Lecture 17: Neural Networks Intro

COMP 343, Spring 2022

Victoria Manfredi

Acknowledgements: These slides are based primarily on content from the book "Machine Learning" by Tom Mitchell and slides created by Vivek Srikumar (Utah) and Dan Roth (Penn)

Today's Topics

Homework 6

- Due Wednesday, April 6 by 5p

Miscellaneous and project discussion

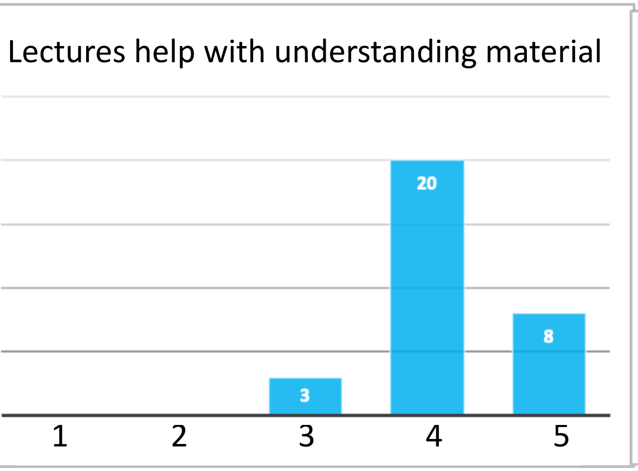
Recap

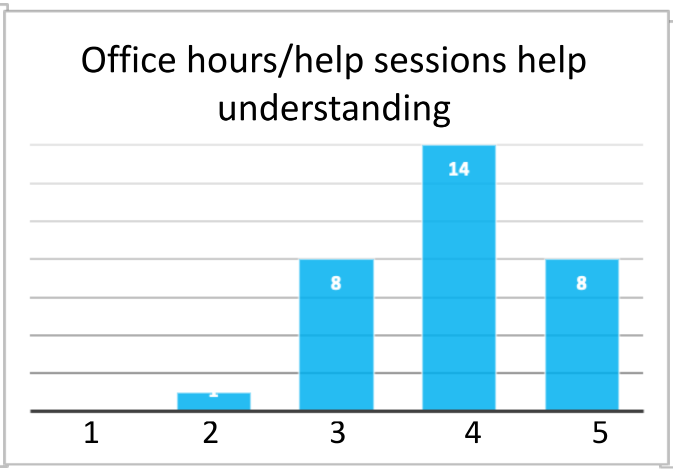
Neural networks

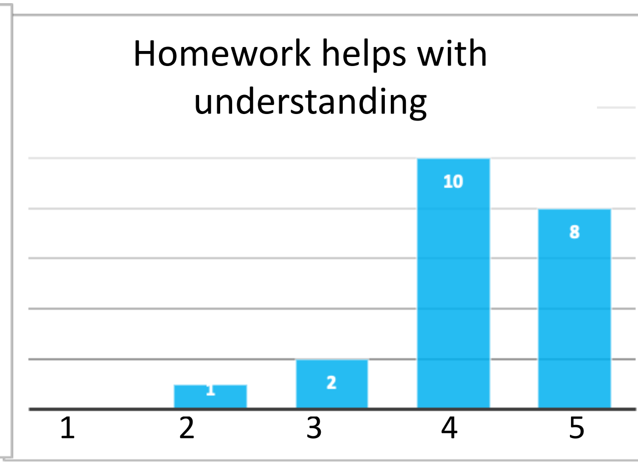
- Structure, expressiveness
- Prediction using a neural network
- Training neural networks
- Practical concerns

