Test (28+2 pts)

- Open book, open notes
- 06/24 (Thursday), 11:59AM – 11:59PM (noon to Midnight), Eastern Time (12 hours)
  - No class on 06/24
- Questions will be published through Canvas
- Solutions must be submitted through Canvas by 11:59PM (06/24)
  - Submission link closes on 12:00AM (06/25)
  - No email submission
  - No exception, unless showing documents (medical etc.)

Support Vector Machines and Deep Learning

CAP 5615 Intro. to NN
2021 Summer

Xingquan Zhu

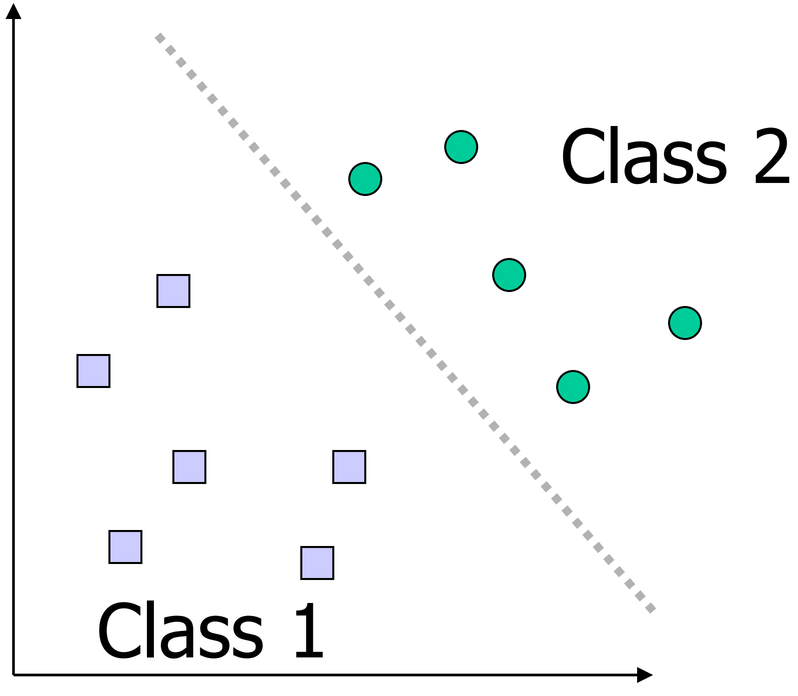
Outline

- Maximum Margin Classifier
- Deep Neural Networks
  - Convolutional Neural Network
  - Stacked AutoEncoder
  - Deep Belief Networks (Restrictive Boltzmann Machines)

What is a Good Decision Boundary?

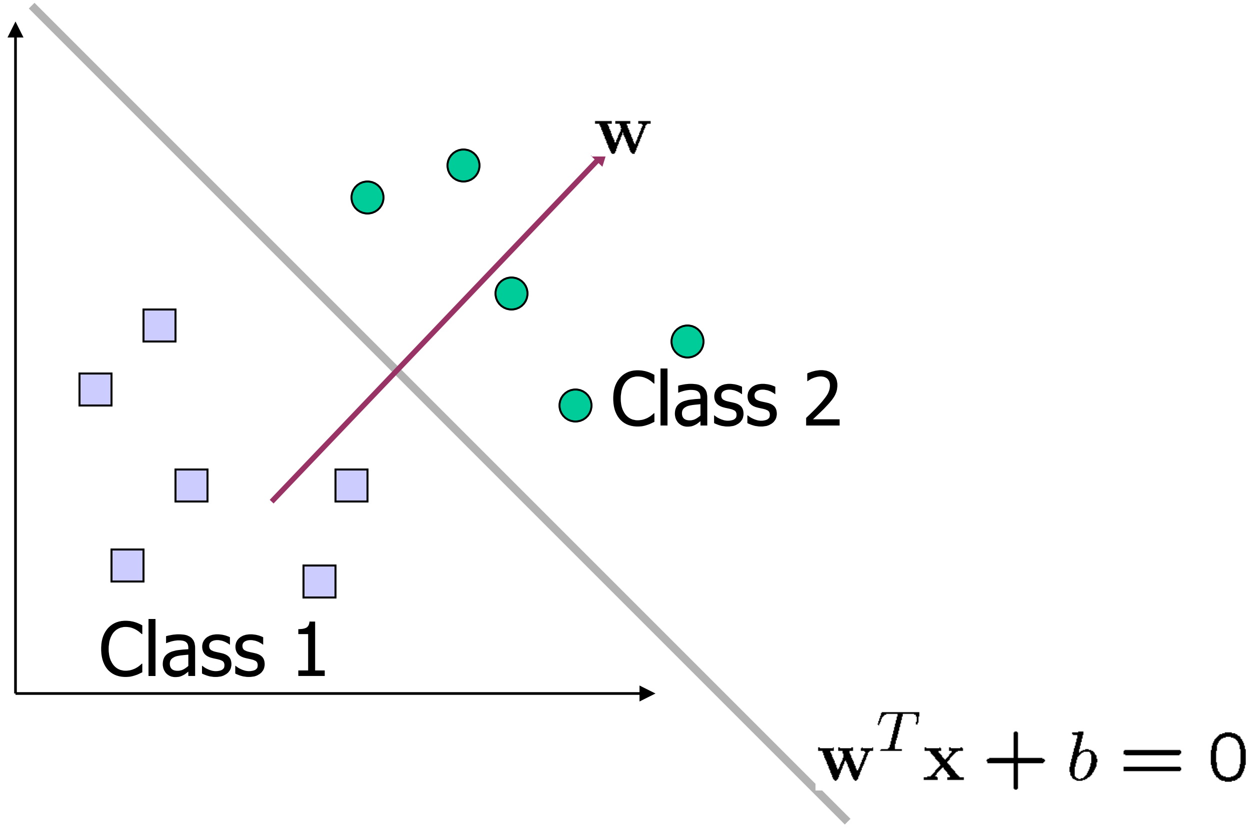
- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
  - The Perceptron algorithm can be used to find such a boundary
- Are all decision boundaries equally good?

$$\mathbf{w}^T\mathbf{x} + b = 0$$

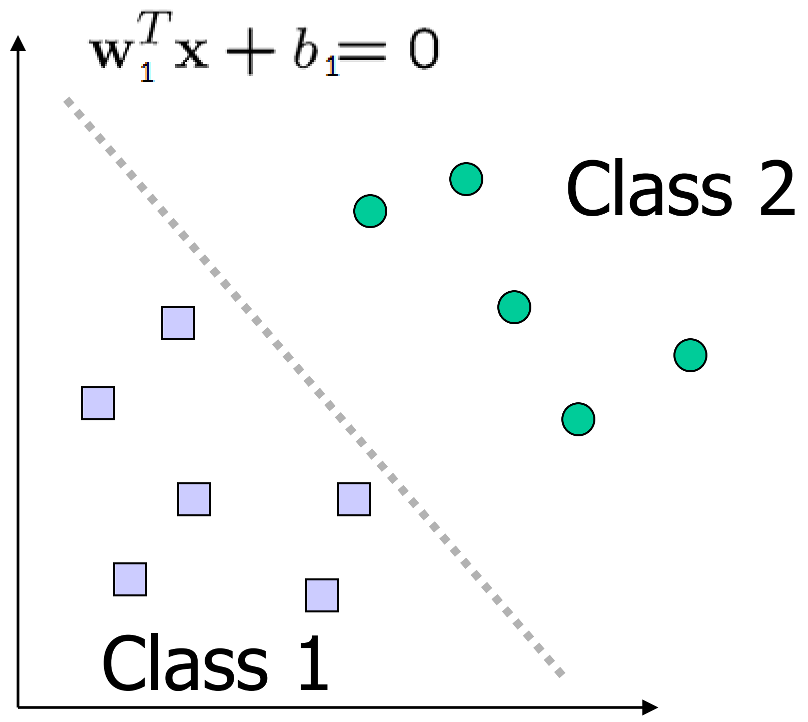


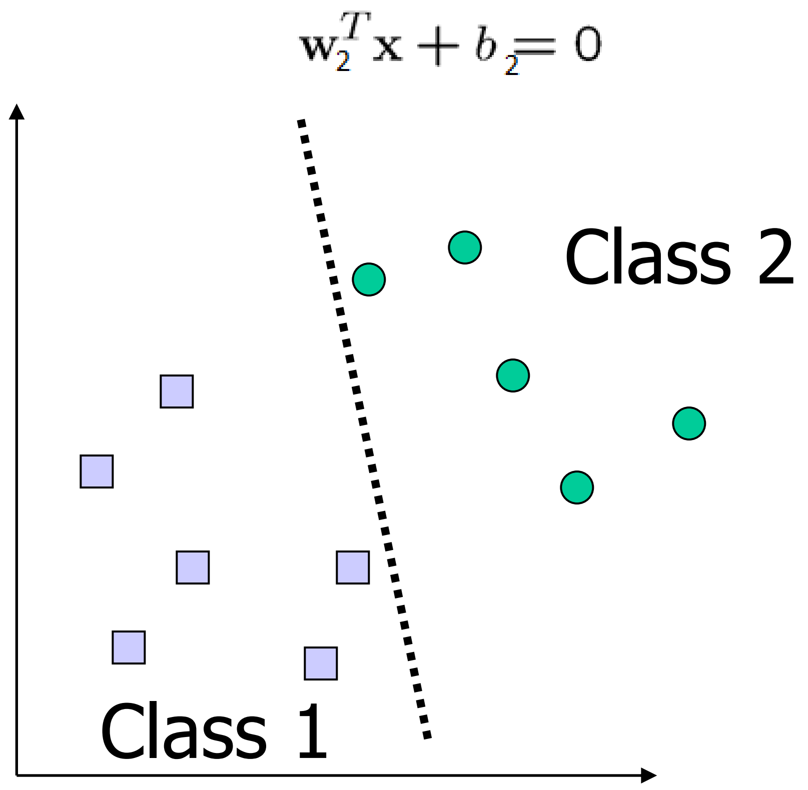
Where is W?

