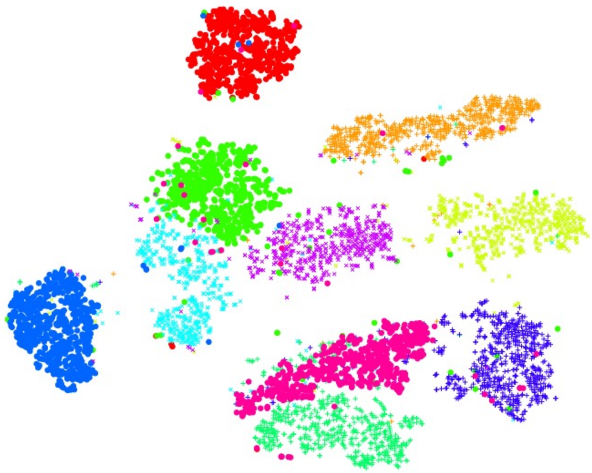
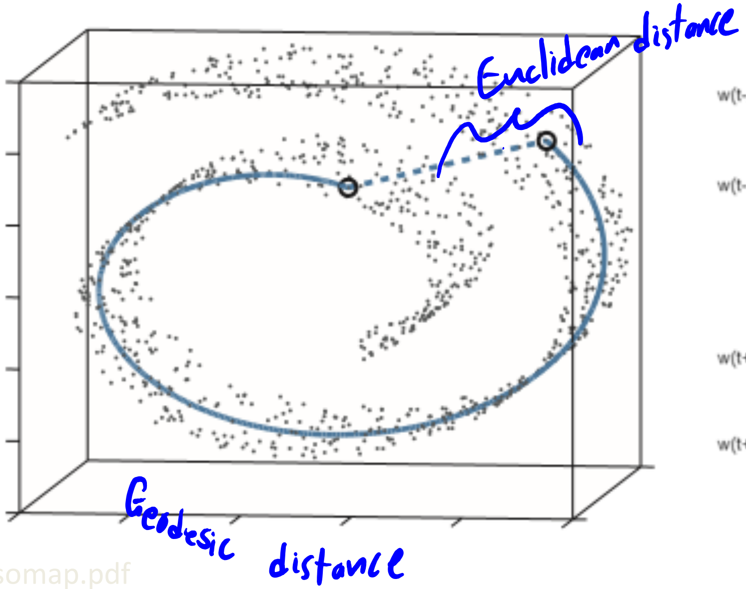
CPSC 340:
Machine Learning and Data Mining

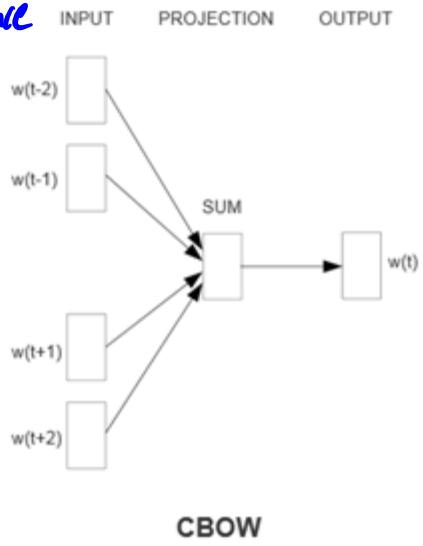
Neural Networks

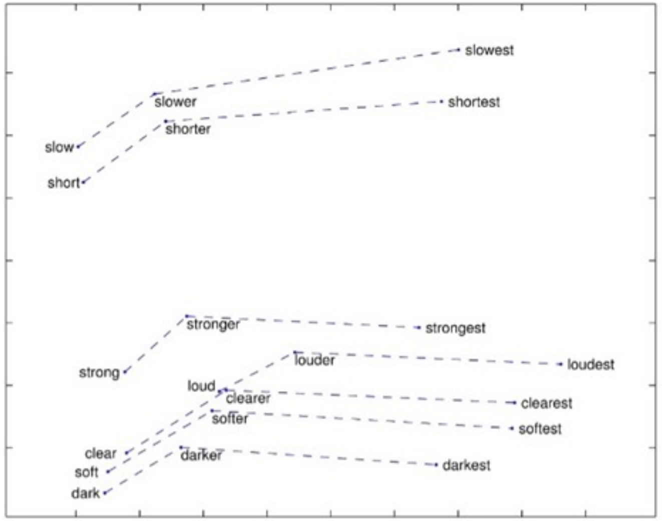
Last Time: Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Non-parametric latent-factor model: directly optimizes the z_i.
  - T-SNE tends to visualize clusters and manifold structures.
  - Word2vec gives continuous alternative to bag of words.









http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf
http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf
http://sebastianruder.com/secret-word2vec
http://sebastianruder.com/secret-word2vec

End of Part 4: Key Concepts

- We discussed linear latent-factor models:

$$f(W,Z) = \sum_{i=1}^{n} \sum_{j=1}^{d} (\langle w^j, z_i \rangle - x_{ij})^2$$

$$= \sum_{i=1}^{n} \| W^T z_i - x_i \|^2$$

$$= \| ZW - X \|_F^2$$

- Represent 'X' as linear combination of latent factors 'w_c'.
  - Latent features 'z_i' give a lower-dimensional version of each 'x_i'.
  - When k=1, finds direction that minimizes squared orthogonal distance.
- Applications:
  - Outlier detection, dimensionality reduction, data compression, features for linear models, visualization, factor discovery, filling in missing entries.

End of Part 4: Key Concepts

- Principal component analysis (PCA):
  - Often uses orthogonal factors and fits them sequentially (via SVD).
  - Or uses non-orthogonal factors and fits with SGD.
- Generalizations of PCA using ideas from linear models:
  - Binary PCA, robust PCA, regularized PCA, sparse PCA, NMF.
- Recommender systems:
  - "Content-based filtering" is usually supervised learning approach.
  - Collaborative-filtering only uses ratings.
- Matrix factorization approach to collaborative filtering.
  - Fits regularized PCA to available entries in matrix, to "fill in" other entries.

End of Part 4: Key Concepts

- We discussed multi-dimensional scaling (MDS):
  - Non-parametric method for high-dimensional data visualization.
  - Tries to match distance/similarity in high-/low-dimensions.
    - "Gradient descent on scatterplot points".
- Main challenge in MDS methods is "crowding" effect:
  - Methods focus on large distances and lose local structure.
- We discussed *t*-SNE:
  - MDS focusing on neighbour distances and not large distances.
- Word2vec is a recent MDS method giving better "word features".

Next Topic: Neural Networks

Neural Network History

- Popularity of neural networks has come in waves over the years.
  - Currently, it is one of the hottest topics in machine learning.
- Recent popularity due to unprecedented performance on some difficult tasks.
  - Natural language processing, speech recognition.
  - Computer vision.
- There are mainly due to big datasets, deep models, and tons of computation.
  - Plus more complex “architectures” (e.g. CNNs, LSTMs, ResNets, transformers).
- Lots of histories of the field online.

Neural Networks: Motivation

Counterpunch

Why Artificial Intelligence Must Be Stopped Now

Those advocating for artificial intelligence tout the huge benefits of using this technology. For instance, an article in CNN points out how...

38 minutes ago

InfoWorld

Microsoft introduces AI-powered UI controls for .NET

Currently experimental .NET Smart Components for Blazor, MVC, and Razor Pages bring Azure OpenAI intelligence to forms, menus,...

14 hours ago

PYMNTS.com

Meet the Titans: Major Players Funding the Future of AI

Saudi Arabia aims to carve out a leadership role in the burgeoning artificial intelligence (AI) field with a proposed $40 billion investment...