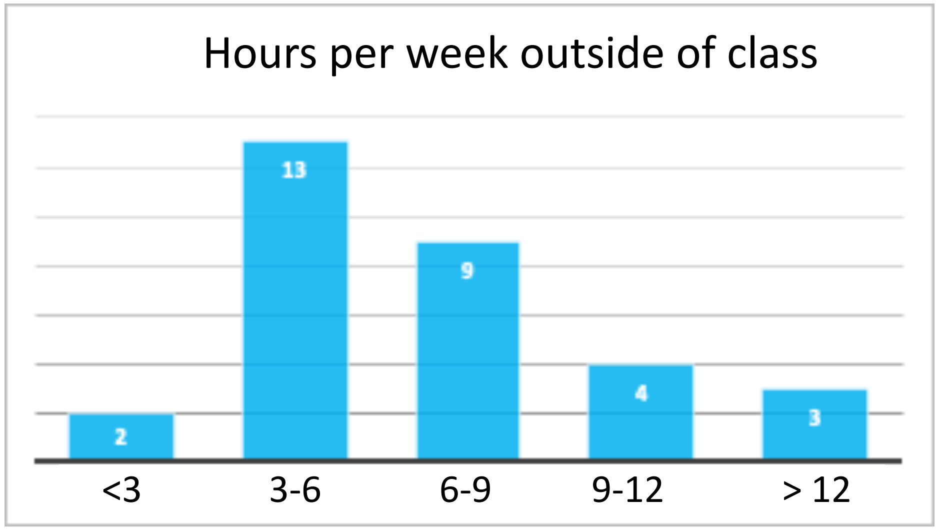
Miscellaneous

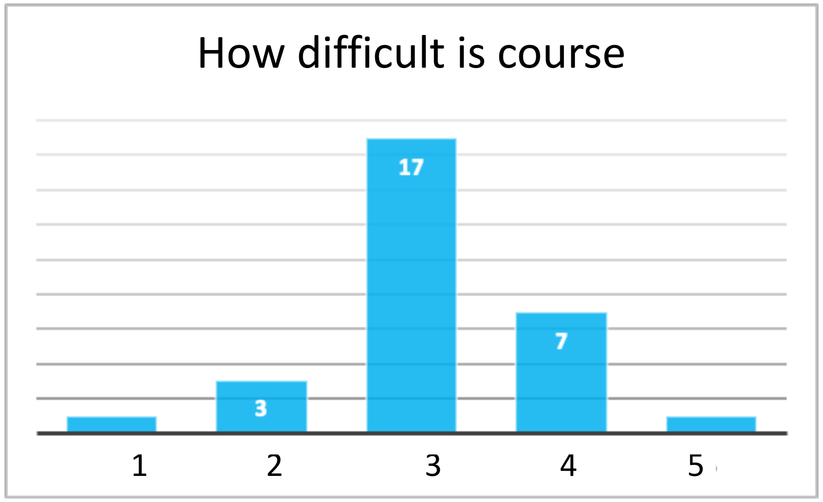
Survey


Lectures help with understanding material
3
20
8
1
2
3
4
5


Office hours/help sessions help understanding
1
8
14
8
1
2
3
4
5


Homework helps with understanding
1
2
10
8
1
2
3
4
5


Hours per week outside of class
2
13
9
4
3
<3
3-6
6-9
9-12
> 12


How difficult is course
3
17
7
1
2
3
4
5

Survey

Best/More

- Like lectures (11)
- Like talking with neighbors on problems (7)
- Interesting material (7)
- Like homework/coding (6)
- Like office hours and help sessions (5)

Worst/Less

- Better organization and documentation for homework code (9)
- Scikit is hard to work with (4)
- Too many slides (3)
- Too much review in lecture (3)
- Homework is difficult (3)
- Disconnect between lectures and homework (2)
- Have more written problems on homework (2)

Would like to have

- Discuss code in class (3)
- More help sessions (maybe from Monday to Thursday)
- More 3 min discussion and activities in class
- More real-life examples in class
- “How everything connects” day to review basic course contents

Other miscellaneous

- Zoom: Are the zoom recordings helpful?
- Exam: We'll discuss once everyone has taken it
- Code organization of homework: Really should be using modules (e.g., create a regression class and put in a separate module named regression.py). Why can modules be hard to work with when developing code?
    - If you change a module, you have to reload module (i.e., import regression) and then delete __pycache__ directory to see changes take effect. Like doing "make clean" with C. So need a script to clean up code before re-running. Would doing this help?
- scikit (and soon tensorflow): Why? In practice, you will almost certainly use rather than writing your own code. Would talking about scikit in class or adding more details in homework help?
- Homework 7: what do folks think of writing most of code from scratch (i.e., implementing forward pass of neural network)?
    - At this point you have seen how to work with vectors, train/test split, cross-validation
    - But we still need to make sure easy for CAs to grade, so should be that running hw7.py produces whatever output is required
- One ask of you: look through recap slides at start of class while waiting, so if any questions can focus on those during recap

Project

What can your project be?

Individual or group project: if you want to work with more than 2 people, you need to convince me. A group project requires more work to be done for the same grade.

Option 1

- For a given dataset, compare performance of different algorithms with different hypothesis spaces. Organize code for algorithms into modules/libraries for easy re-use

Option 2

- Learn about a more complex model in scikit or tensorflow and apply to data

Option 3

- Implement from scratch an algorithm not implemented in homework and apply to data
- E.g., support vector machine, convolutional neural network, recurrent neural network …

Option 4

- Something more theoretical: e.g., work through understanding the theory more deeply for an algorithm (e.g., perceptron proof of number mistakes needed to converge, or neural network universal approximation theorem …)

If you have another option in mind, please come talk to me

How will your project be graded?

Code

- Are algorithms implemented and used correctly?
- Is there a clear approach to evaluate how well algorithm is learning?
- How well is code organized/written?
- Included a readme file explaining how to run code?

Project report: 5-6 pages, should be typed pdf only

- What problem did you work on?
- Why is problem interesting/important/relevant?
- What are the important ideas you explored?
- What ideas from class did you use?
- What did you learn?
- Do you clearly describe your proposed algorithmic approach?
- Does your proposed algorithmic approach make sense for addressing problem?
- What are your results? Are they presented clearly?
- How clear and well-organized/written is project report overall?
- If you had more time, what would be future work for project?

Ambition component to get an A/A+

- Reward for trying something more difficult

Tentative project timeline

April 20: Project proposal due (2.5/20)

- 1 page write-up
- Describe problem and approach

April 27: Checkpoint 1 of code with me or CAs (2.5/20)

- Design code skeleton
- Describe any issues

May 4: Checkpoint 2 of code with me or CAs (5/20)

- Draft of code
- Draft of final report
- Describe any issues

May 11: Project code and final report submitted (10/20)

Recap

Batch gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

1. Initialize $\mathbf{w}_0$

2. For $t = 0,1,2,\ldots$

What is the gradient of J?