The vector **w** is perpendicular to the decision boundary, why?



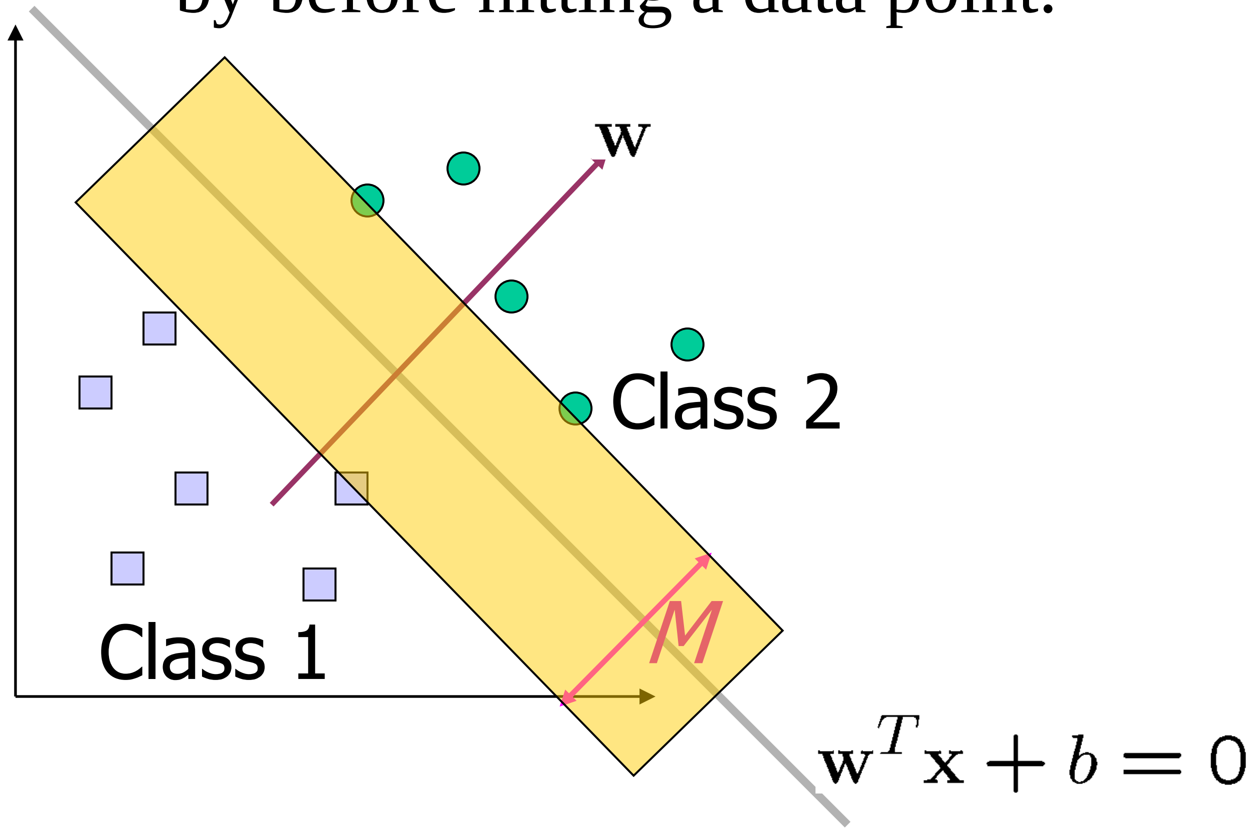
Other Decision Boundaries





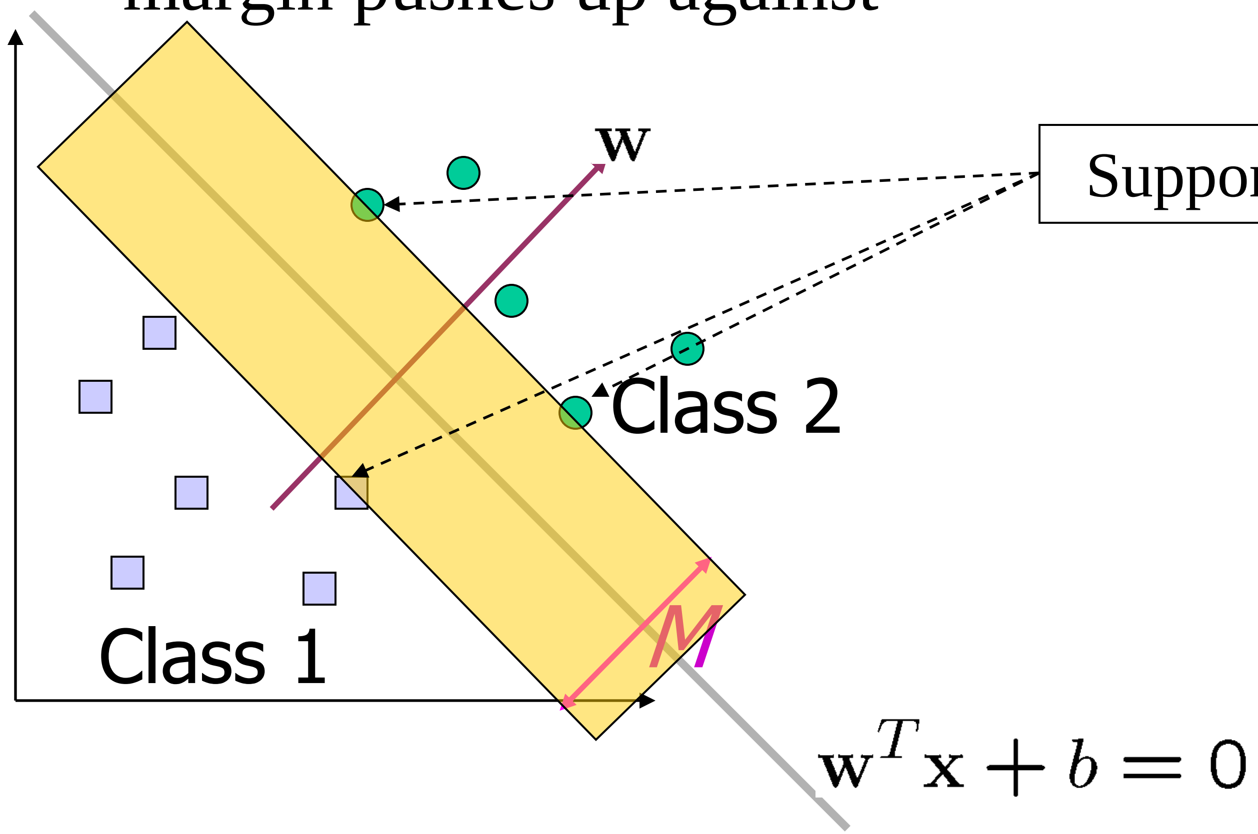
Classifier Margin

- The margin of a linear classifier (M)
  - The width that the boundary could be increased by before hitting a data point.



The Maximum Margin Classifier

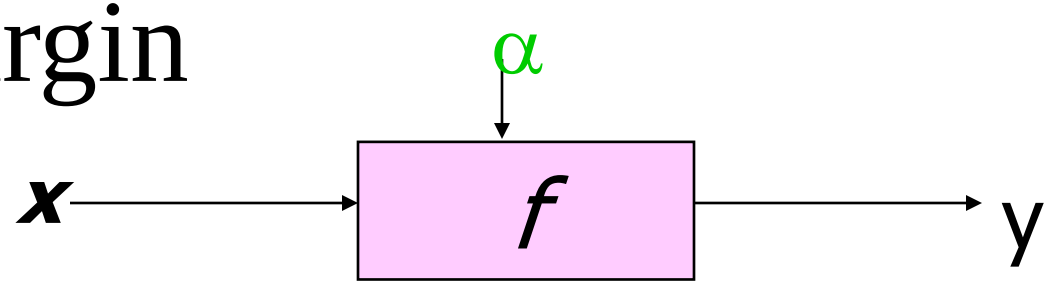
- The maximum margin linear classifier is the linear classifier with the maximum margin M
- The support vectors are those data points that the margin pushes up against



Why We Bother to Maximize the Margin?