7 hours ago

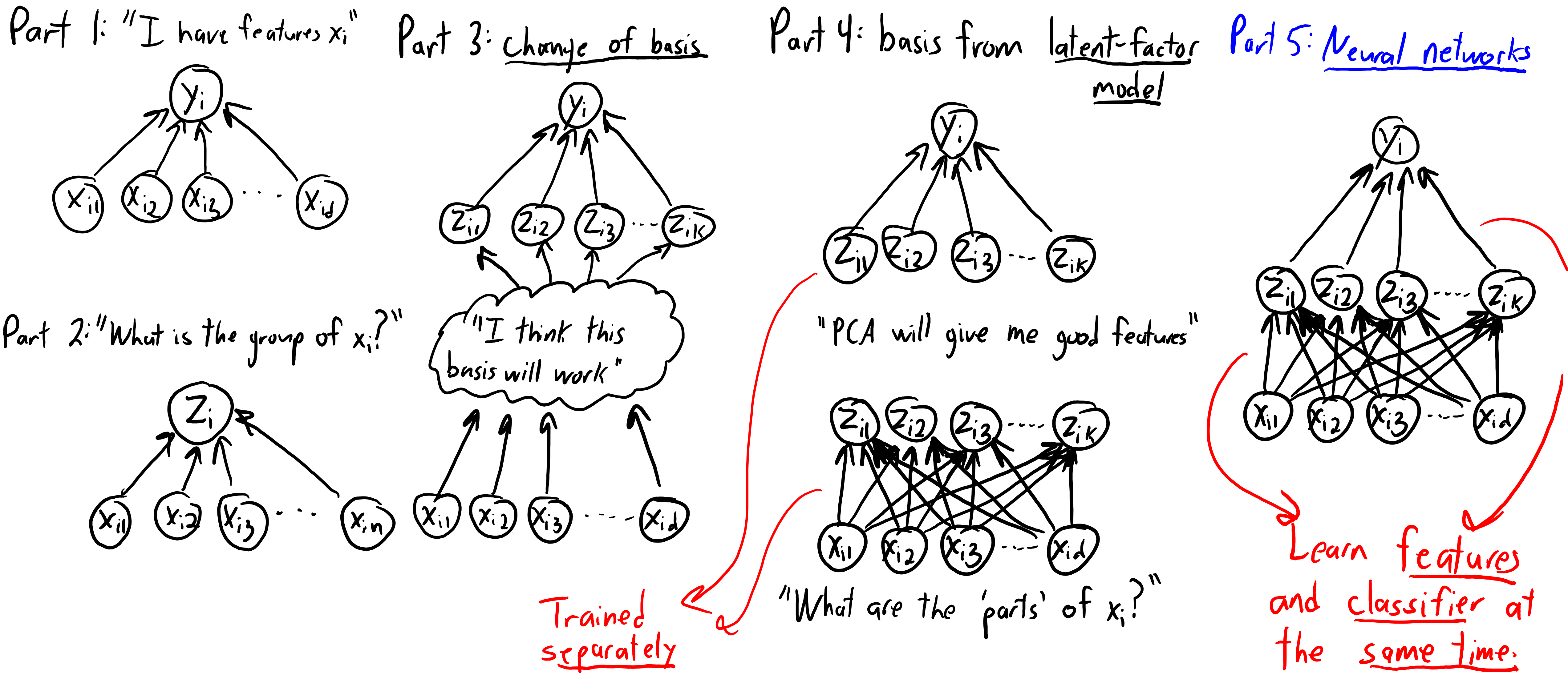
Neural Networks: Motivation

- Many domains require non-linear transforms of the features.
  - But, it may not be obvious which transform to use.

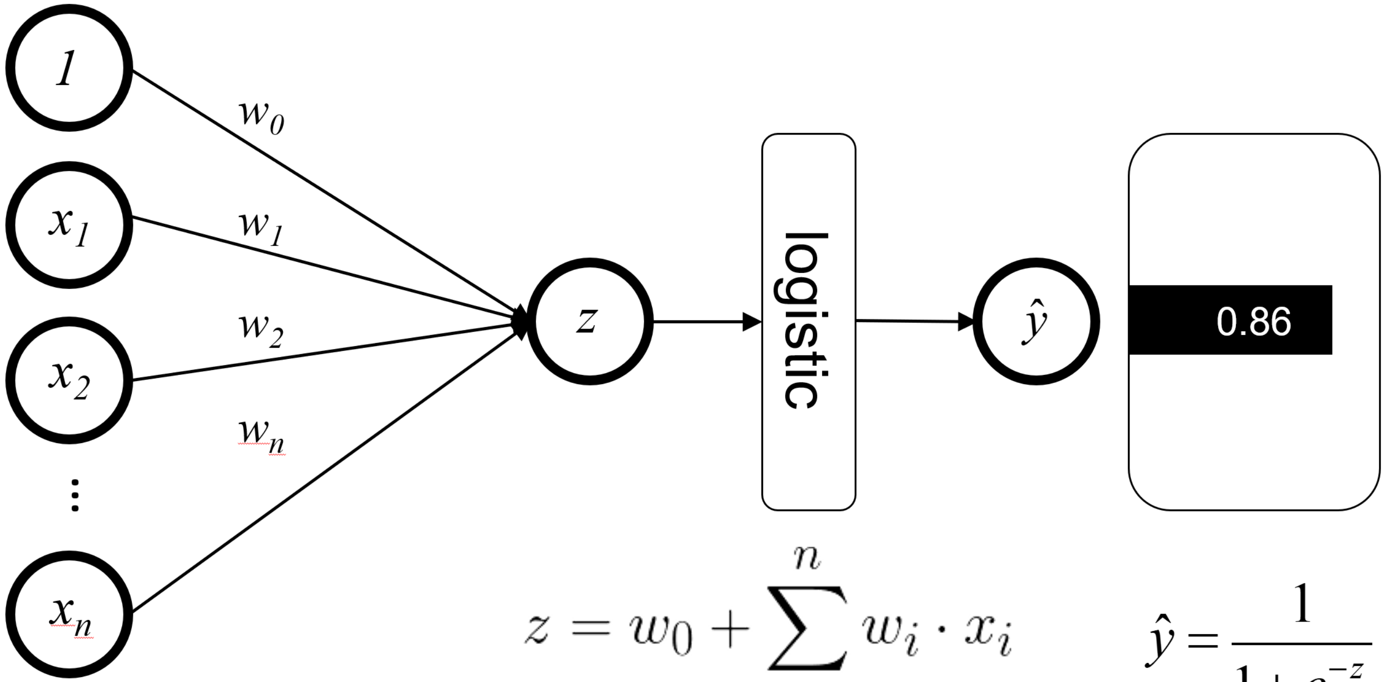
- Neural network models try to learn good transformations.
  - Optimize the “parameters of the features”.
    - And choose a class of features that help solve the classification/regression problems.

- We will first discuss the special case of “one hidden layer”.
  - Then we will move onto “deep learning” with uses multiple layers.