   – $\boxed{\text{Compute gradient of } J(\mathbf{w}) \text{ at } \mathbf{w}_t\text{. Call it } \nabla J(\mathbf{w}_t)}$

   – Update $\mathbf{w}$ as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - r\nabla J(\mathbf{w}_t)$$

where r is the learning rate (a small constant)

Gradient of the cost J at point $\mathbf{w}$

Remember that $\mathbf{w}$ is a vector with d elements

$$\mathbf{w} = [w_1, w_2, w_3, \ldots, w_j, \ldots, w_d]$$

To find the best direction in the weight space $\mathbf{w}$ we compute the gradient of J with respect to each of the components of

$$\nabla J(\mathbf{w}_t) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \cdots, \frac{\partial J}{\partial w_d} \right]$$

Gradient will be vector with d elements since $\mathbf{w}$ is a vector with d elements

Each element is a partial derivative

Need to compute every element of $\nabla J(\mathbf{w}_t)$ to define gradient

Gradient of the cost J at point $\mathbf{w}$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

The gradient is of the form $\nabla J(\mathbf{w}_t) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \cdots, \frac{\partial J}{\partial w_d}\right]$

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots) \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)(-x_{ij}) \\
&= \boxed{-\underbrace{\sum_{i=1}^{m}}_{\text{Sum of}}\underbrace{(y_i - \mathbf{w}^T\mathbf{x}_i)}_{\text{Error}}\ \text{x}\ \underbrace{x_{ij}}_{\text{Input}}}
\end{aligned}$$

One element of the gradient vector

Negative of this gradient is how much to change jth weight

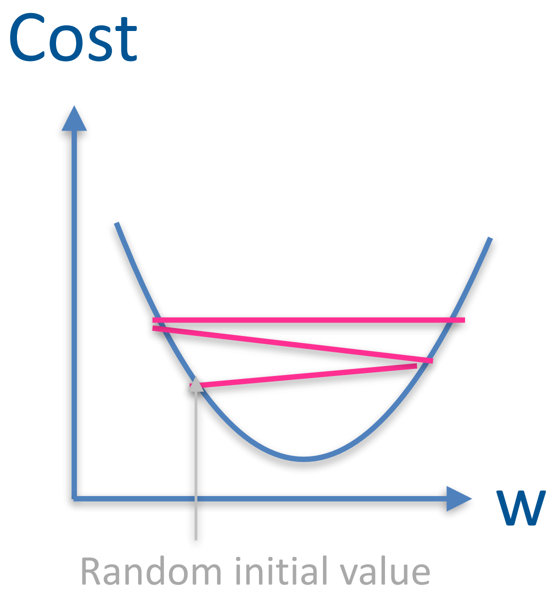
Larger features (x_{ij}) with larger errors will cause larger change

Review

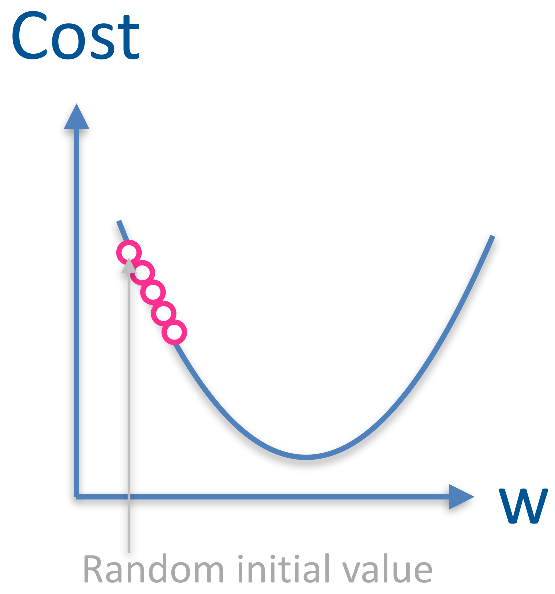
How to improve likelihood of convergence

- Normalize values of features and predictions: Important to normalize features when using gradient descent (otherwise takes longer to converge). All features should have a similar scale
- Decrease learning rate over time
- Check for weights converging
- Cross-validation to determine how best to set hyper-parameters like number of epochs or learning rate

Impact of learning rate



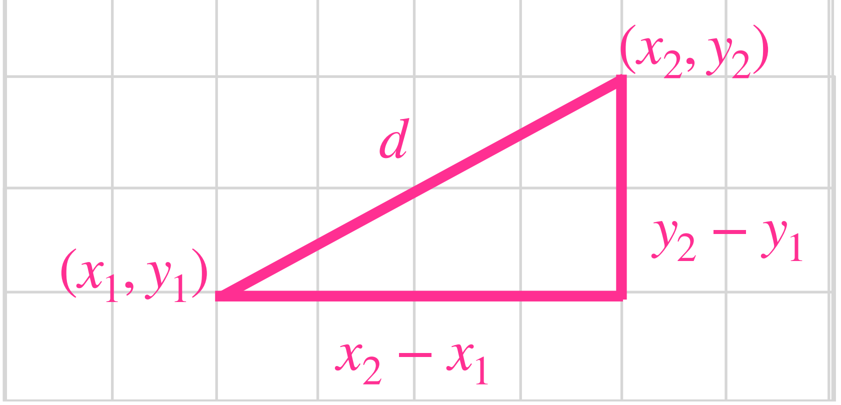
Learning rate
too large



Learning rate
too small

Euclidean distance

How far apart are two points?