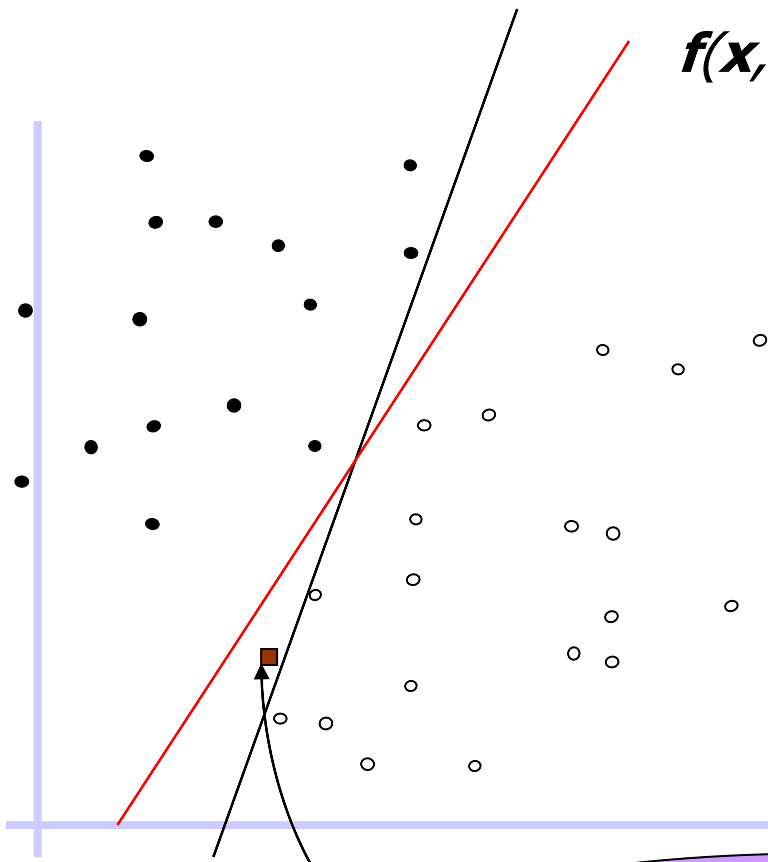
- Intuitively, a linear classifier with maximum margin is probably the safest one
  - If we've made a small error in the location of the boundary this gives us least chance of causing a misclassification
- Practically, only support vectors are important; other training examples are ignorable.
- Empirically it works very very well.

Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

- • denotes +1
- ◦ denotes -1

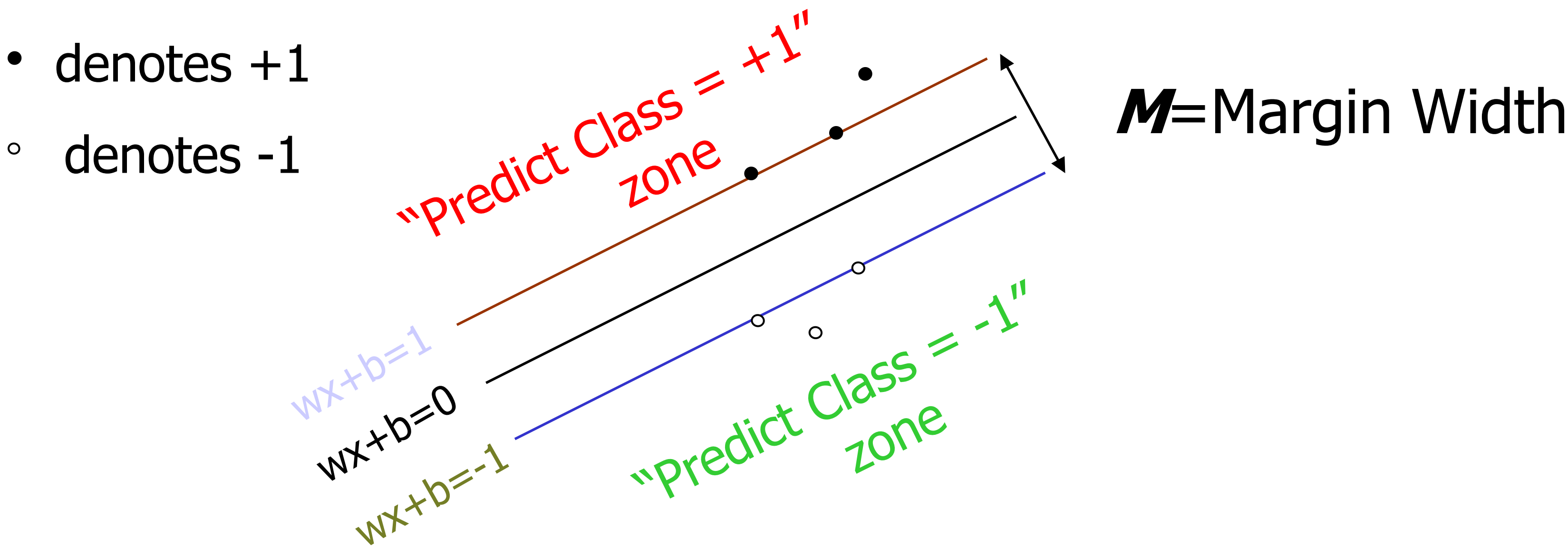


How would you classify this data?

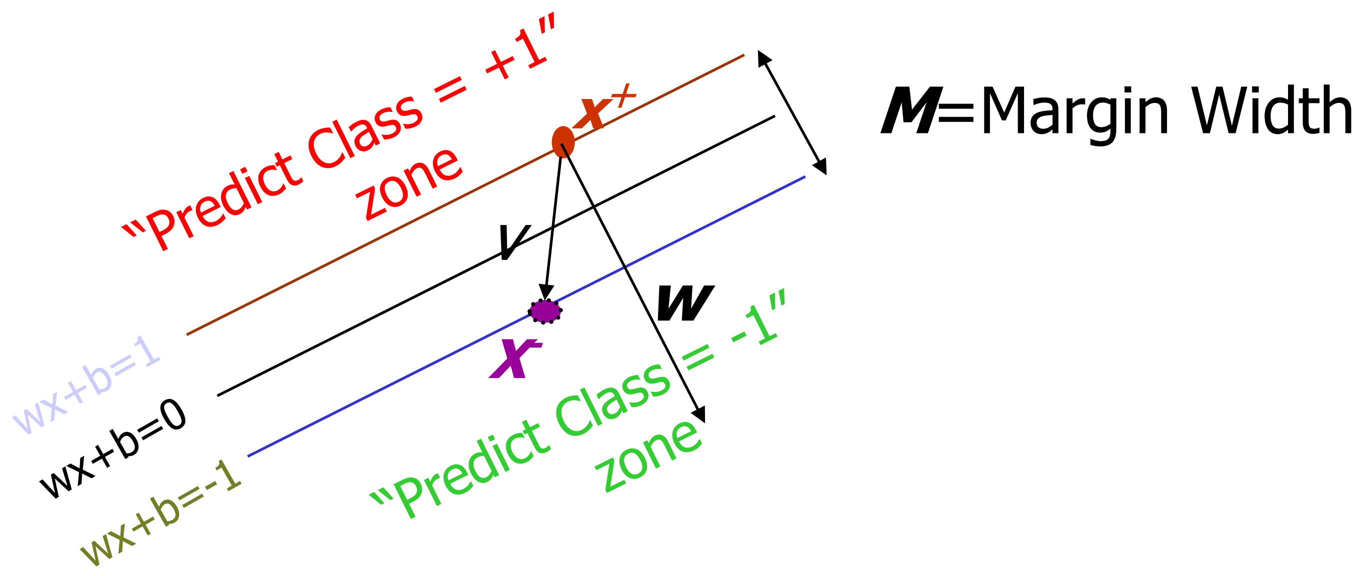
The same instance can be misclassified by an inferior decision boundary

Then How to Maximize M?

- We have to represent the margin mathematically.
  - Plus-plane = { **x** : **w** . **x** + b = +1 }
  - Minus-plane = { **x** : **w** . **x** + b = -1 }



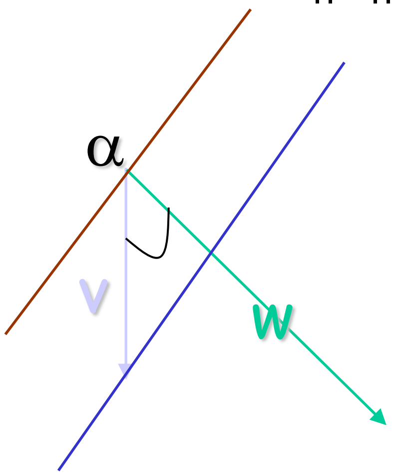
How to Mathematically Represent M



- Assume two vectors x^+ and x^-
  - $\boldsymbol{w} \, . \, \boldsymbol{x}^+ + b = +1$
  - $\boldsymbol{w} \, . \, \boldsymbol{x}^- + b = -1$
- Then M is just the projection of the vector v, $v = x^+ - x^-$, on the vector w)