A Graphical Summary of CPSC 340 Parts 1-5



Review: Logistic Regression



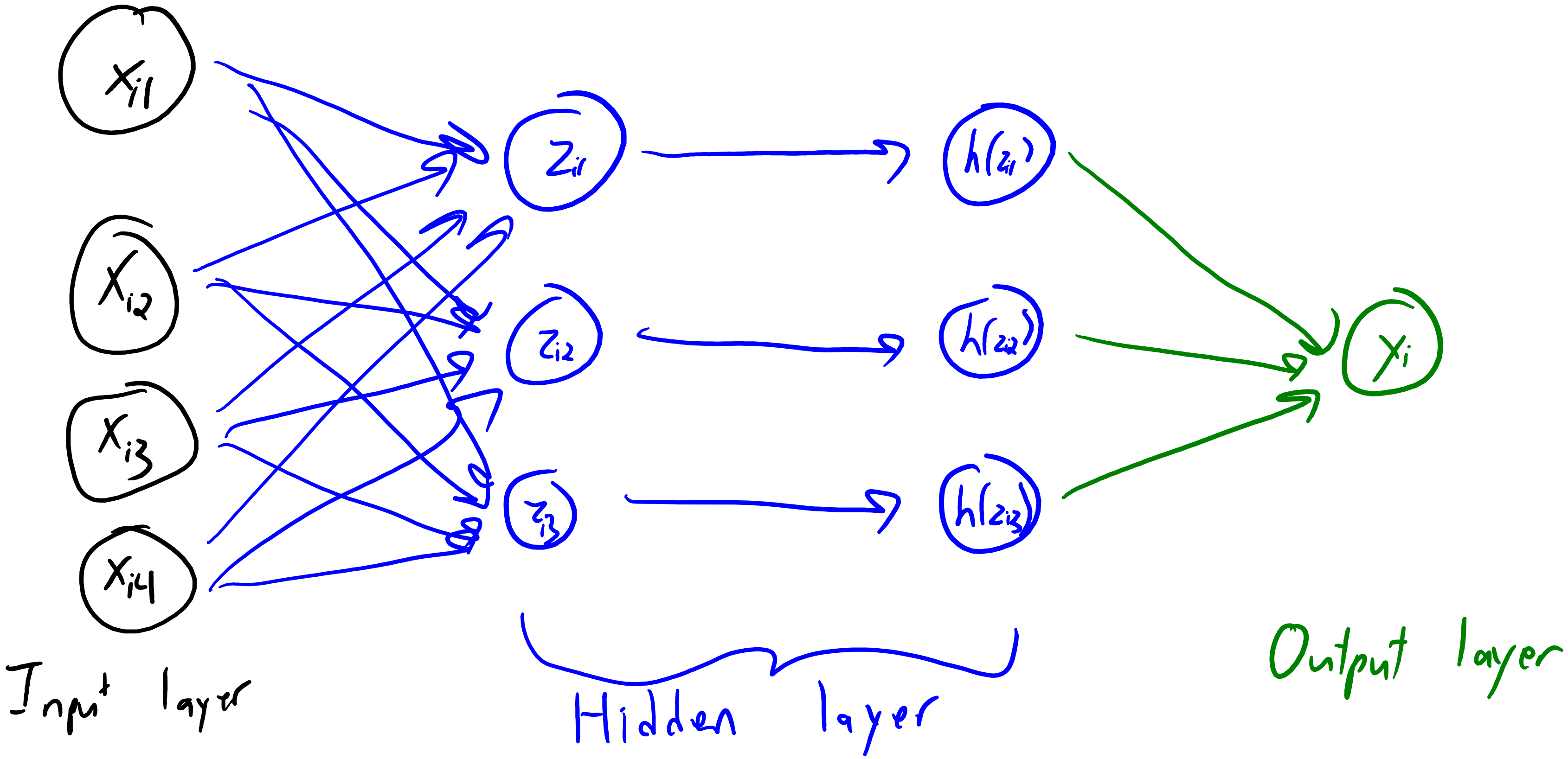
$$z = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$$

$$\hat{y} = \frac{1}{1+e^{-z}}$$

https://carpentries-incubator.github.io/ml4bio-workshop/05-logit-ann/index.html

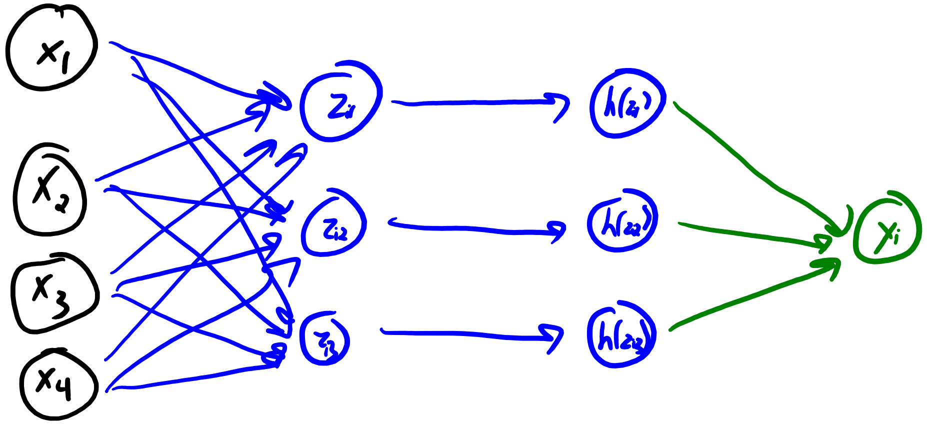
Neural Network with One Hidden Layer

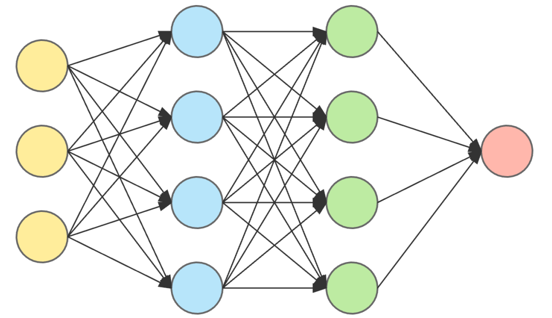
- Classic neural network structure with one hidden layer:



Neural Network with One Hidden Layer

- As a picture:





- As a function:

$$\hat{y}_i = v^T h(\underbrace{W \underbrace{x_i}_{\text{input}}}_{\text{"}z_i\text{": linear combination of input}})$$

Linear combination of "activations"

Non-linear transformation of each z_{ic}

i^{th} training example
c^{th} z value

$$Z = \begin{bmatrix} — & z_1^T & — \\ — & z_2^T & — \\ & \vdots & \\ — & z_n^T & — \end{bmatrix}$$

$n \times K$

Neural Network with One Hidden Layer

- As a function:

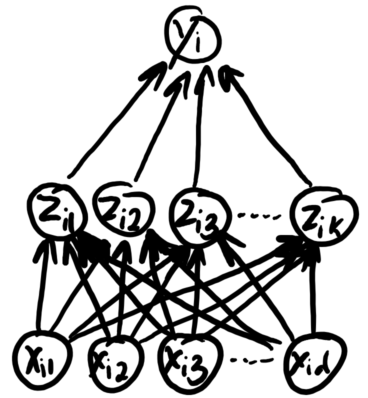
$$\hat{y}_i = v^T h(\underbrace{W \underbrace{x_i}_{\text{input}}}_{\text{"}z_i\text{": linear combination of input}})$$

Linear combination of "activations"

Non-linear transformation of each z_{ic}

- Parameters: the "k times d" matrix 'W', and length-k vector "v".
  - Using 'k' as number of "hidden units", the dimension are:

$$W = \underbrace{\begin{bmatrix} \text{---} & w_1^T & \text{---} \\ \text{---} & w_2^T & \text{---} \\ & \vdots & \\ \text{---} & w_k^T & \text{---} \end{bmatrix}}_{k \times d} \qquad v = \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}}_{k \times 1}$$



Neural Network with One Hidden Layer

- As a function:

$$\hat{y}_i = v^T h(\underbrace{W \underbrace{x_i}_{\text{input}}}_{\text{"}z_i\text{": linear combination of input}})$$

Linear combination of "activations"

Non-linear transformation of each z_{ic}

- Linear transformation $z_i = Wx_i$ is like doing PCA.
  - Mixes together the features in a way that we learn.
- Non-linear transform 'h' is often sigmoid applied element-wise.
  - Without a non-linear transformation it degenerates to a linear model:
    - $\hat{y}_i = v^T(Wx_i) = (v^TW)x_i = w^Tx_i$ (if we set 'w' using $w=W^Tv$).