Euclidean distance between 2 points in 2 dimensions:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

General formula for Euclidean distance between 2 points with k dimensions:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{k} (p_i - q_i)^2}$$

where $\mathbf{p}$ and $\mathbf{q}$ are each k-dimensional vector

Euclidean norm (aka L2 norm)

Defined as distance between a point and zero (the origin)

$$d(\mathbf{p}, \mathbf{0}) = \sqrt{\sum_{i=1}^{k} p_i^2}$$ also written $||\mathbf{p}||$

- A norm is a measure of a vector's length

Batch gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

1. Initialize $\mathbf{w}_0$

2. Update the parameters:

$$\nabla J(\mathbf{w}_t) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \cdots, \frac{\partial J}{\partial w_d}\right]$$ where $$\frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m}(y_i - \mathbf{w}^T x_i)x_{ij}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - r_t \nabla J(\mathbf{w}_t)$$

$$r_t = \frac{c_1}{t^a + c_2}$$

3. Repeat step 2 until $||\nabla J(\mathbf{w})|| < \theta$ or until the maximum number of iterations is reached or until error is less than threshold

Stochastic gradient descent for LMS

1. Initialize $\mathbf{w}_0$

2. Update the parameters:

   For each training example $(\mathbf{x}_i, y_i)$, update $\mathbf{w}$. For each element of the weight vector (w_j)

$$w_j^{t+1} = w_j^t - r_t(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}$$

$$r_t = \frac{c_1}{t^a + c_2}$$

May not converge, but online/ incremental algorithms often preferred when training set is very large

3. Repeat step 2 until $||\nabla J(\mathbf{w})|| < \theta$ or until the maximum number of iterations is reached or until error is less than threshold

Linear regression: summary

What we want: predict a real-valued output using feature representation of the input

Assumption: output is a linear function of the inputs

Learning by minimizing total cost

- Gradient descent and stochastic gradient descent to find the *best* weight vector
- This particular optimization can be computed directly by framing the problem as a matrix problem

3 min discussion

Question: Recall Root Mean Squared error on homework. If cost function achieves a value of zero on a set of training data points, what property do the training data points have?

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i}^{m}(y_{true,i} - y_{pred,i})^2}$$

Question: Suppose you build a linear regression model for some data and you observe the weight of one of the features has a relatively high negative value. This suggests that

(a) This feature has a strong effect on the model (should be retained)
(b) This feature does not have a strong effect on the model (should be ignored)
(c) It is not possible to comment on the importance of this feature without additional information

3 min discussion

Question: Recall Root Mean Squared error on homework. If cost function achieves a value of zero on a set of training data points, what property do the training data points have?

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i}^{m}(y_{true,i} - y_{pred,i})^2}$$

Answer: A root mean squared error value is 0.0 means that all predictions matched the expected values exactly

Question: Suppose you build a linear regression model for some data and you observe the weight of one of the features has a relatively high negative value. This suggests that

(a) This feature has a strong effect on the model (should be retained)
(b) This feature does not have a strong effect on the model (should be ignored)
(c) It is not possible to comment on the importance of this feature without additional information

Answer is (c). A high magnitude suggests that the feature is important. However, another feature may be highly correlated with this feature and its weight also has a high magnitude with the opposite sign, in effect cancelling out the effect of the former. Thus, we cannot really remark on the importance of a feature just because its weight has a relatively large magnitude.

Linear Regression

REGULARIZATION

Regularization

- Possible to minimize training error to well. Instead, modify learning algorithm to favor "simpler" prediction rules to avoid overfitting

- Most commonly, regularization refers to modifying the loss function to penalize certain values of the weights you are learning. Specifically, penalize weights that are large

- How do we define whether weights are large? Use L2 norm of $\mathbf{w}$ again (but bias term is not regularized)

L2 regularization

New goal for minimization

$$J(\mathbf{w}) + \lambda ||\mathbf{w}||^2$$

Square to eliminate square root: easier to work with mathematically

This is whatever loss function we are using

By minimizing this we prefer solutions where $\mathbf{w}$ is closer to $\mathbf{0}$

λ is a hyperparameter that adjusts trade-off between low training loss and having low weights

Eventually, penalty of large $||w||^2$ will outweigh gains in loss function. Helps with generalization because won't give large weight to features unless there is sufficient evidence that they are useful

L2 Regularization

- When the regularizer is the squared L2 norm $||\mathbf{w}||^2$, this is called L2 regularization.

- This is the most common type of regularization

- When used with linear regression, this is called Ridge regression

- L2 regularization can be added to other algorithms like perceptron (or any gradient descent algorithm)

L1 Regularization

- Another common regularizer is the L1 norm:

$$||\mathbf{w}||_1 = \sum_{j=1}^{k} |w_j|$$

- When used with linear regression, this is called Lasso regression

- Often results in many weights being exactly 0 (while L2 just makes them small but nonzero)

Neural Networks

MOTIVATION

Where are we?

