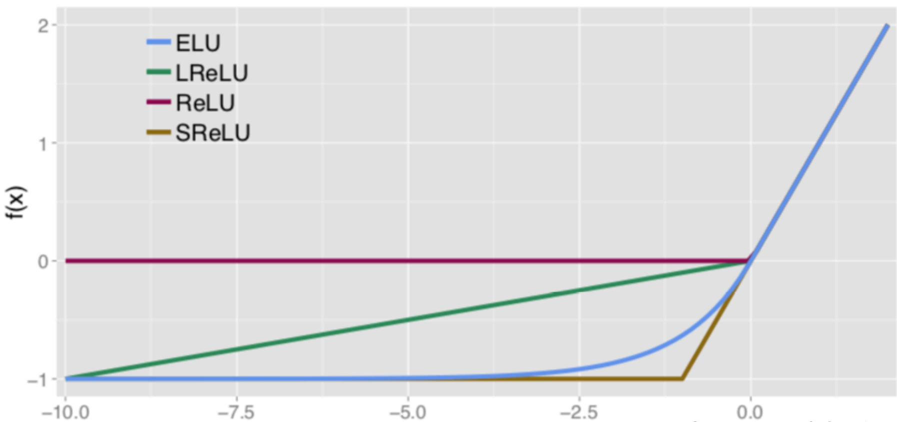
pReLU (parametric…) [He+2015]

Leaky ReLU

$$f(u) = \begin{cases} u & \text{if } \ u > 0 \\ au & \text{otherwise} \end{cases}$$

ELU (exponential…) [Clevert+2016]

$$f(u) = \begin{cases} u & \text{if } \ u > 0 \\ a(\exp(u) - 1) & \text{otherwise} \end{cases}$$

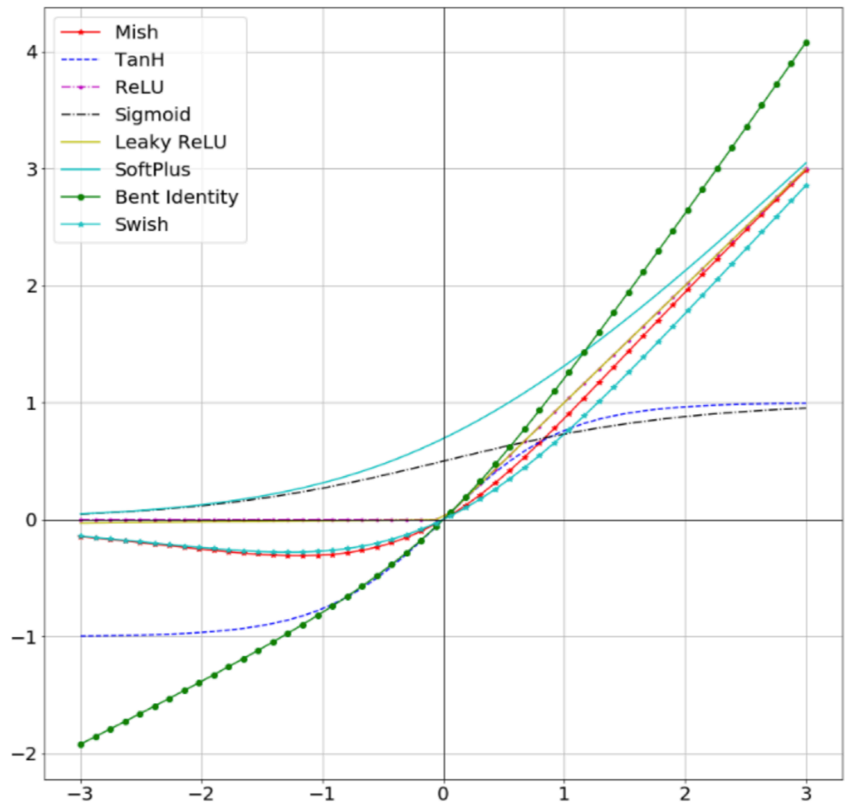


[Clevert+2016]

Swish [Ramachandran+2017]

Mish [Misra, arXiv1908]

$$f(u) = u \cdot \tanh(\log(1 + e^u))$$



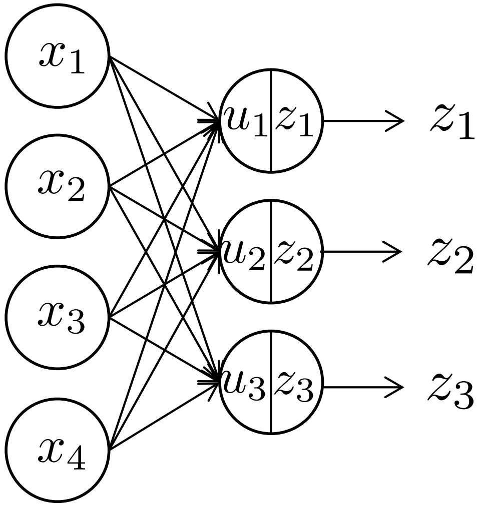
[Misra2019]