Neural Network with One Hidden Layer

- As a function:

$$\hat{y}_i = v^T h(W \underbrace{x_i}_{\text{input}})$$

Linear combination of "activations"

Non-linear transformation of each z_{ic}

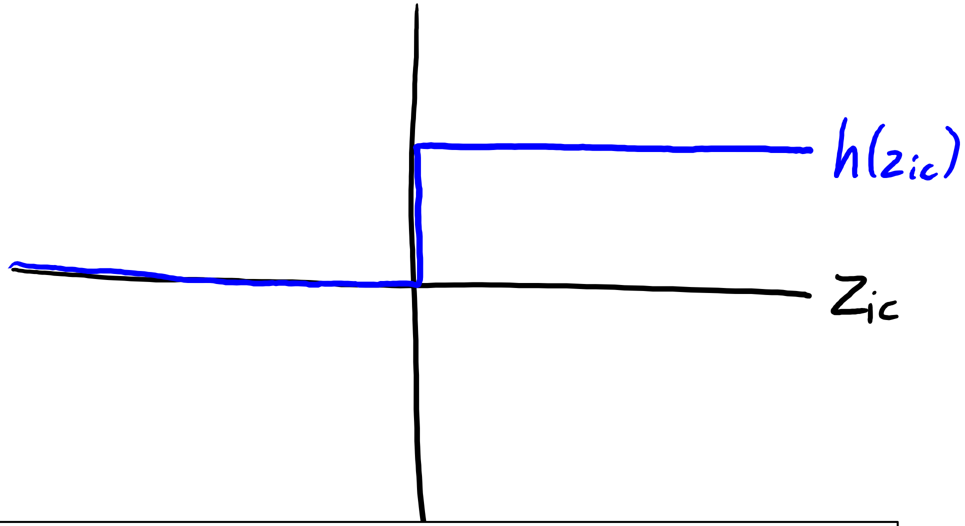
"z_i": linear combination of input

- Second linear transformation $v^T h(z_i)$ gives final value.
  – This is like using a linear model with non-linear feature transformations.
    - But in this case we learned the features.
- Cost of computing $\hat{y}_i$ above is O(kd).
  – O(kd) to compute Wx_i, O(k) to apply 'h', then O(k) to multiply by 'v'.

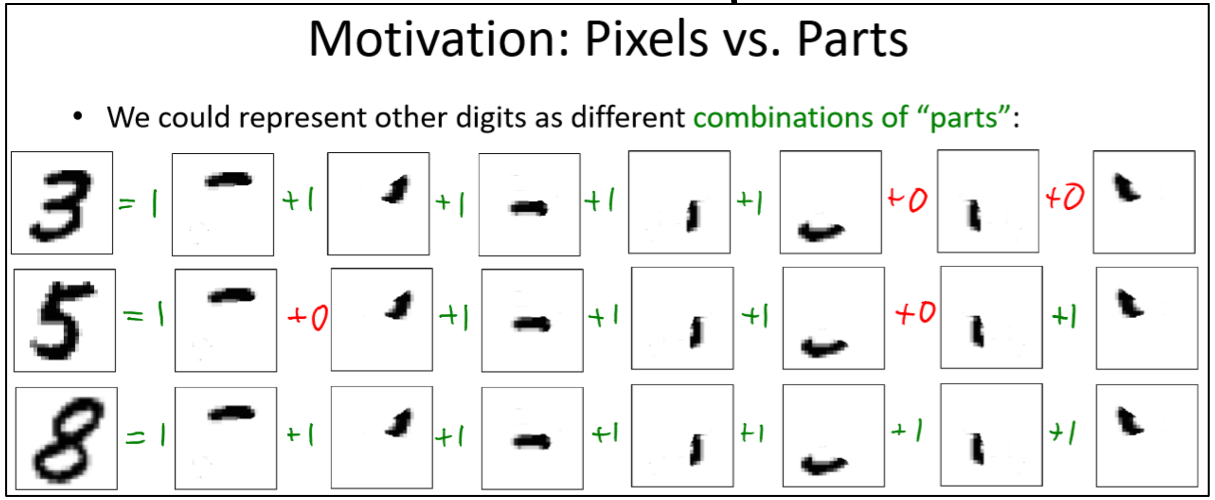
Why Sigmoid as Non-Linear Transform?

- Consider setting 'h' to define binary features z_i using:

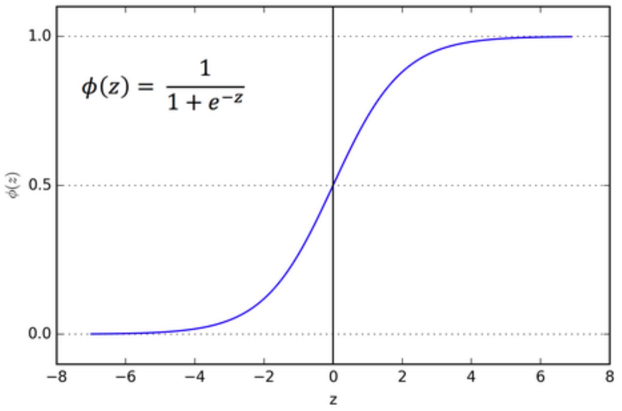
$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



  - Each $h(z_i)$ can be viewed as binary feature.
    - "You either have this 'part' or you don't have it."
  - We can make 2^k objects by all the possible "part combinations".

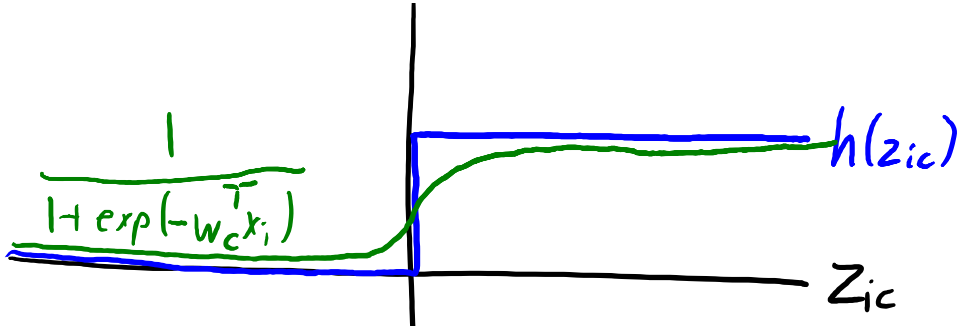


Why Sigmoid as Non-Linear Transform?



- Consider setting ‘h’ to define binary features z_i using

$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



  – Each $h(z_i)$ can be viewed as binary feature.
    - “You either have this ‘part’ or you don’t have it.”
  – But this is hard to optimize (non-differentiable/discontinuous).
- Sigmoid is a smooth approximation to these binary features.
  – Allows you to train the model using gradient descent or SGD.

Universal Approximation with One Hidden Layer

- Classic choice of "activation" function 'h' is the sigmoid function.
- With enough hidden "units", this is a "universal approximator".
  - Any continuous function can be approximated arbitrarily well (on bounded domain).

- But this result is for a non-parametric setting of the parameters:
  - The number of hidden "units" must be a function of 'n'.
  - A fixed-size network is not a universal approximator.

- Other universal approximators (always non-parametric):
  - K-nearest neighbours.
    - Need to have 'k' depending on 'n'.
  - Linear models with polynomial non-linear features transformations.
    - Degree of polynomial depends on 'n'.
  - Linear models with Gaussian RBFs as non-linear features transformations or kernels.
    - With RBF centered on each x^i.