Learning algorithms

- Decision trees
- Perceptron
- Linear regression

Produce *linear* classifiers, regressors

General learning principles

- Overfitting
- Training and generalization errors
- (Regularized Empirical) Loss Minimization

Not really resolved

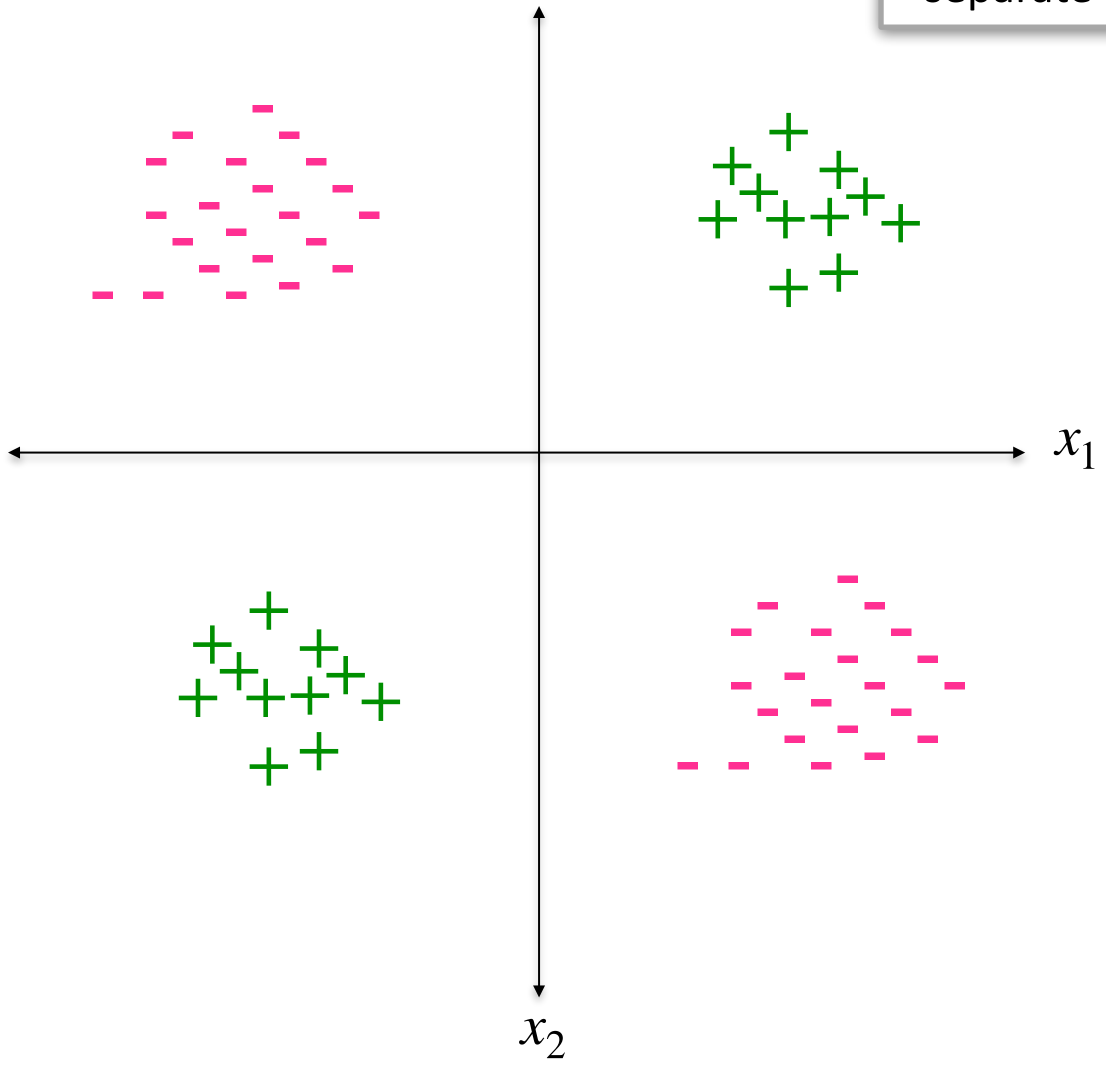
- What if we want to train non-linear classifiers?
- Could transform features but then where does transformation come from

Not all functions are linearly separable

(The parity or XOR function)