Single-layer network

- Arranging multiple units to form a layer
  - All the units share their inputs → fully-connected layer

$$u_j = \sum_{i=1}^{I} w_{ji} x_i + b_j \qquad \Rightarrow \qquad \mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$z_j = f(u_j) \qquad \qquad \mathbf{z} = \mathbf{f}(\mathbf{u})$$



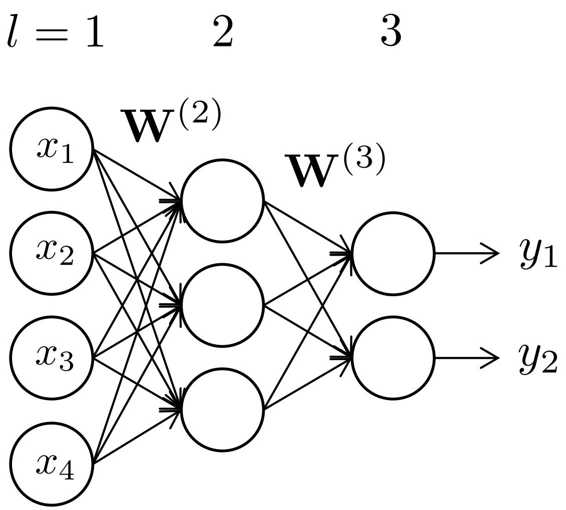
$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_J \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_I \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_J \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_J \end{bmatrix},$$

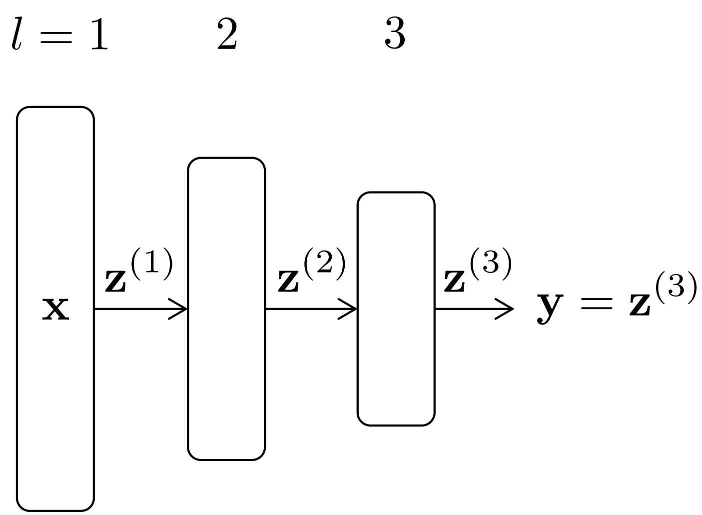
$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1I} \\ \vdots & \ddots & \vdots \\ w_{J1} & \cdots & w_{JI} \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} f(u_1) \\ \vdots \\ f(u_J) \end{bmatrix}$$

Multi-layer network

- Stacking layers → a multi-layer network
  - We will call this a feed-forward neural network

1 st layer (input)	$\mathbf{x} \equiv \mathbf{z}^{(1)}$
l th to l+1 st layer propagation	$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}$ $\mathbf{z}^{(l+1)} = \mathbf{f}(\mathbf{u}^{(l+1)})$
Last layer (output)	$\mathbf{y} \equiv \mathbf{z}^{(L)}$





Quick explanation of training neural networks

- A network expresses a function from input x to output y

$$\mathbf{y}(\mathbf{x}; \underbrace{\mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(L)}, \mathbf{b}^{(2)}, \cdots, \mathbf{b}^{(L)}}_{})$$

$$\mathbf{y}(\mathbf{x}; \mathbf{w})$$

- Assume we have a set of pairs of input x and desired output d

$$\{(\mathbf{x}_1, \mathbf{d}_1), (\mathbf{x}_2, \mathbf{d}_2), \ldots, (\mathbf{x}_N, \mathbf{d}_N)\}$$

- We adjust the parameters of the net, i.e., w = (W, b), so that it will replicate the input-output pairs:

$$\mathbf{y}(\mathbf{x}_n; \mathbf{w}) \sim \mathbf{d}_n$$

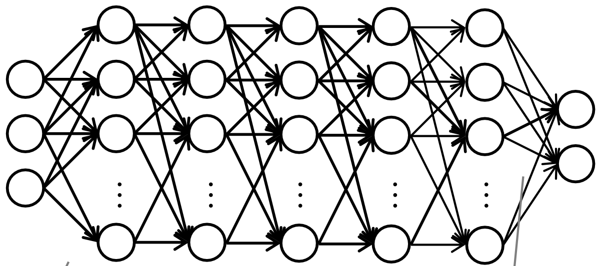
  – We hope the trained net will able to predict y for any novel x