Adding Bias Variables

- Recall fitting linear models with a bias variable (so $\hat{y}_i \neq 0$ when x_i=0).

$$\hat{y}_i = \sum_{j=1}^{d} w_j x_{ij} + \beta$$

  - We often implement this by adding a column of ones to X.
- In neural networks we often include biases on each z_{ic}:

$$\hat{y}_i = \sum_{c=1}^{k} v_c h(w_c^T x_i + \beta_c)$$

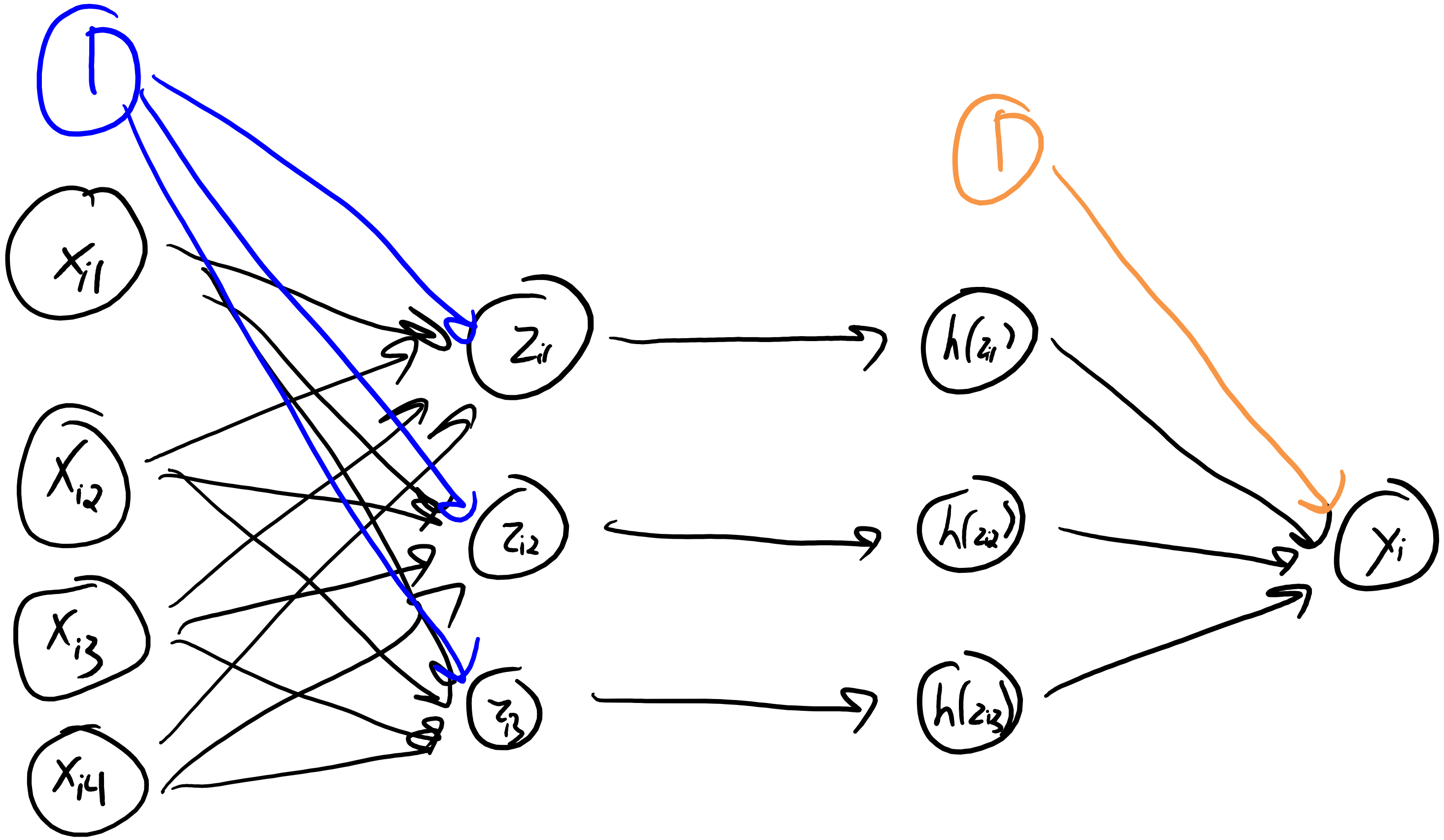
  - As before, we could implement this by adding a column of ones to X.
- We often also want a bias on the output:

$$\hat{y}_i = \sum_{c=1}^{k} v_c h(w_c^T x_i + \beta_c) + \beta$$

  - For sigmoid 'h', you could equivalently fix one row of W to be 0.
    - Since h(0) is a constant.

Adding Bias Variables

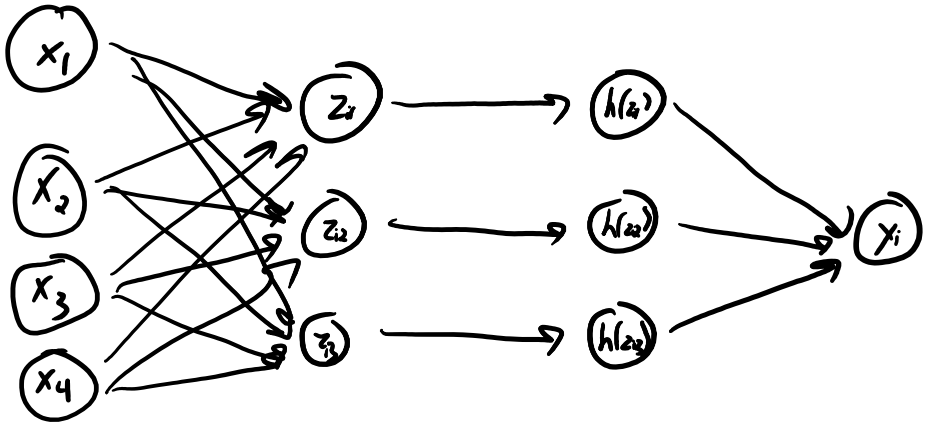
$$\hat{y}_i = \sum_{c=1}^{k} v_c h(w_c^T x_i + \beta_c) + \beta$$



Regression vs. Binary Classification

- For regression problems, our prediction (ignoring biases) is:

$$\hat{y}_i = v^T h(W x_i)$$



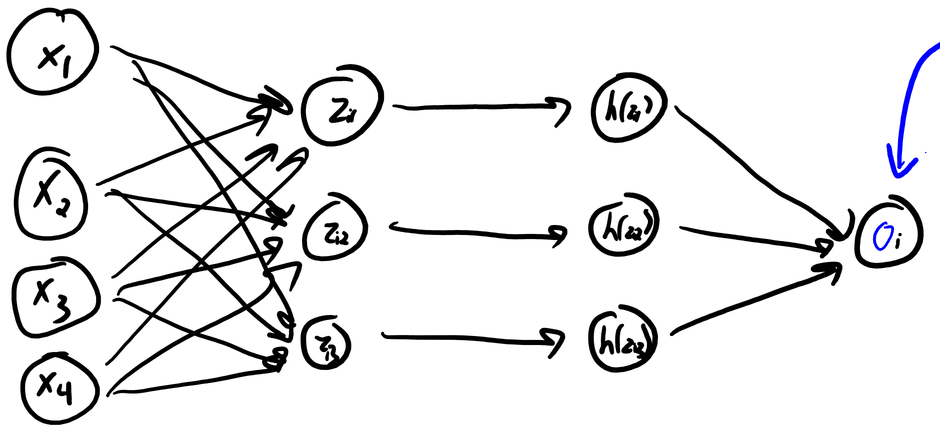
- And we might train to minimize the squared residual:

$$f(W,v) = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(v^T h(Wx_i) - y_i)^2$$