How to Mathematically Represent M

- Recall vector basic (inner product)
  - $v \cdot w = \|v\| \, \|w\| \cos\alpha$



$$v \cdot w = (x_1, x_2) \cdot (y_1, y_2) = x_1 y_1 + x_2 \, y_2$$

$$v \cdot w = (x_1, x_2) \cdot (y_1, y_2) = \| v \| \| w \| \cos\alpha$$

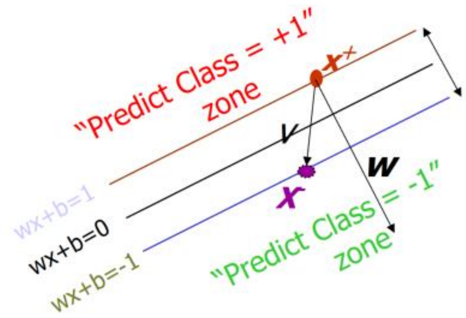
$$v_w = \| v \| \cos\alpha$$

- Therefore

$$M = \| v \| \cos\alpha$$

$$= \frac{\| v \| \| w \| \cos\alpha}{\| w \|} = \frac{v \cdot w}{\| w \|}$$

How to Mathematically Represent M



M=Margin Width

$$M = \frac{v \cdot w}{\| w \|}$$

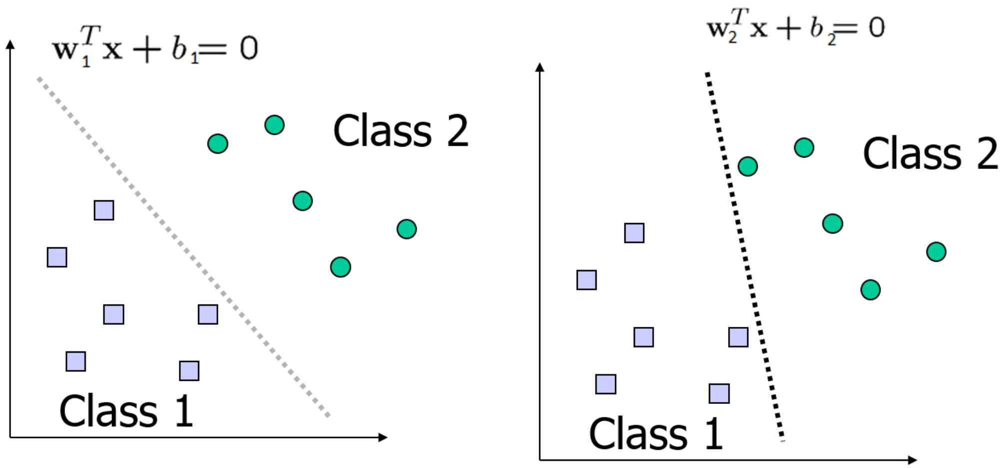
- Because $v = x^+ - x^-$
  - $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
  - $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$

$$M = \frac{v \cdot w}{\| w \|} = \frac{(x^+ - x^-) \cdot w}{\| w \|} = \frac{2}{\| w \|}$$

$$M = \frac{v \cdot w}{\| w \|} = \frac{(x^- - x^+) \cdot w}{\| w \|} = \frac{2}{\| w \|}$$

The Maximum Margin Classifier

- Now the problem becomes
  - Given a set of training examples
    - Find a set of decision surfaces (*w.r.t.* w and b values)
      - Each decision surface corresponding to one margin size
    - Compute the width of the margin
    - Select the one with the maximum margin *M*
      - which we believe, is the best classifier.



Outline

- Maximum Margin Classifier
- Deep Neural Networks
  - Convolutional Neural Network
  - Stacked AutoEncoder
  - Deep Belief Networks (Restrictive Boltzmann Machines)

Multiple Layered Networks

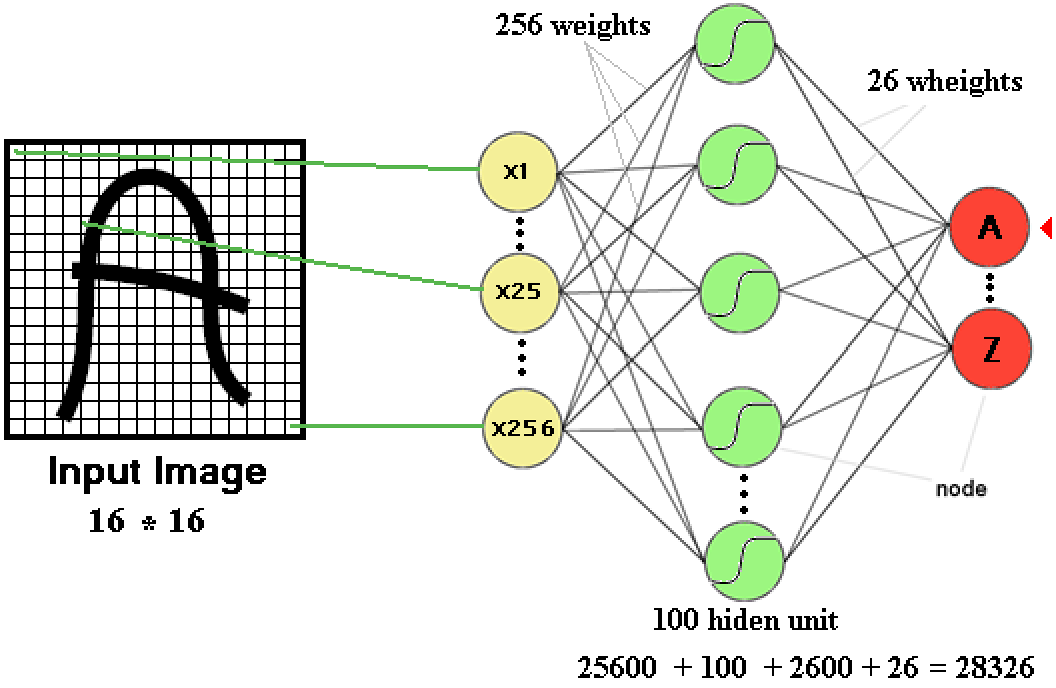
- Gradient descent weight updating is still effective, but …
  - Poor interpretation of knowledge/patterns gained from respective layers.
  - Early layers of MLP do not get trained well
    - Gradient Vanishing
      - Error reduces as it propagates to earlier layers
      - Top couple layers can usually learn any task "pretty well" and thus the error to earlier layers drops quickly as the top layers "mostly" solve the task
      - Lower layers never get the opportunity to use their capacity to improve results
    - Leads to very slow training
- Difficulties of supervised training of deep networks
  - Often not enough labeled data available while there may be lots of unlabeled data
    - Can we use unsupervised/semi-supervised approaches to take advantage of unlabeled data
  - Deep networks tend to have more local minima problems than shallow networks during supervised training

Overcome the Limitation of Back-Propagation

- Keep the efficiency of using a gradient method for adjusting the weights
  - Use it for modeling the structures of the input data.
    - Feature hierarchy
  - Adjust the network weights to maximize the probability that a generative model would have produced the input data (with maximum probability).
    - Train Generative models
  - Learning P(image) but not P(label | image)

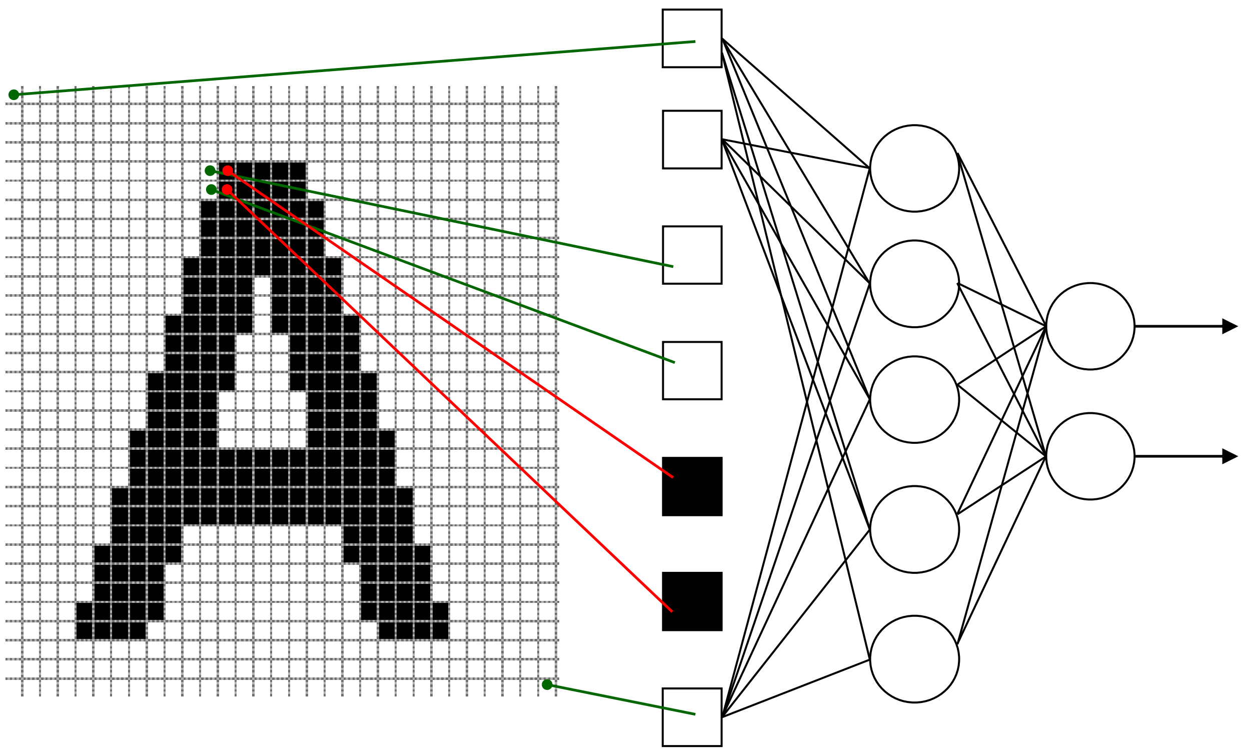
Drawbacks of previous neural networks

The number of trainable parameters becomes extremely large



Drawbacks of previous neural networks

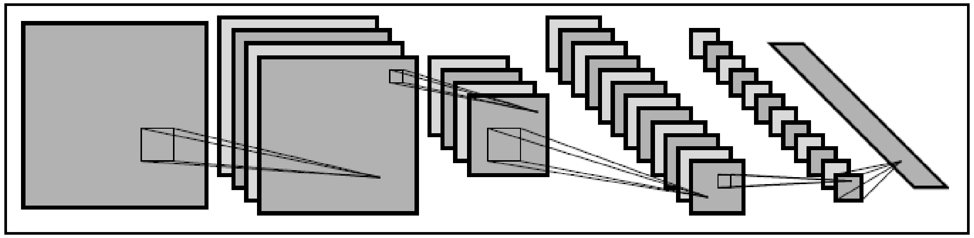
Little or no invariance to shifting, scaling, and other forms of distortion