Can't draw a line to separate the two classes

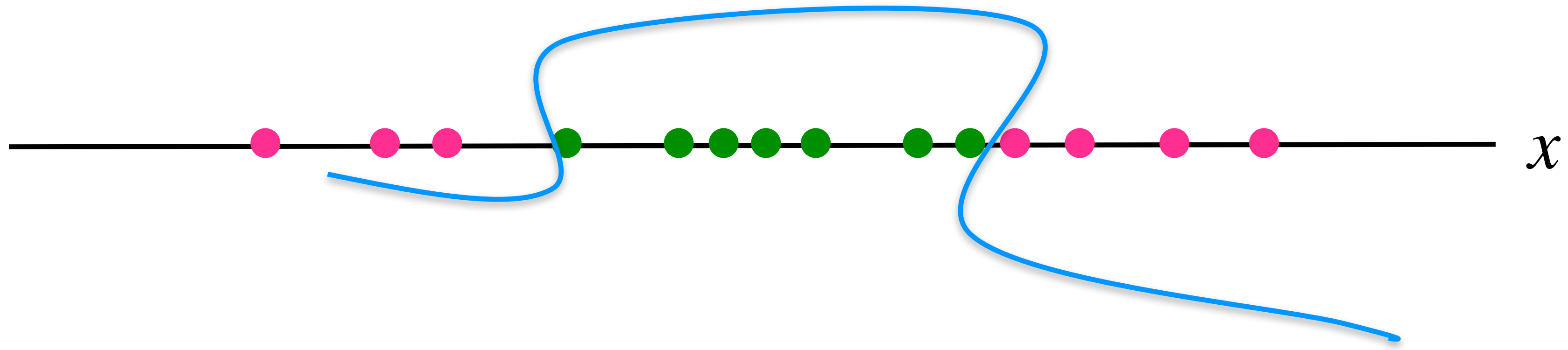
x_1	x_2	f
-1	-1	+1
+1	-1	-1
-1	+1	-1
+1	+1	+1



Even these functions can be made linear

These points are not separable in 1-dimension by a line

What is a one-dimensional line, by the way? Point

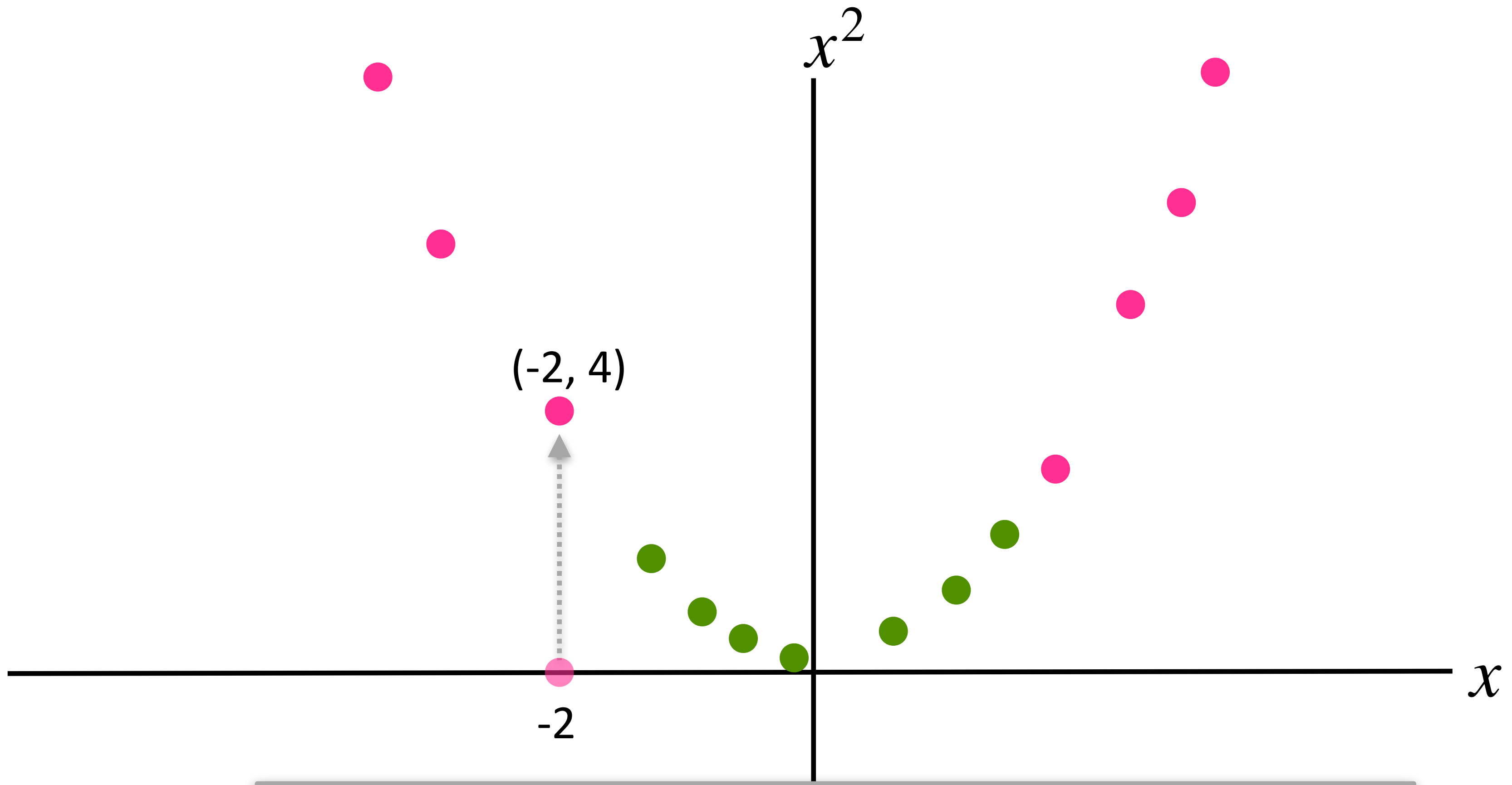


What can we do?