Regression vs. Binary Classification

- For binary classification problems, our prediction is:

$$o_i = v^T h(Wx_i) \qquad \hat{y}_i = \text{sign}(o_i)$$



output unit here rather than prediction of label

- And we might train to minimize the logistic loss:

$$f(W,v) = \sum_{i=1}^n \log(1+\exp(-y_i o_i)) = \sum_{i=1}^n \log(1+\exp(-y_i v^T h(Wx_i)))$$

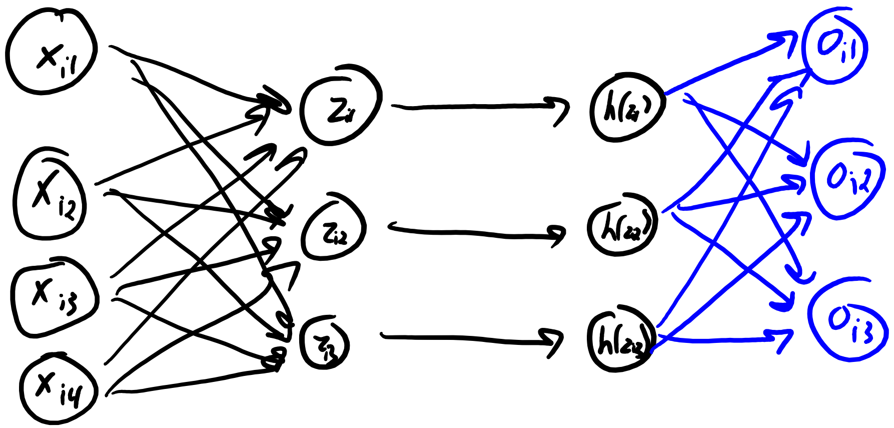
  – This is like logistic regression with learned features.

$$p(y_i \mid W, v, x_i) = \frac{1}{1+\exp(-y_i \underbrace{v^T h(Wx_i)}_{o_i})}$$

Use a sigmoid on output to get a probability

Neural Network for Multi-Class Classification

- Multi-class classification a neural network:
  - Input is connected to a hidden layer (same as regression and binary case).
  - Hidden layer is connected to multiple output units (one for each label.).



$$o_{i1} = v_1^T h(Wx_i)$$
$$o_{i2} = v_2^T h(Wx_i)$$
$$o_{i3} = v_3^T h(Wx_i)$$

Now have a matrix of parameters:

$$V = \begin{bmatrix} — & v_1^T & — \\ — & v_2^T & — \\ & \vdots & \\ — & v_{k'}^T & — \end{bmatrix}$$

$\ell \times k$

ℓ: number of classes

k: number of hidden units

  - We can predict by maximizing o_{ic} over all 'c'.
  - We can convert to probabilities for each class using softmax to the o_{ic} values:

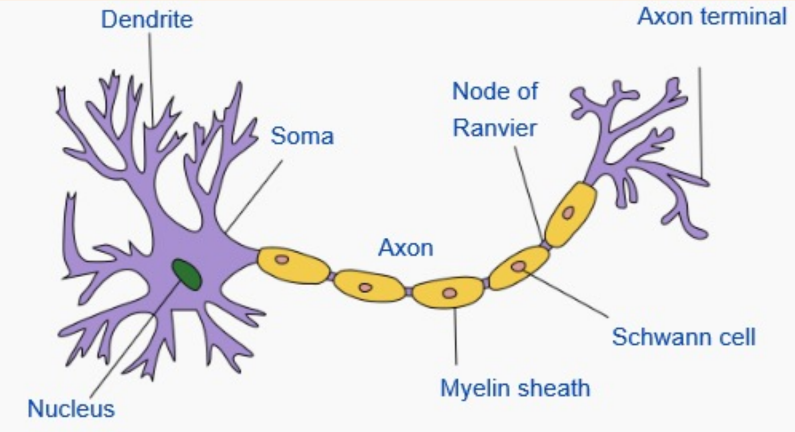
$$\frac{\exp(o_{ic})}{\sum_{c'=1}^{k'} \exp(o_{ic'})}$$

  - We train by minimizing negative log of this probability (softmax loss, summed across examples).
  - Notice that we changed tasks by only changing last layer (and loss function).

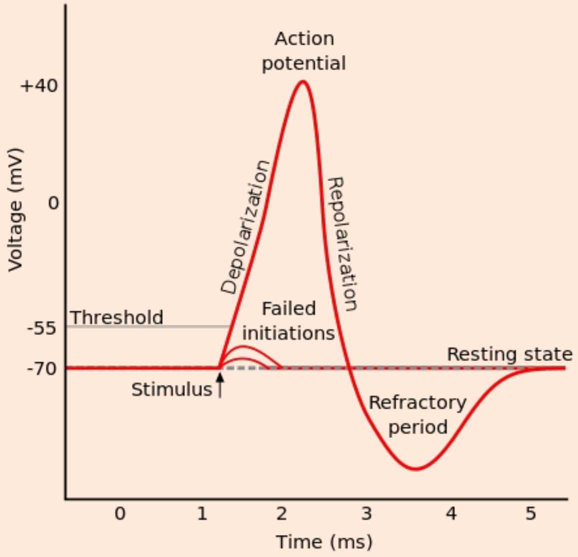
Next Topic: Biological Motivation

Why “Neural Network”?

- Cartoon of “typical” neuron:

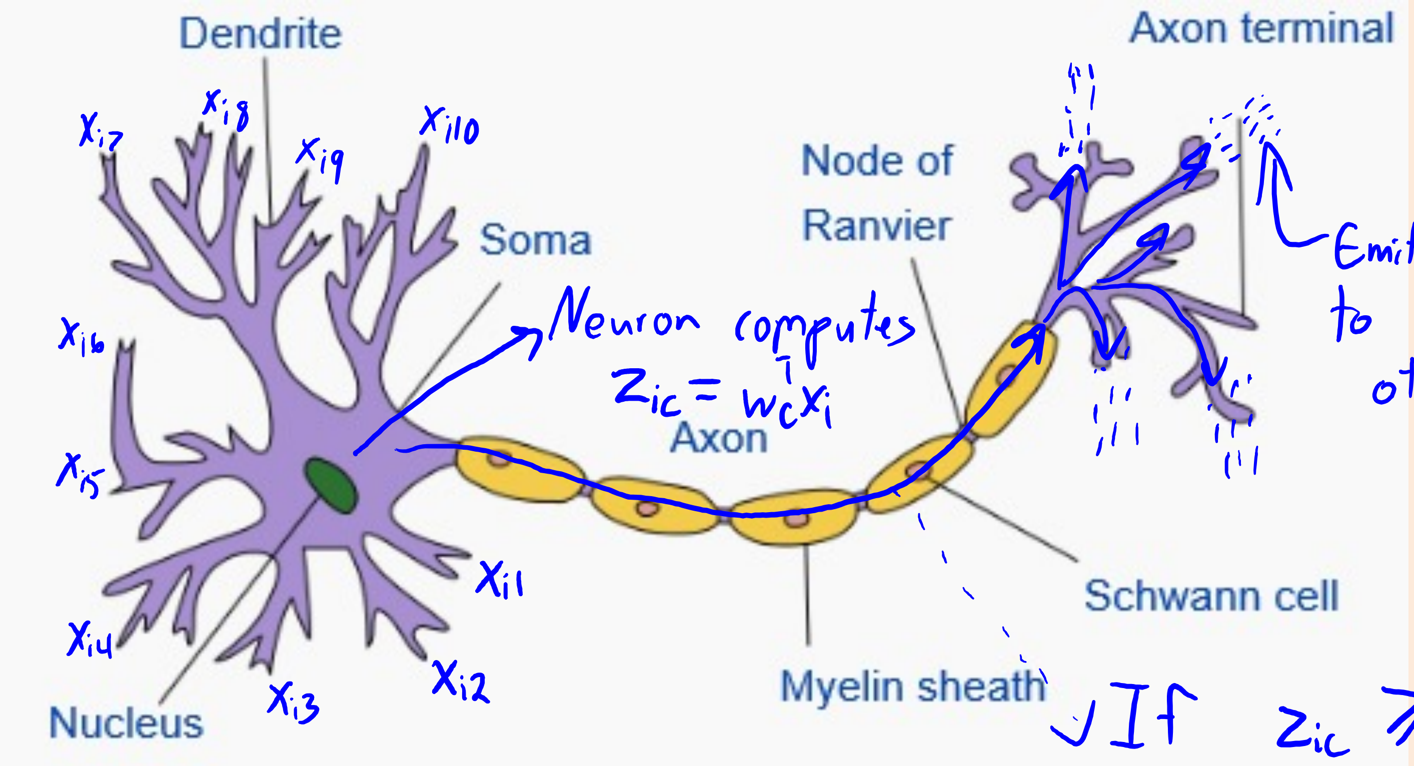


- Neuron has many “dendrites”, which take an input signal.
- Neuron has a single “axon”, which sends an output signal.
- With the right input to dendrites:
  - “Action potential” along axon (like a binary signal):



https://en.wikipedia.org/wiki/Neuron
https://en.wikipedia.org/wiki/Action_potential

Why “Neural Network”?



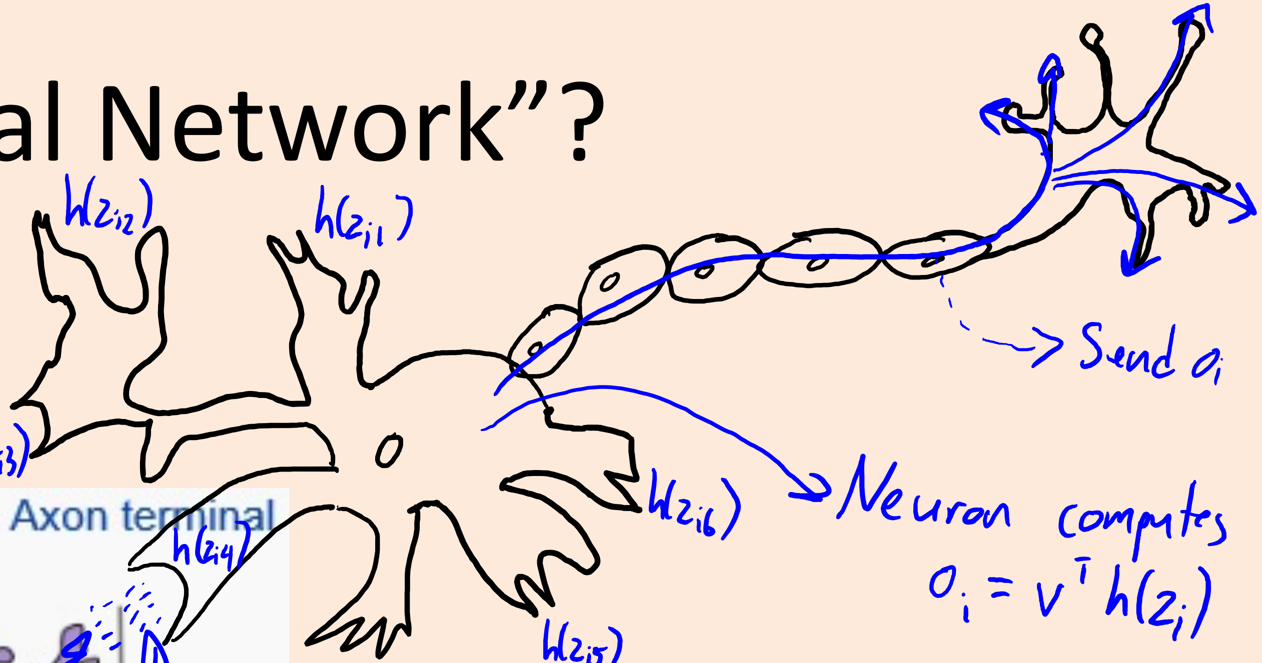
Emits "neurotransmitter" to send signal to other neurons.

If $z_{ic} \geq 0$ neuron sends signal along axon.

We approximate binary signal with $\frac{1}{1+\exp(-z_{ic})}$

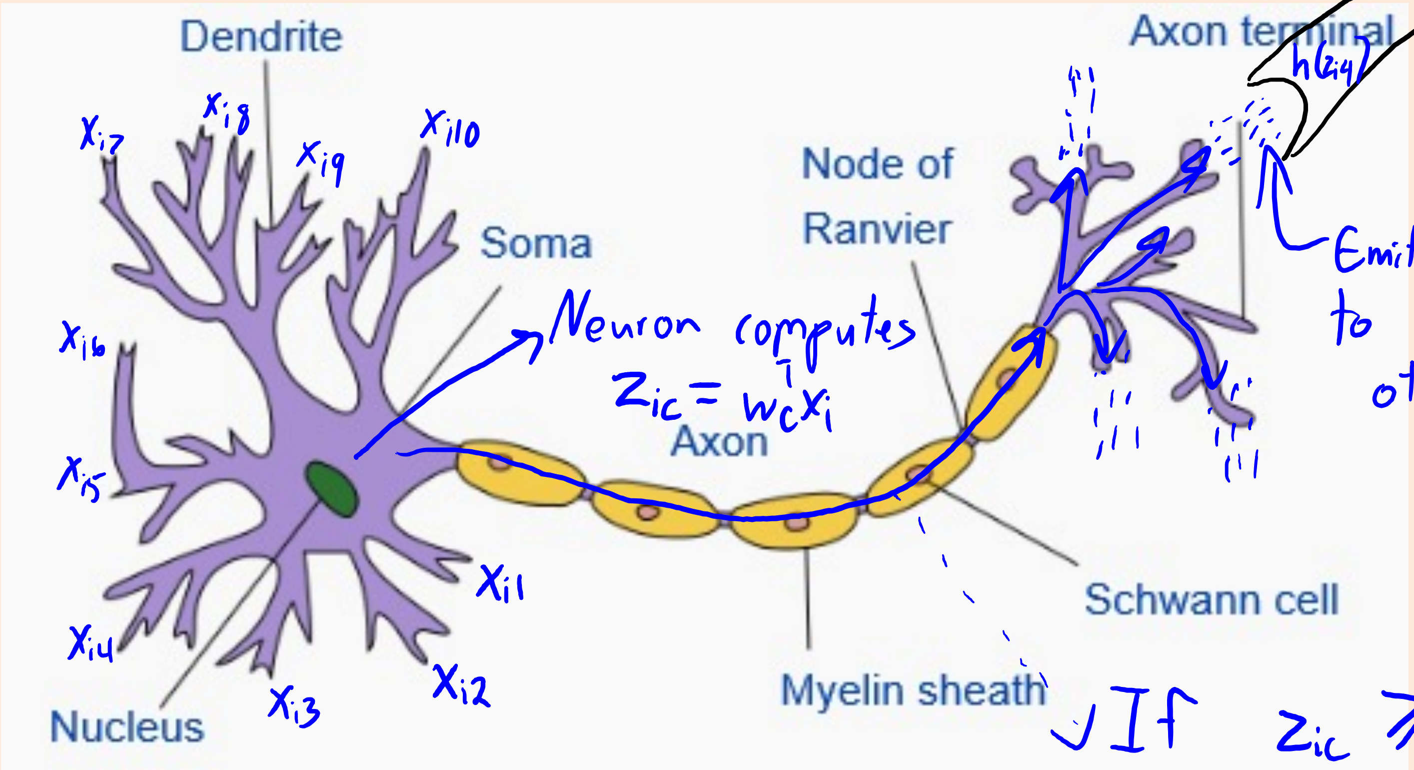
https://en.wikipedia.org/wiki/Neuron

Why “Neural Network”?