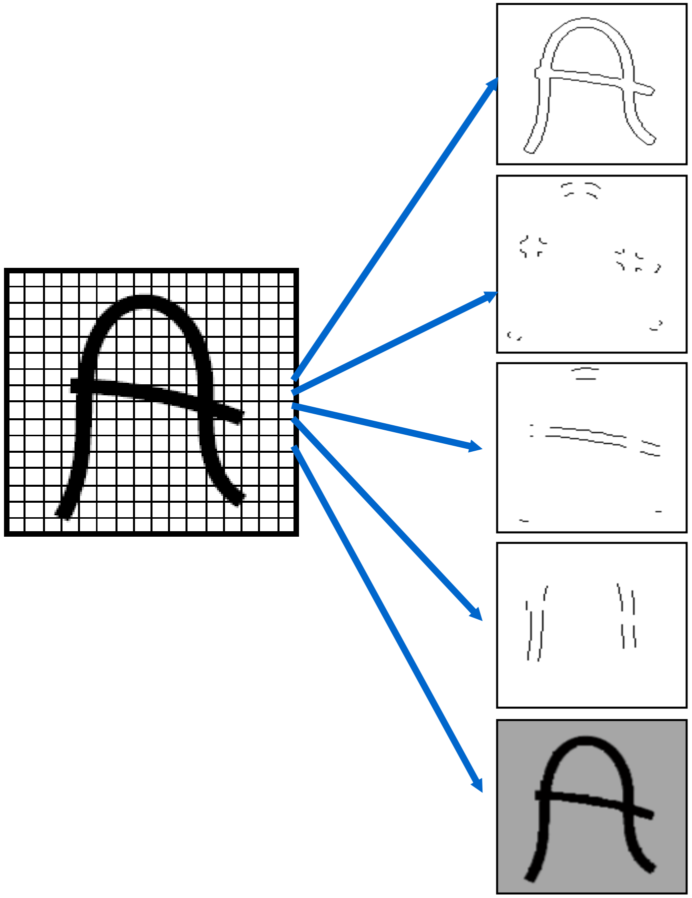


Scaling and Other Forms of Distortion

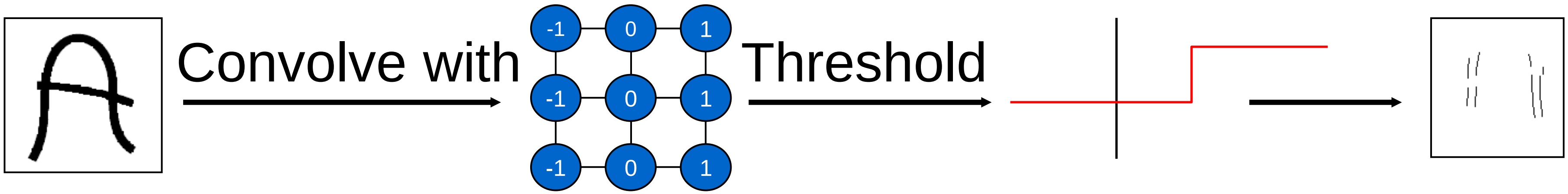
Feature extraction layer or Convolution layer

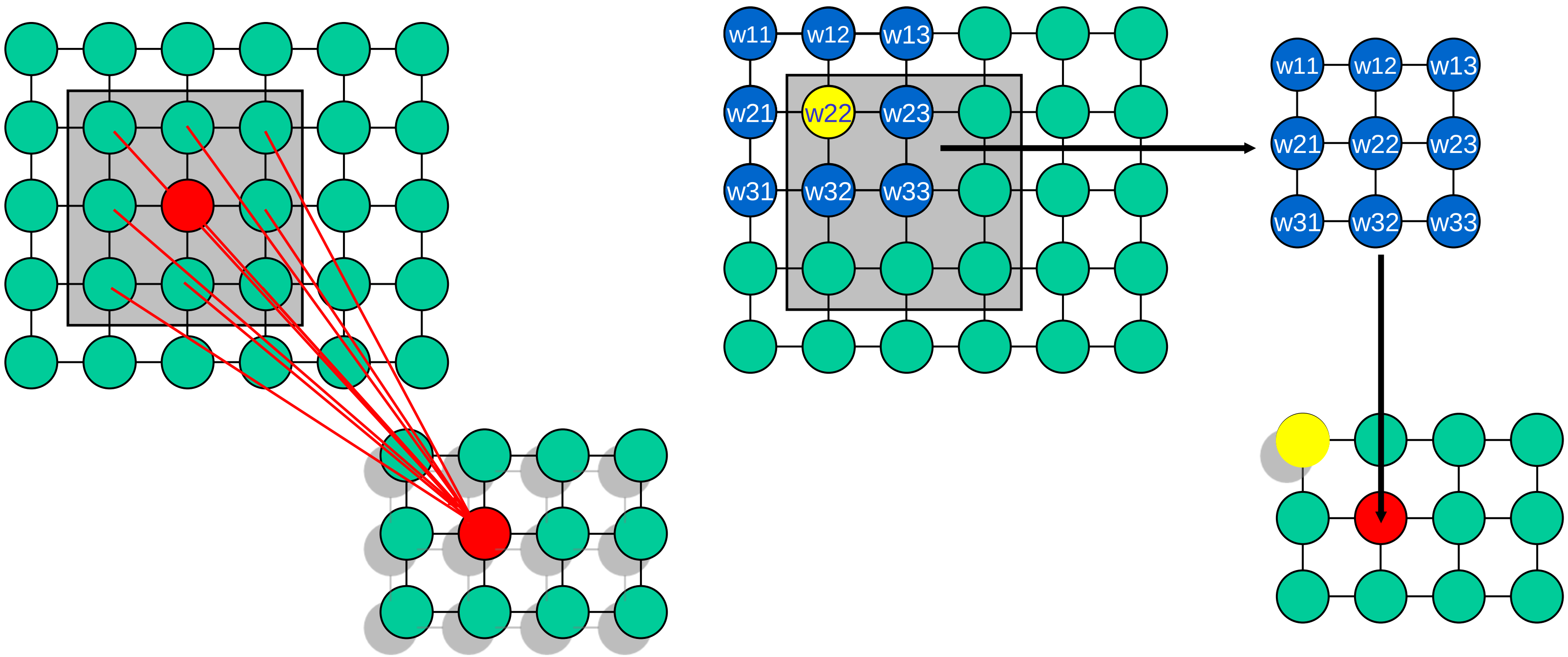
Detect the same feature at different positions in the input image.