Quick explanation of training neural networks

- Training is formulated as a minimization problem

Given a set of I/O pairs: $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{d}_1), \ldots, (\mathbf{x}_N, \mathbf{d}_N)\}$

$\mathbf{x}_n \Rightarrow$



$\Rightarrow \mathbf{y}_n \leftrightarrow \mathbf{d}_n$

$\Downarrow$

Weights：

$$\mathbf{w} = \{(\mathbf{W}^{(2)}, \mathbf{b}^{(2)}), \cdots, (\mathbf{W}^{(L)}, \mathbf{b}^{(L)})\}$$

Loss: $E(\mathbf{w}, \mathcal{S})$

We want to solve $\min_{\mathbf{w}} E(\mathbf{w}, \mathcal{S})$

- We need to specify the output layer and a loss for each problem

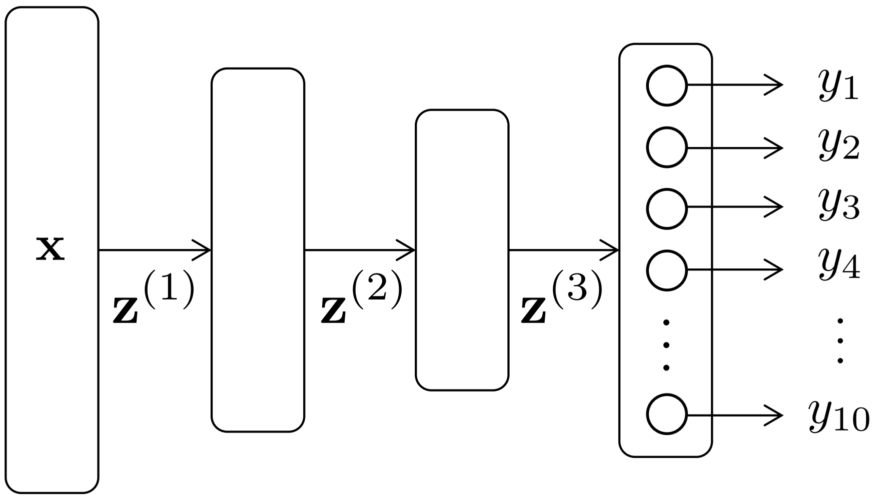
Problems and corresponding output layers/losses

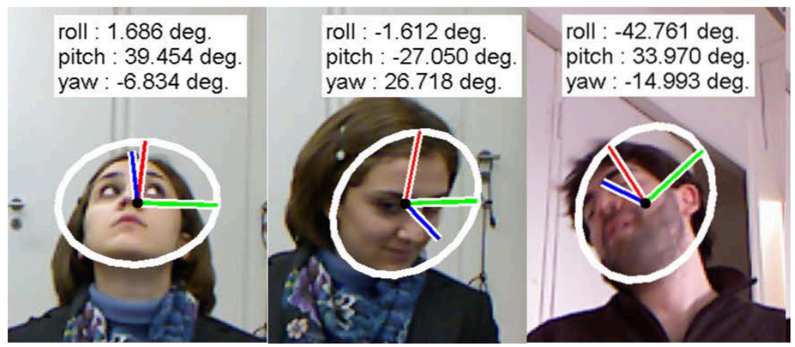
- Regression
  - E.g., Prediction of height of a person/face orientation etc.
- Multi-class classification
  - E.g., Dogs, cats, foxes, rats, …
- Multi-label classification
  - E.g., Binary attributes of a person's face in a picture; wearing eyeglasses, hat, beard, mask…
- Ordinal regression
  - E.g., Height of a person divided into 8 ranks
- Metric learning
  - E.g., Judge if two faces in different picture belong to the same person

Regression

- Output layer = the same number of units as target variables
  - Usually tanh or identity is chosen for activation func.
  - Its range should match the range of the target variables
- The most common loss: sum of the squared difference between *d* and *y*

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\|\mathbf{d}_n - \mathbf{y}(\mathbf{x}_n;\mathbf{w})\|^2$$