The blown up feature space

The trick: use feature conjunctions

Transform points: represent each point x in 2 dimensions by (x, x^2)

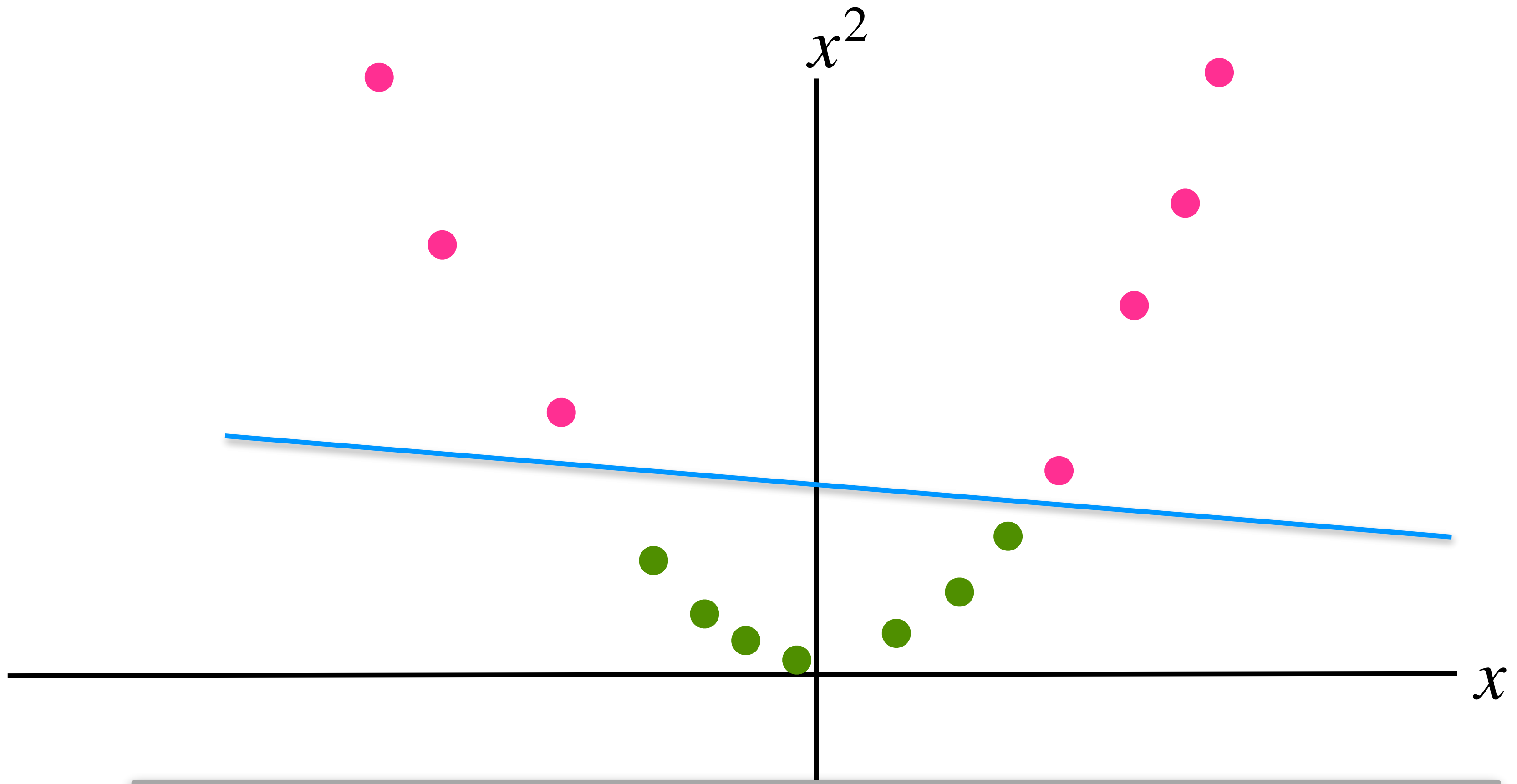


Now the data is linearly separable in this space!

The blown up feature space

The trick: use feature conjunctions

Transform points: represent each point x in 2 dimensions by (x, x^2)



Key issue: representation. What transformation to use?

Non-linear ⇒ Neural networks

Linear separability depends on FEATURES!!