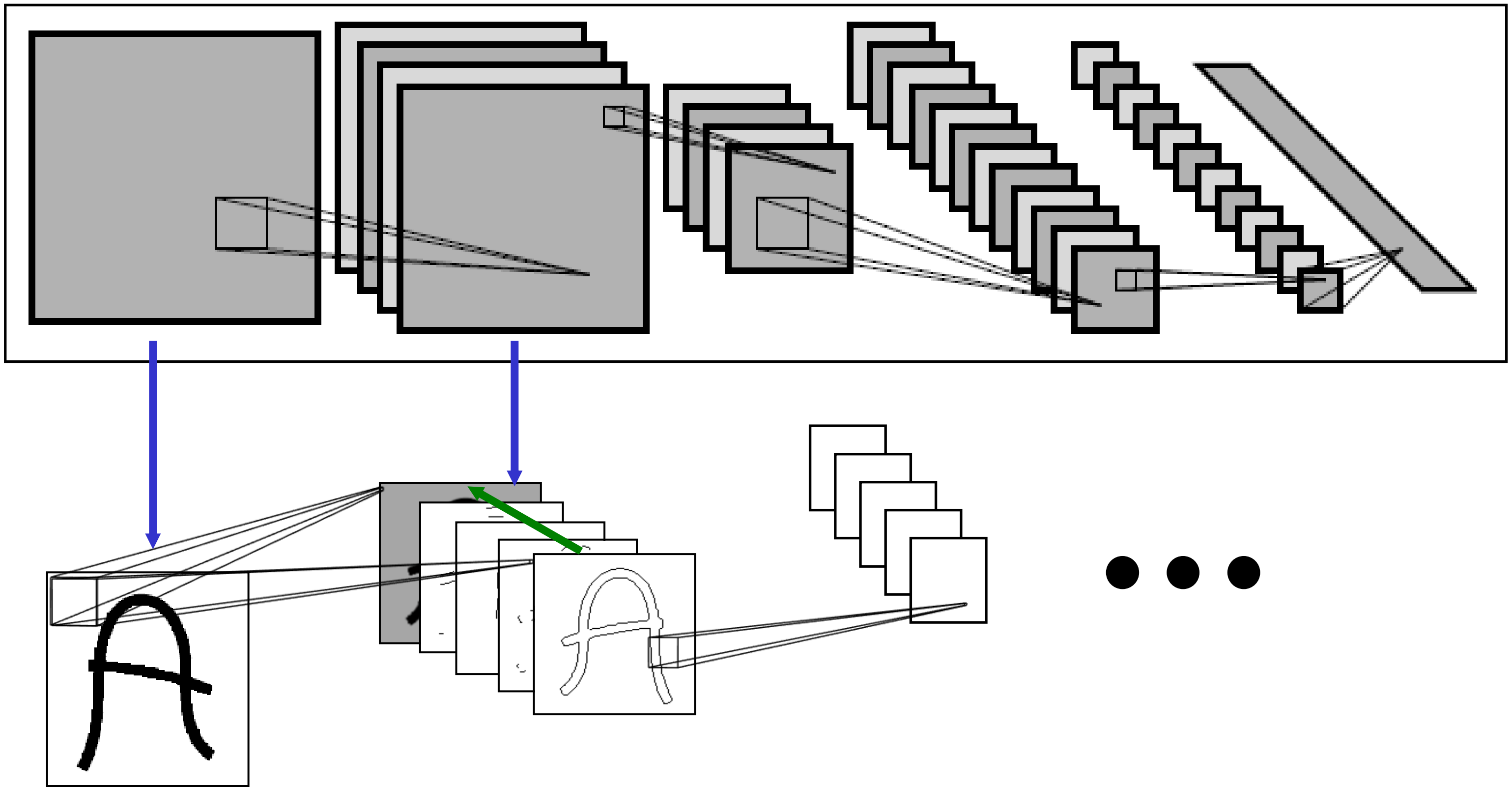
Feature extraction





Feature extraction

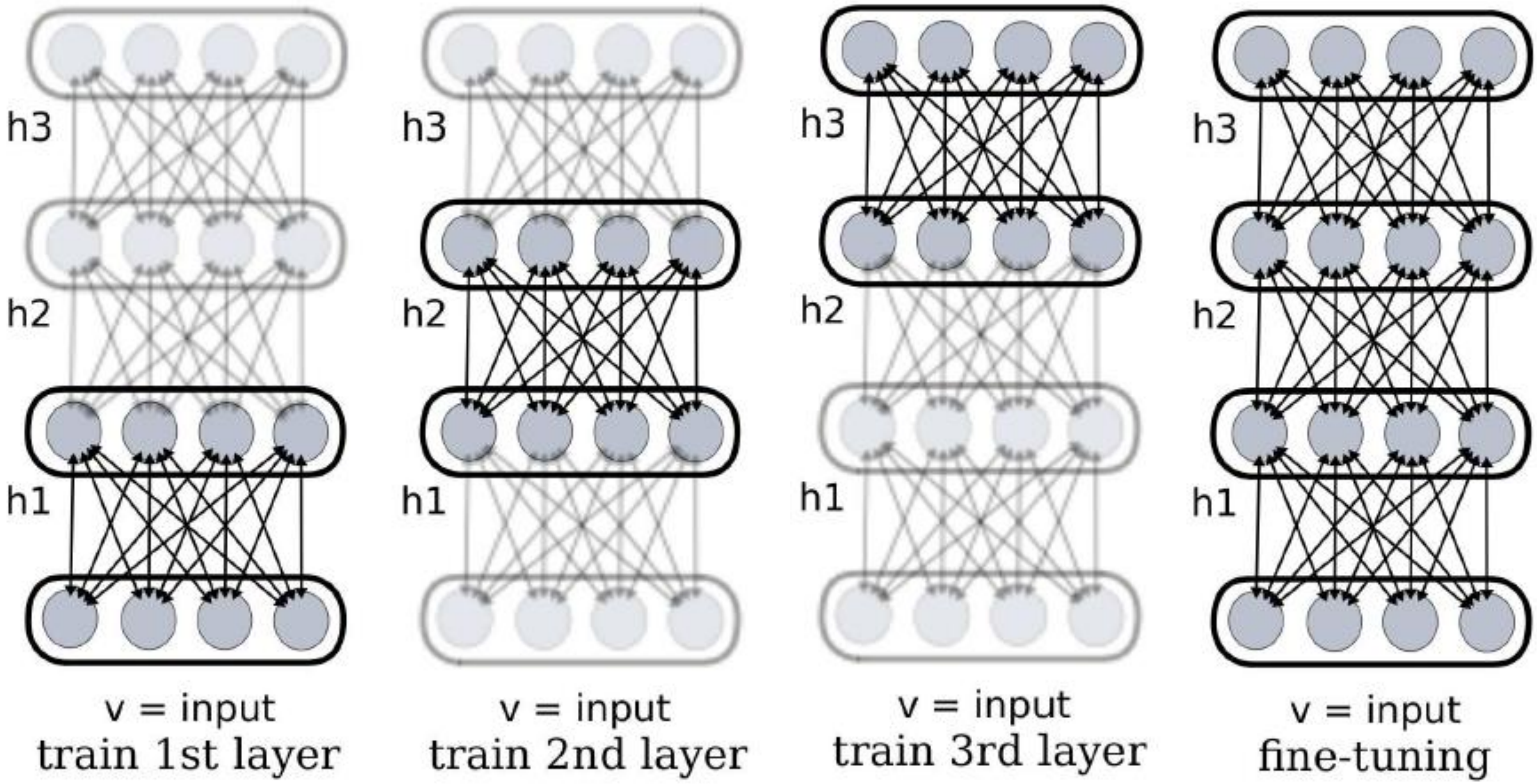
If a neuron in the feature map fires, this corresponds to a match with the template.



Outline

- Maximum Margin Classifier
- Deep Neural Networks
  - Convolutional Neural Network
  - Stacked AutoEncoder
  - Deep Belief Networks (Restrictive Boltzmann Machines)

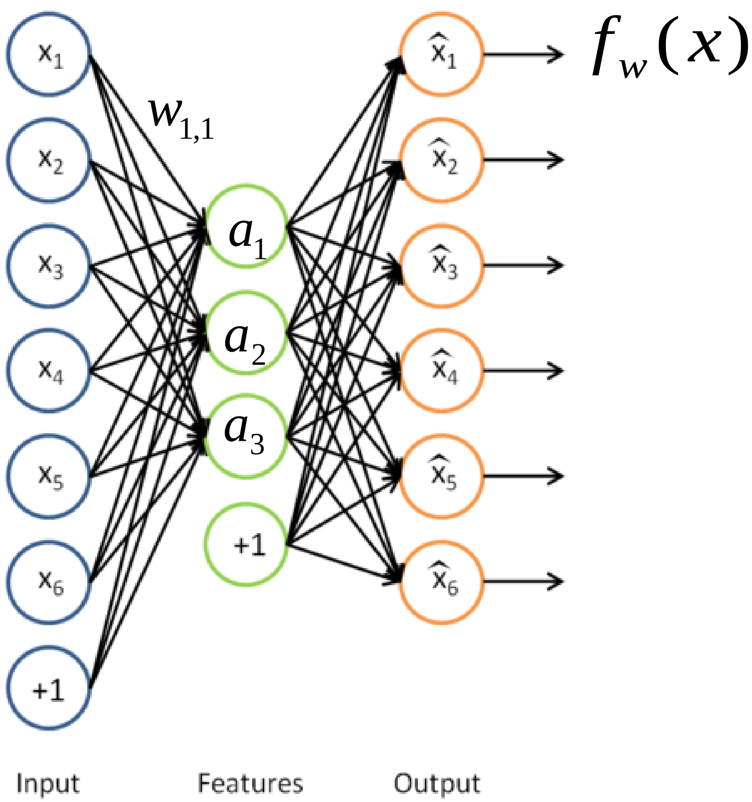
Deep Learning Scheme (Unsupervised+ supervised)



- A greedy unsupervised layer-wise pre-training stage followed by a supervised fine-tuning stage (affecting to all layers).

Auto-Encoders

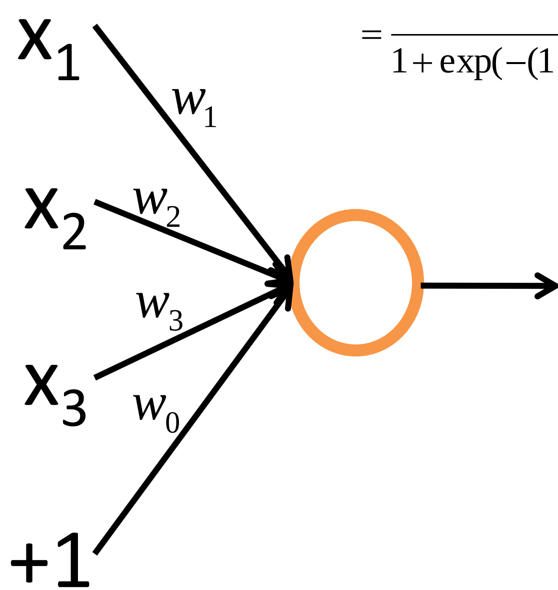
- A type of unsupervised learning which tries to discover generic features of the data
  - Learn identification function by learning important sub-features (not by just passing through data)
  - Compression, etc.