Send o_i

Neuron computes $o_i = v^T h(z_i)$



Neuron computes $z_{ic} = w_c^T x_i$

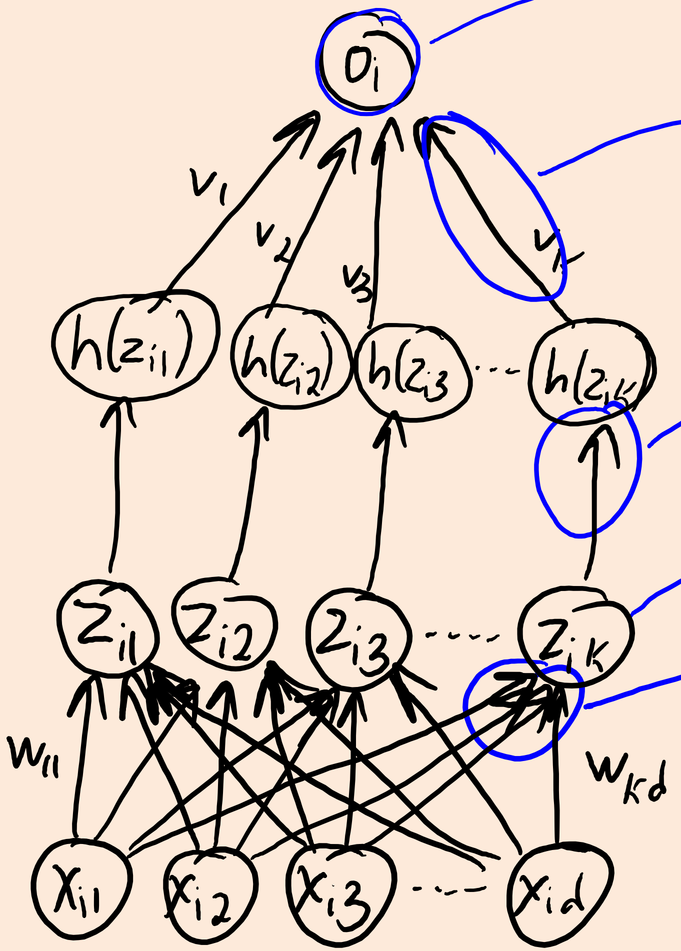
Emits "neurotransmitter" to send signal to other neurons.

If $z_{ic} \geq 0$ neuron sends signal along axon.

We approximate binary signal with $\frac{1}{1+\exp(-z_{ic})}$

https://en.wikipedia.org/wiki/Neuron

Why "Neural Network"?



Predictions based on aggregation at o_i "neuron"

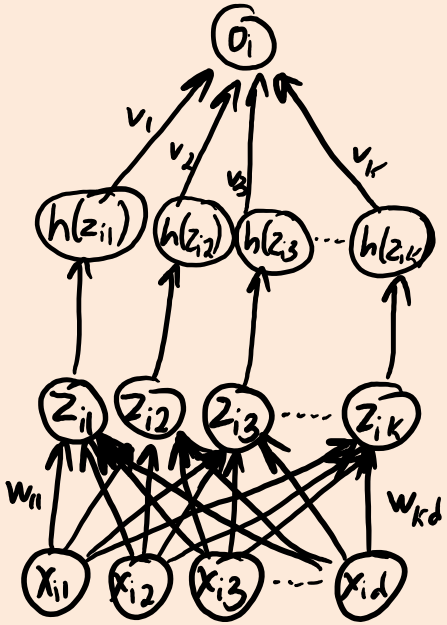
Synapse between z_{ik} and o_i "neuron"

binary signal $h(w_c^T x_i)$ sent along "axon"

Neuron aggregates signals: $w_c^T x_i$

"dendrites" for z_{ik} "neuron" are reciving x_{ij} values

"Artificial" Neural Nets vs. "Real" Networks Nets



- Artificial neural network:
  - x_i is measurement of the world.
  - z_i is internal representation of world.
  - o_i is output of neuron for classification/regression.
- Real neural networks are more complicated:
  - Timing of action potentials seems to be important.
    - "Rate coding": frequency of action potentials simulates continuous output.
  - Neural networks don't reflect sparsity of action potentials.
  - How much computation is done inside neuron?
  - Brain is highly organized (e.g., substructures and cortical columns).
  - Connection structure changes.
  - Different types of neurotransmitters.

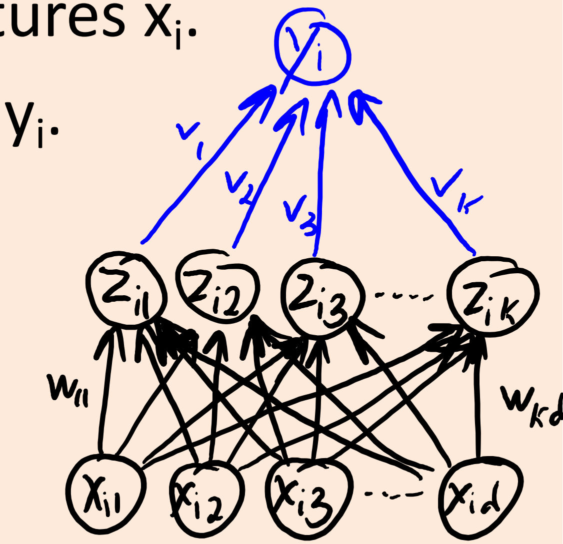
Summary

- Unprecedented performance on difficult pattern recognition tasks.
- Neural networks with one hidden layer:
  - Simultaneous learn a linear model and its features z_i.
- Non-linear transform avoids degeneracy.
  - Universal approximator if size of layer grows with number of examples 'n'.
- Bias variables added to each layer.
- Outputting probabilities and training with SGD.
- Biological motivation for (deep) neural networks.

- Next time: neural networks overfit less with more parameters?

Supervised Learning Roadmap

- Part 1: "Direct" Supervised Learning.
  - We learned parameters 'w' based on the original features x_i and target y_i.
- Part 3: Change of Basis.
  - We learned parameters 'v' based on a change of basis z_i and target y_i.
- Part 4: Latent-Factor Models.
  - We learned parameters 'W' for basis z_i based on only on features x_i.
  - You can then learn 'v' based on change of basis z_i and target y_i.
- Part 5: Neural Networks (one hidden layer).
  - Jointly learn 'W' and 'v' based on x_i and y_i.
  - **Learn basis z_i that is good for supervised learning.**



Why $z_i = Wx_i$?

- In PCA we had that the optimal $Z = XW^T(WW^T)^{-1}$.
- If W had normalized+orthogonal rows, $Z = XW^T$ (since $WW^T = I$).
  - So $z_i = Wx_i$ in this normalized+orthogonal case.

- Why we would use $z_i = Wx_i$ in neural networks?
  - We didn't enforce normalization or orthogonality.

- Well, the value $W^T(WW^T)^{-1}$ is just "some matrix".
  - You can think of neural networks as just directly learning this matrix.

Softmax NLL vs. Cross-Entropy

- Multi-class objective often written as minimizing cross-entropy:

$$f(W,V) = \sum_{i=1}^{n} \sum_{j=1}^{\ell} I[y^i = c](-\log p(y^i = c \mid X, W, V))$$

- The indicator function is zero except for true label y^i:

$$f(W,V) = -\sum_{i=1}^{n} \log p(y^i \mid X, W, V)$$

- When we plug in the softmax likelihood, we get the softmax NLL.
  - So cross-entropy is the softmax NLL with extra terms that do nothing.
    - Cross-entropy way of writing would make more sense if training data had “soft” assignments to classes.