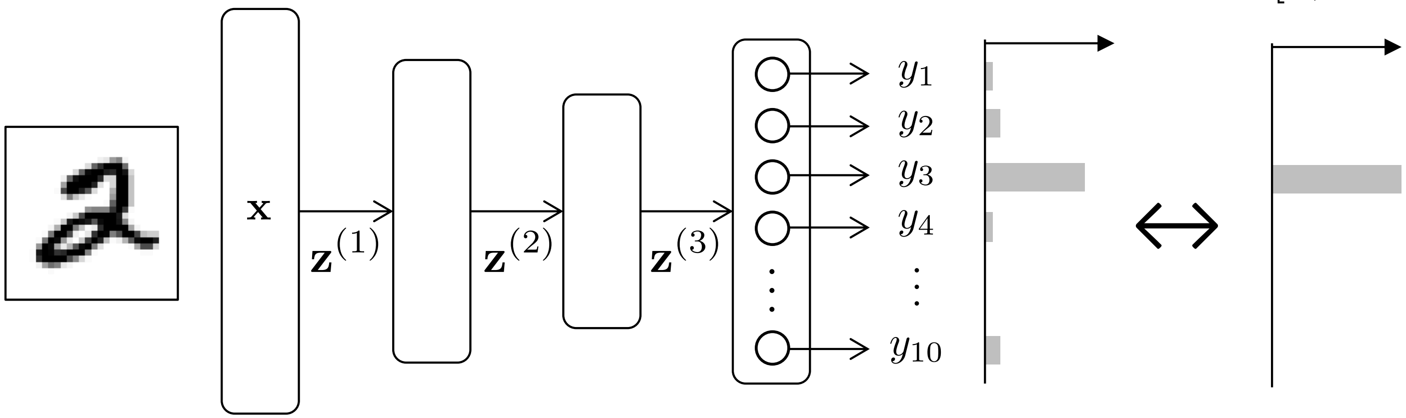


Multi-class classification

- Output layer: the same number of units as classes
- Softmax func. is used for the activation func.
- The cross-entropy loss is used for the measure between prediction and truth

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log y_k(\mathbf{x}_n; \mathbf{w})$$

$$\mathbf{d} = [0, \cdots, 0, 1, 0, \cdots, 0]$$



$$p(\mathcal{C}_k|\mathbf{x}) = y_k = z_k^{(L)} \qquad d_k$$

Softmax

- Softmax: (normalizing the outputs in range [0,1] and their sum is equal to 1
  - u_k's (inputs to the output layer) are called logits

$$y_k \equiv z_k^{(L)} = \frac{\exp(u_k^{(L)})}{\sum_{j=1}^{K} \exp(u_j^{(L)})} \qquad \left(\sum_{k=1}^{K} y_k = 1 \right)$$

- We regard the outputs y_k's as a posterior probability of class k
  - Conditional probability of class k given an input x
  - This may considered as *likelihood* of class k

$$p(\mathcal{C}_k|\mathbf{x}) = y_k = z_k^{(L)}$$

- Desired output d is usually defined to be a vector having 1 for the true class and 0 for others
  - Called one-hot vector or 1-of-K coding

Derivation of cross-entropy loss

- We build a model:
  - The output of our net represents posterior probs: $p(\mathcal{C}_k|\mathbf{x}) = y_k = z_k^{(L)}$
- Want to determine its parameter; how to do this?; an infinite ways
  - A sure way is to use *maximum likelihood estimation*
- How likely do we get the current observations from our model?
  - Observations = $\mathbf{d}_n$ for a given $\mathbf{x}_n$
  - Likelihood of the net parameter for all the N samples is

$$L(\mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{d}_n \mid \mathbf{x}_n; \mathbf{w}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathcal{C}_k \mid \mathbf{x}_n)^{d_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} (y_k(\mathbf{x}; \mathbf{w}))^{d_{nk}}$$

A standard trick

$$p(\mathbf{d} \mid \mathbf{x}) = \prod_{k=1}^{K} p(\mathcal{C}_k \mid \mathbf{x})^{d_k}$$

$$p(d_k = 1 \mid \mathbf{x}) \equiv p(\mathcal{C}_k \mid \mathbf{x})$$

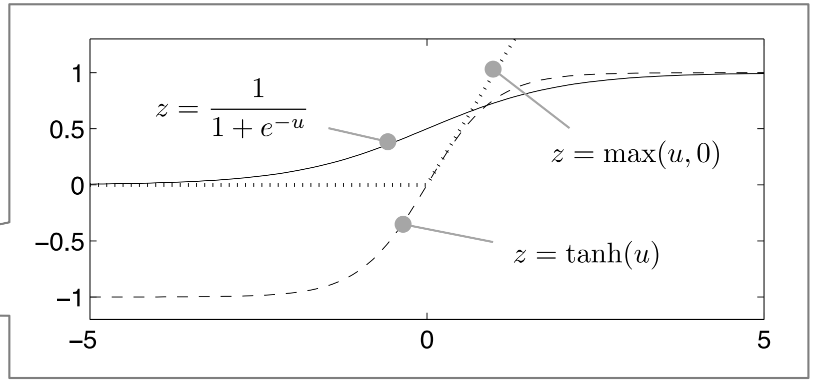
Take the logarithm and change the sign

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} d_{nk} \log y_k(\mathbf{x}_n; \mathbf{w})$$