Neuron:

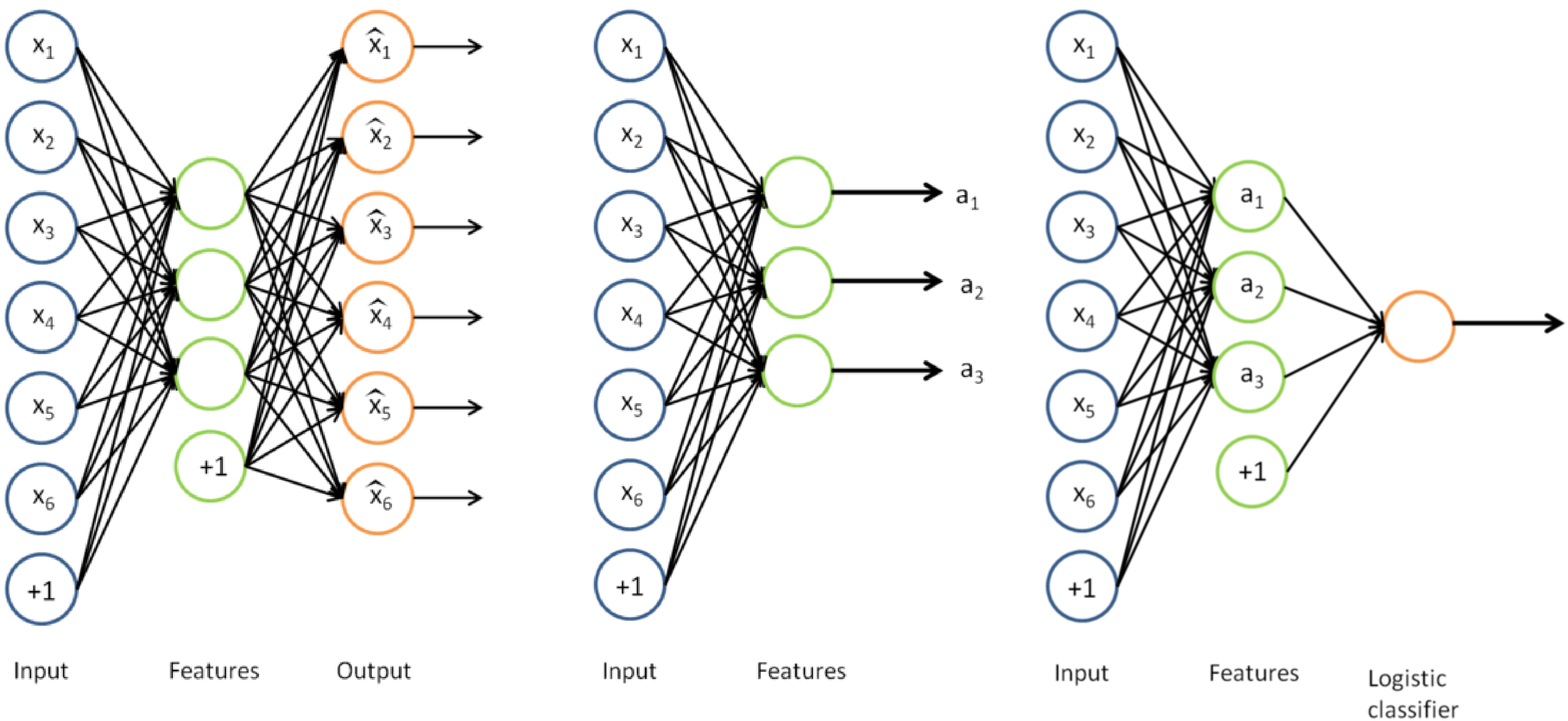
$$f_w(x) = \frac{1}{1+\exp(-w^T x)}$$

$$= \frac{1}{1+\exp(-(1\times w_0 + x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3))}$$



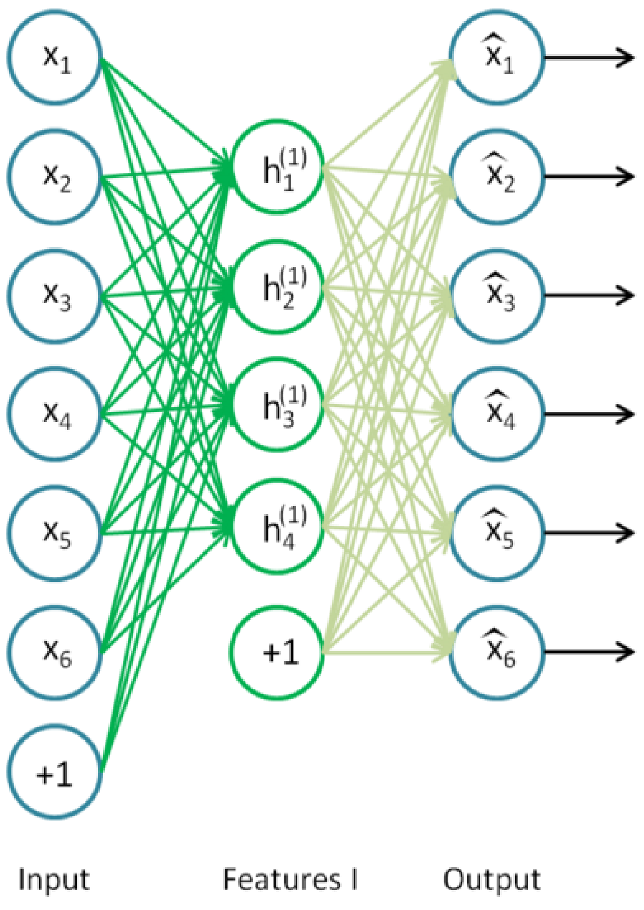
Auto-Encoders

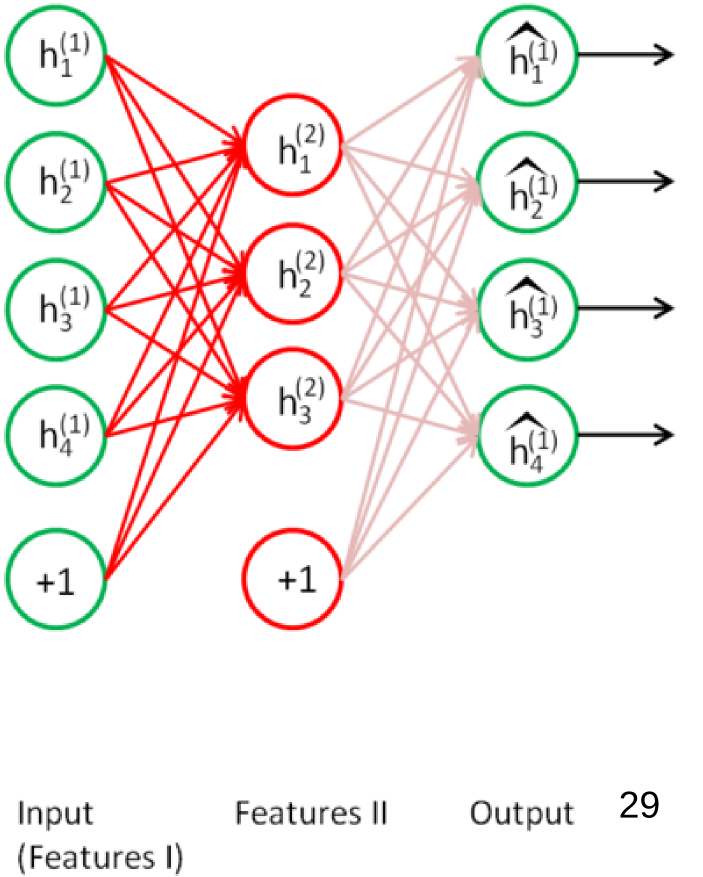
- Using pre-trained network for classification



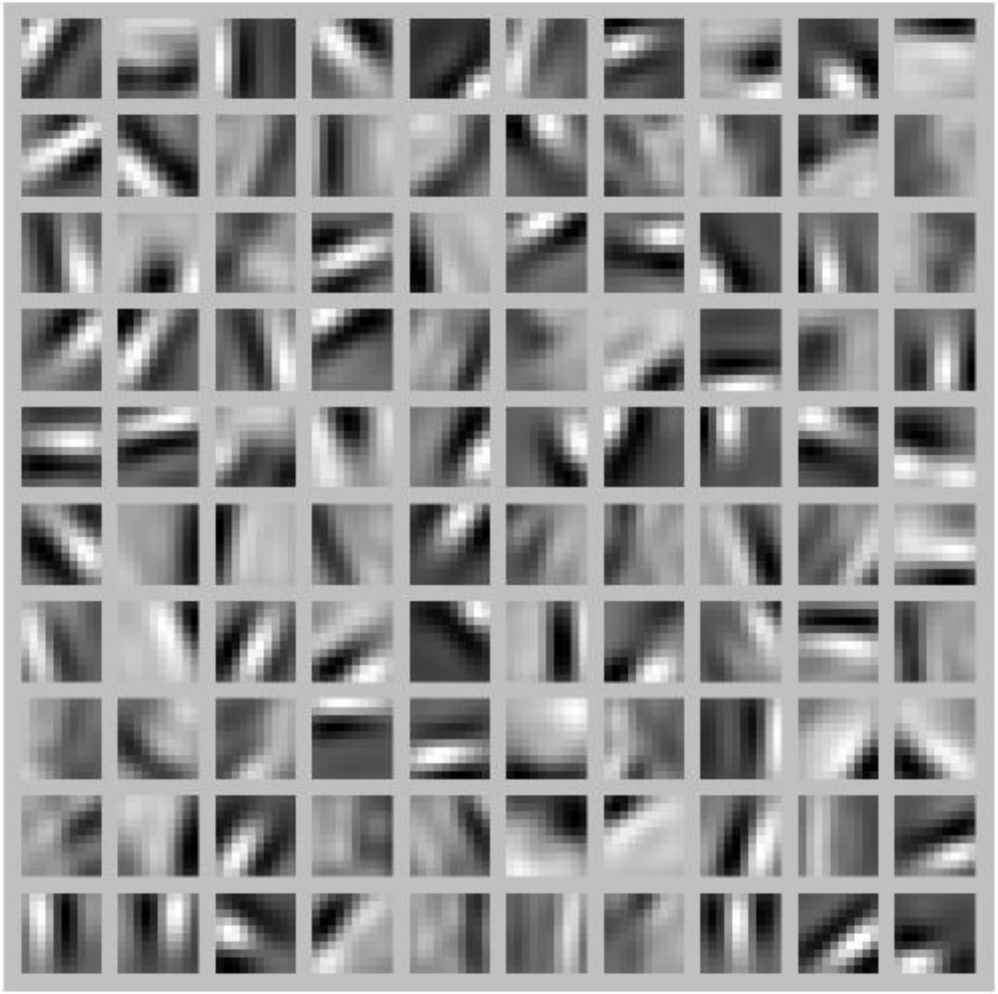
Stacked Auto-Encoders

- Stack many (sparse) auto-encoders in succession and train them using greedy layer-wise training
- Drop the decode layer each time





Visualize hidden units



- Different hidden units have learned to detect edges at different positions and orientations in the image.

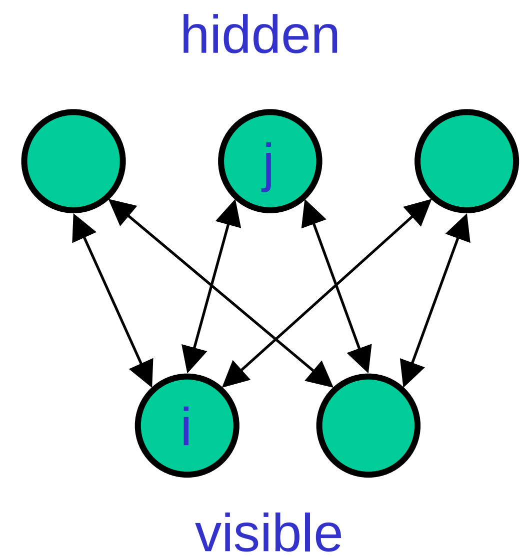
Outline

- Maximum Margin Classifier
- Deep Neural Networks
  - Convolutional Neural Network
  - Stacked AutoEncoder
  - Deep Belief Networks (Restrictive Boltzmann Machines)

Restricted Boltzmann Machines

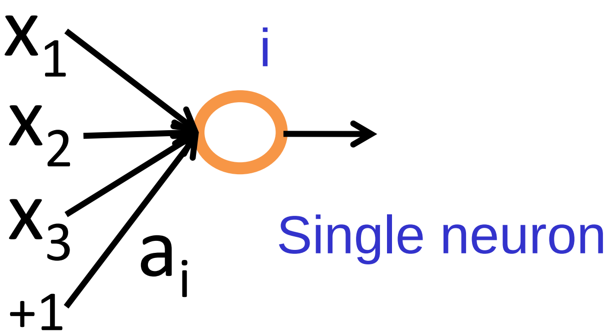
(Smolensky ,1986, called them "harmoniums")

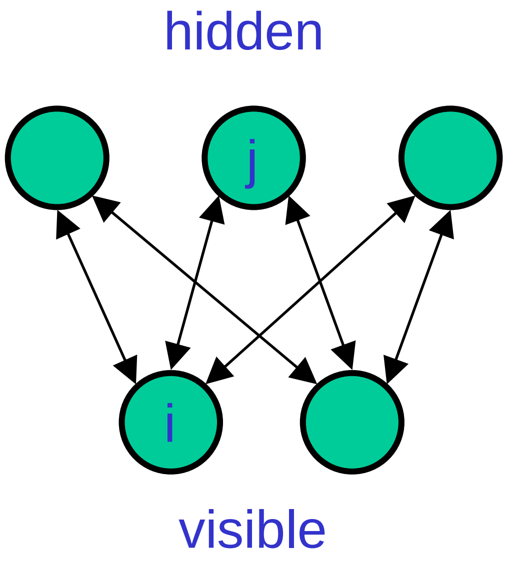
- Restrict the connectivity to make learning easier.
  - Only one layer of hidden units.
    - Can be stacked to form many layers
  - No connections between hidden units.
  - Hidden units have binary states:
    - "0" or "1".
- In an RBM, the hidden units are conditionally independent given the visible states.
  - So we can quickly get an unbiased sample from the posterior distribution when given a data-vector.



$$P(\mathrm{v} \mid \mathrm{h}) = \prod_{i=1}^{m} P(v_i \mid h)$$

$$P(\mathrm{h} \mid \mathrm{v}) = \prod_{j=1}^{n} P(h_j \mid v)$$





$$p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h}) \quad \text{and} \quad p(v_i = 1|\mathbf{h}) = \text{sigm}\left(a_j + \sum_j h_j w_{ij}\right)$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v}) \quad \text{and} \quad p(h_j = 1|\mathbf{v}) = \text{sigm}\left(b_j + \sum_i v_i w_{ij}\right)$$

- Sigm(x)=1/(1+exp(-x)): logistic (sigmoid) activation function

The Energy of a joint configuration

(ignoring terms to do with biases)

binary state of hidden unit j

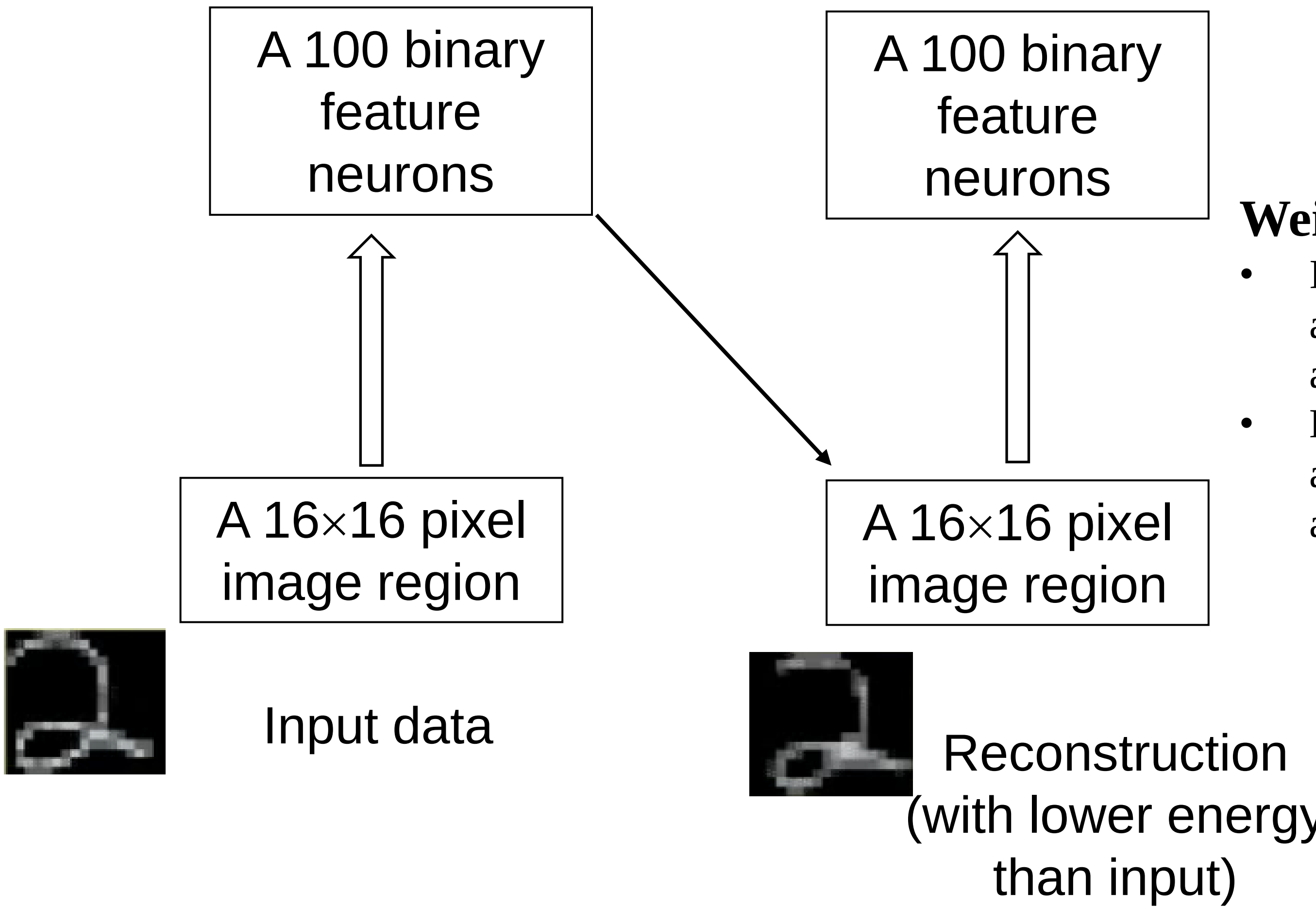
binary state of visible unit i

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j h_j w_{i,j} v_i$$

Energy with configuration v on the visible units and h on the hidden units

weight between units i and j

A practical view



Weight updating:

- Increase weights between an active pixel and an active feature
- Decrease weights between an active pixel and an active feature

Outline

- Maximum Margin Classifier
- Deep Neural Networks
  - Convolutional Neural Network
  - Stacked AutoEncoder
  - Deep Belief Networks (Restrictive Boltzmann Machines)