Negative log-likelihood

Binary classification

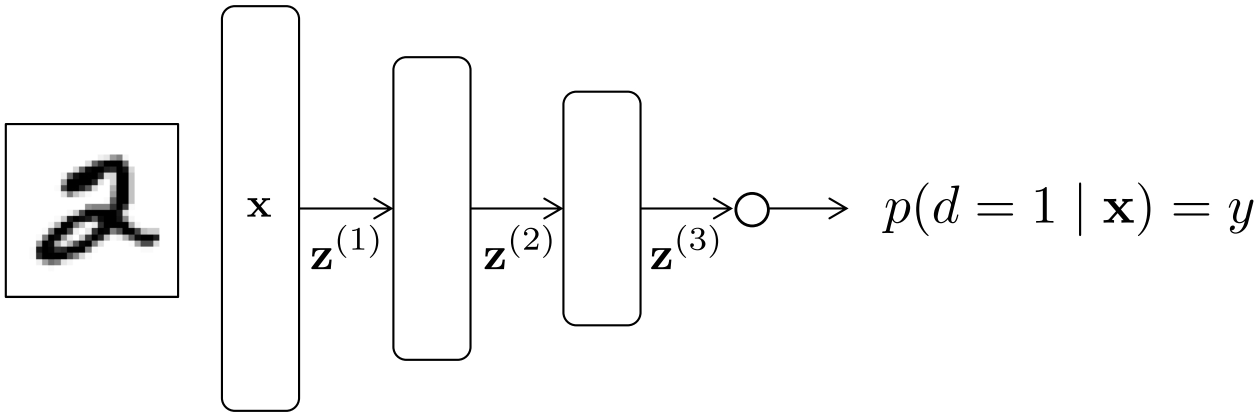
- Two solutions
  - softmax + CE loss for two classes
    - *Formulated as a multi-class classification*
  - i) logistic activafiton func. + ii) CE loss



i) $y = 1/(1 + \exp(-u))$

ii) $$E(\mathbf{w}) = -\sum_{n=1}^{N} [d_n \log y(\mathbf{x}_n; \mathbf{w}) + (1 - d_n) \log \{1 - y(\mathbf{x}_n; \mathbf{w})\}]$$

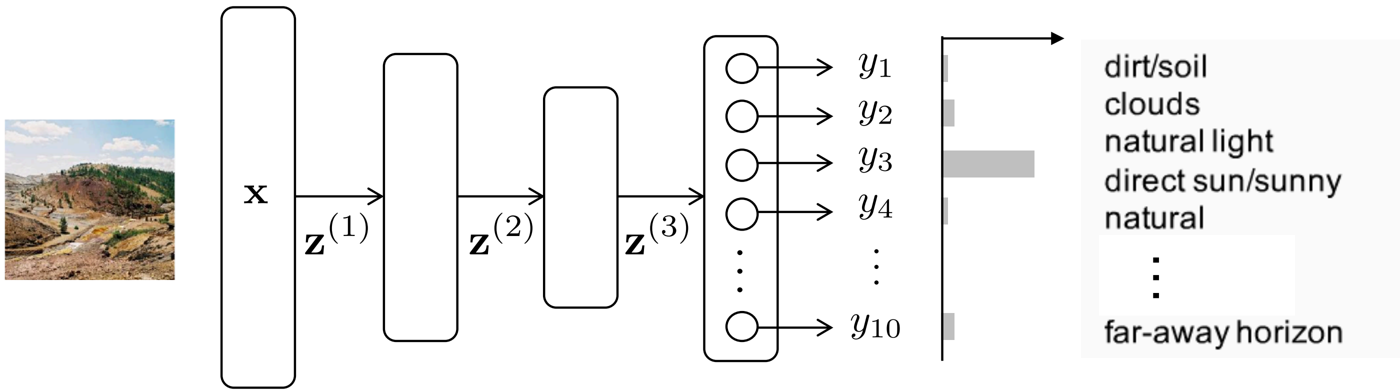
$$\left(L(\mathbf{w}) \equiv \prod_{n=1}^{N} p(d_n \mid \mathbf{x}_n; \mathbf{w}) = \prod_{n=1}^{N} \{y(\mathbf{x}_n; \mathbf{w})\}^{d_n} \{1 - y(\mathbf{x}_n; \mathbf{w})\}^{1-d_n} \right)$$

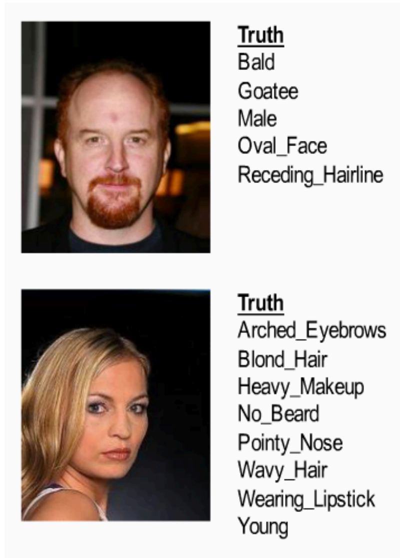


Multi-label classification

- Multi-label classification = binary classification for each label
- Output layer: the same number of units as labels
  - Activation func.: logistic sigmoid
  - Output of k^{th} unit = likelihood of label k for input x
- Sum of cross-entropy loss for each label

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk}\log y_k(\mathbf{x}_n;\mathbf{w}) + (1-d_{nk})\log(1-y_k(\mathbf{x}_n;\mathbf{w}))$$

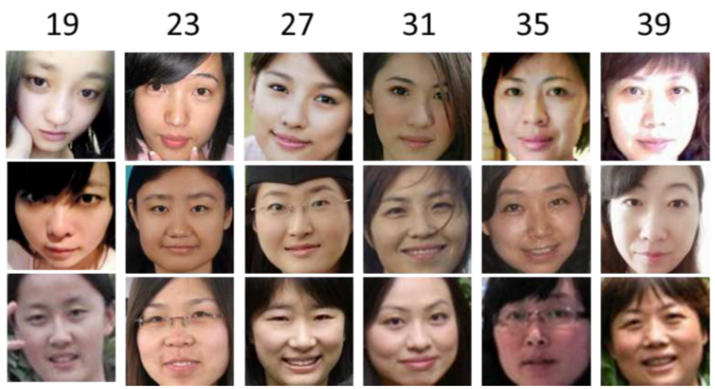




[Yamaguchi+2017]

Ordinal regression

- Looks similar to multi-class classification, but differs in that there is order in the classes
- E.g., We want to predict the age (e.g., 0-99) of a person from its face image
  - 100-class classification? → A slight error is penalized equally to large errors, e.g., 32(true) vs. 31(pred); and 32 vs. 50
  - We need to take distance between prediction and its truth into account



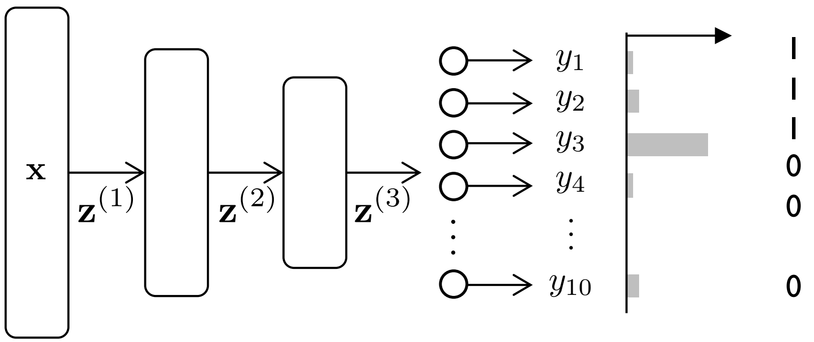
Ordinal regression: two approaches

- Convert into *K*-1 independent binary classifications
  - k^{th} unit predicts if x is larger than k^{th} class → yes(1) or no(0)

$$
\begin{cases} d_k = 1 & \text{if } \ r_k < r, \\ d_k = 0 & \text{otherwise} \end{cases}
$$

- Sum of the *K*-1 binary classification results gives the class id

$$
q = 1 + \sum_{k=1}^{K-1} f_k(\boldsymbol{x}').
$$

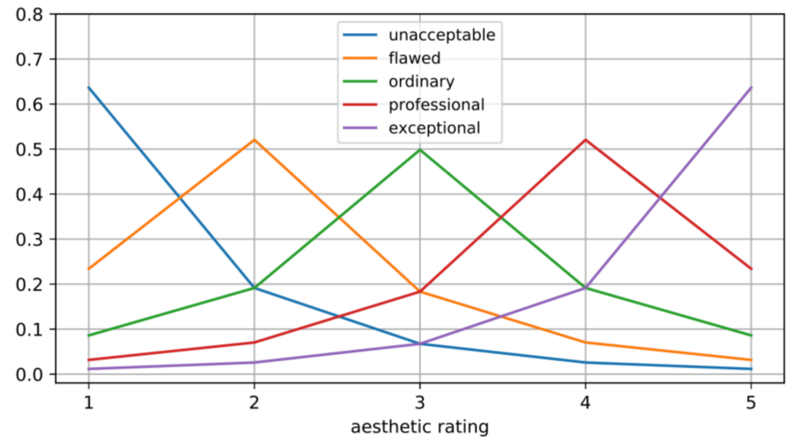


Niu+, Ordinal Regression with Multiple Output CNN for Age Estimation, CVPR2016

- Set the target label to be a soft label

$$
d_k = \frac{e^{-\phi(r_t, r_k)}}{\sum_{i=1}^{K} e^{-\phi(r_t, r_i)}}
$$

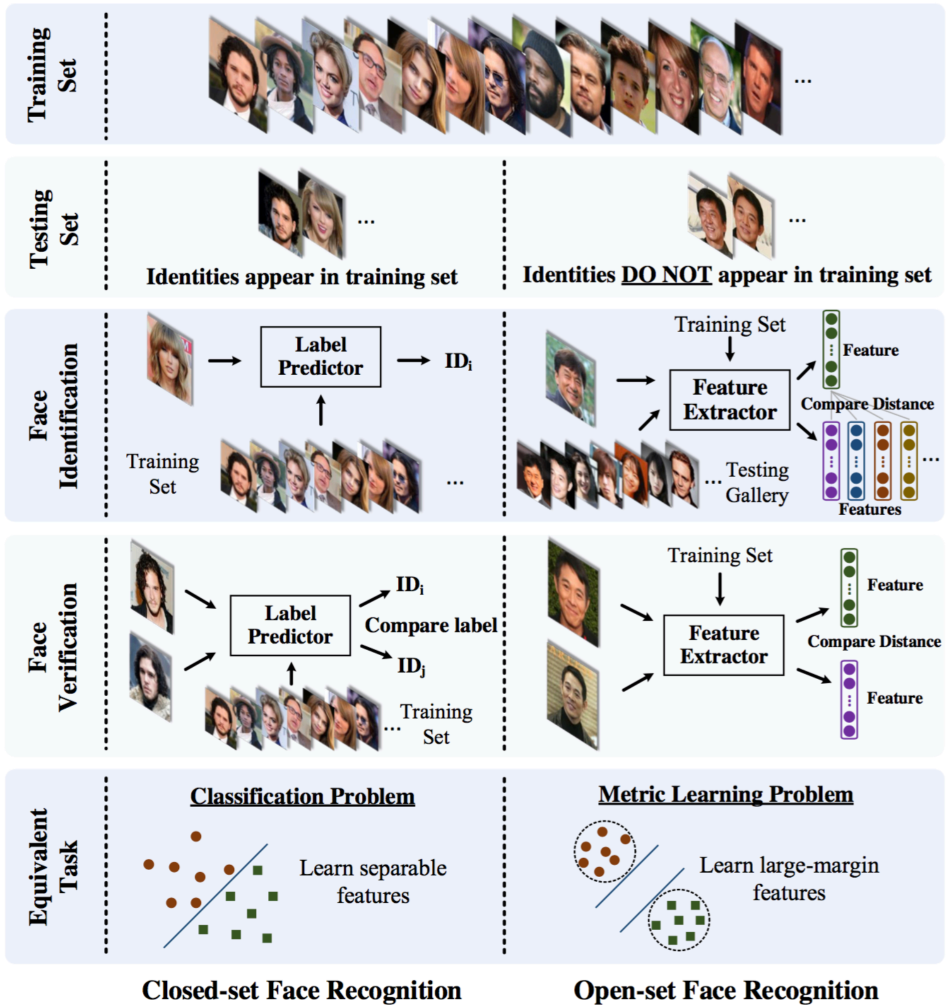
- Formulate as *K*-class classification
  - Use an ordinary model and train it to predict the soft label



Diaz-Maarthe, Soft Labels for Ordinal Regression, CVPR2019

Metric learning

- The set of classes to recognize is not closed but open
  - Aka. similarity learning, distance (metric) learning
  - E.g., Face verification; in a border control, a novel person's face needs to be matched against a passport picture
- We wish to learn a feature space suitable for the task



[Liu+, SphereFace, 2017]

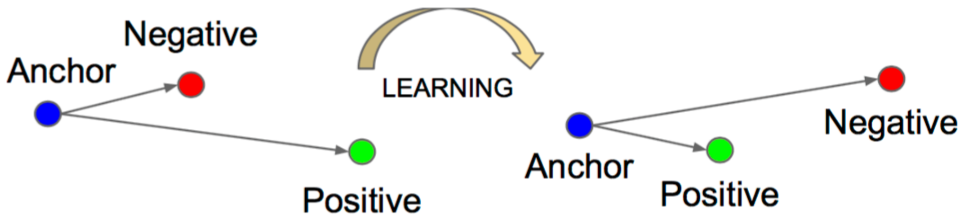
Metric learning: two approaches

- A pair of samples of the same class should be mapped to close points, while those of different classes should be mapped distant points

- Formulated as multi-class classification but with slightly different formulation
- Intra-class dispersion should be minimized, while inter-class dispersion should be maximized

Triplet loss:

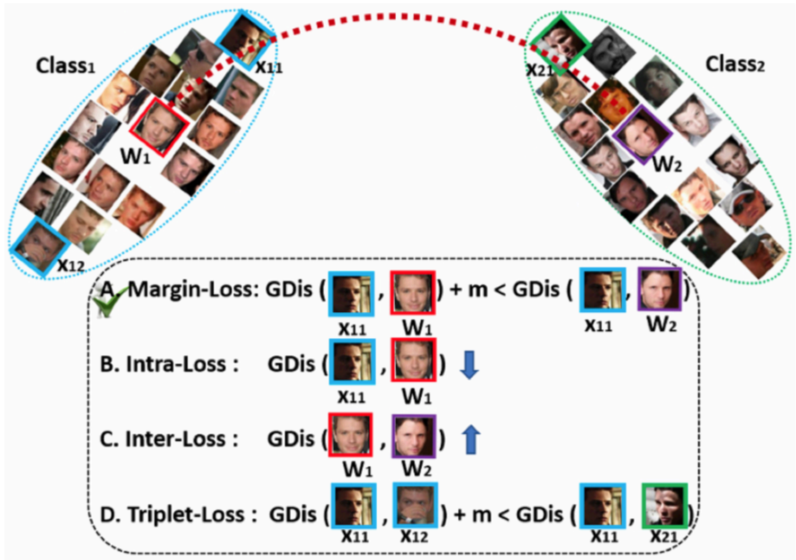
$$L = \|f(x_a) - f(x_p)\|_2^2 - \|f(x_a) - f(x_n)\|_2^2$$



[Schroff+, DeepFace, 2015]

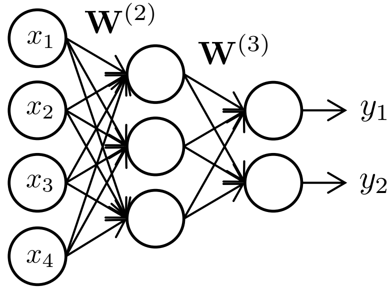
Contrastive loss:

$$L = \begin{cases} \|f(x_i) - f(x_j)\|_2^2, & \text{if } i \text{ and } j \text{ same identity} \\ \max(0, m - \|f(x_i) - f(x_j)\|_2)^2, & \text{otherwise} \end{cases}$$

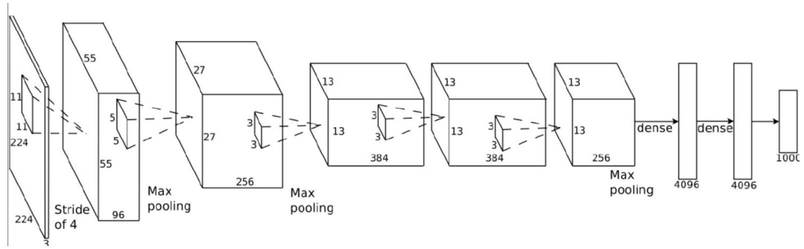


[Deng+, ArcFace, 2019]

Overview of various networks



Feed-forward nets



Convolutional nets
(1D, 2D, 3D)

Fixed-size inputs

Graph neural nets

Graphs

Supervised

Unsupervised

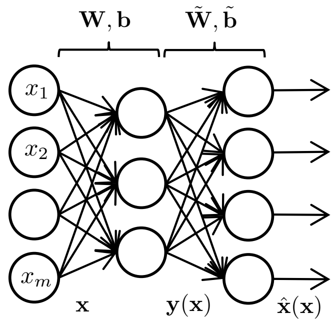
Variable-size inputs

Sets

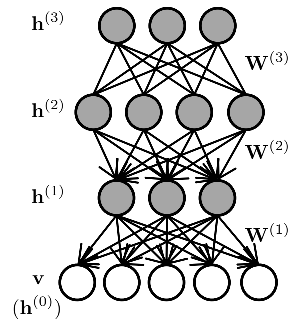
Input data

Sequences

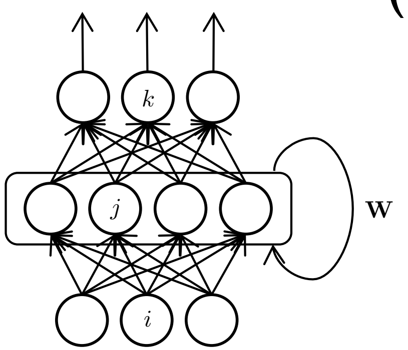
Self-attention nets
(Transformer)



Auto-encoders



Boltzmann machines



GANs

Variational AEs

Recurrent nets

Deterministic

Probabilistic