Have system to produce features, that make function linearly separable

⇒ Neural networks

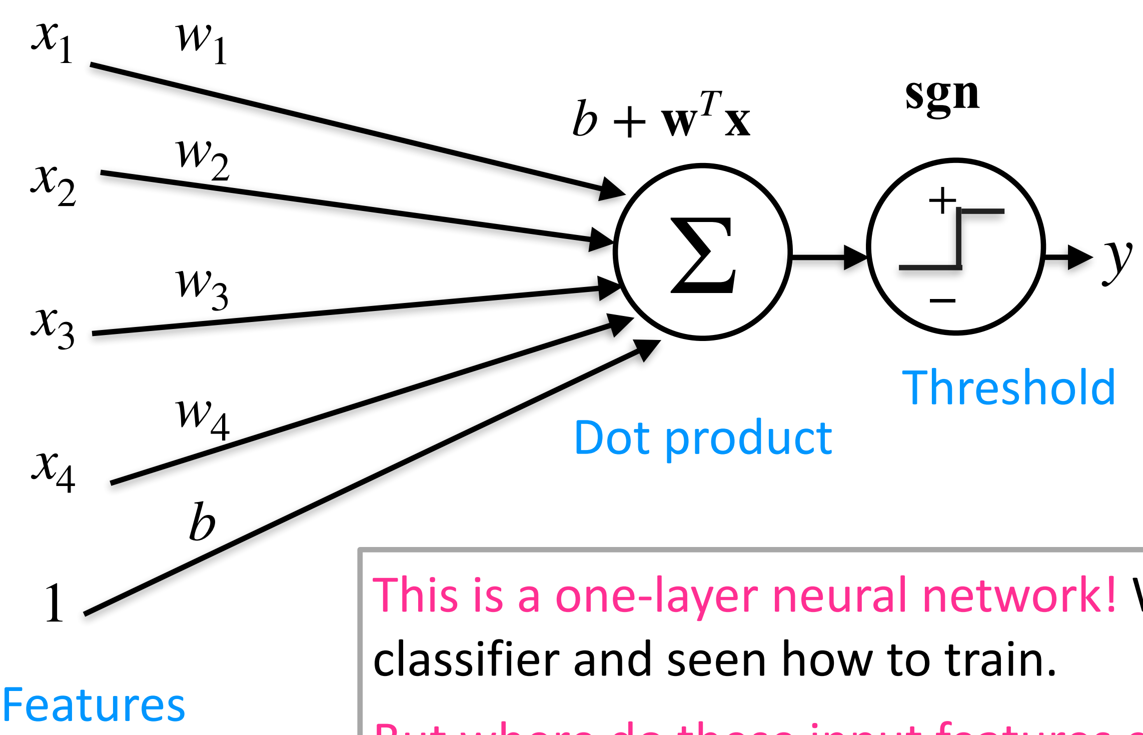
Neural Networks

OVERVIEW

Neural networks: what we will cover

- What is a neural network?
  - What is the hypothesis class?
  - What is the structure?
  - What kinds of mathematical functions do they express?
- Predicting with a neural network
- Training neural networks
- Practical concerns

We have seen linear threshold units



Prediction

$$\text{sgn}(\mathbf{w}^T x + b) = \text{sgn}(\sum_i w_i x_i + b)$$

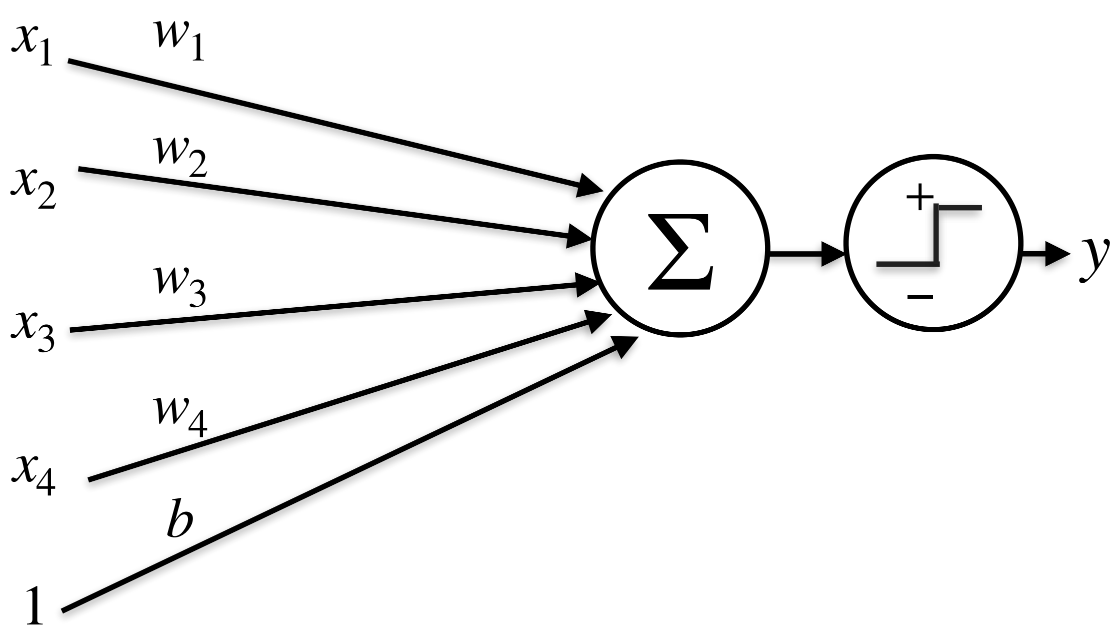
Learning

Various algorithms, in general, minimize loss

This is a one-layer neural network! We've seen as linear classifier and seen how to train.

But where do these input features come from? Do we have a domain expert who helps us create features? What if we transform problem of creating features into a learning problem instead? What if the features themselves were outputs of another classifier?

Features from classifiers



How to convert raw inputs (sensor readings, pixels in image, words in a book) to features, $\mathbf{x}$?

Features from classifiers

x_1

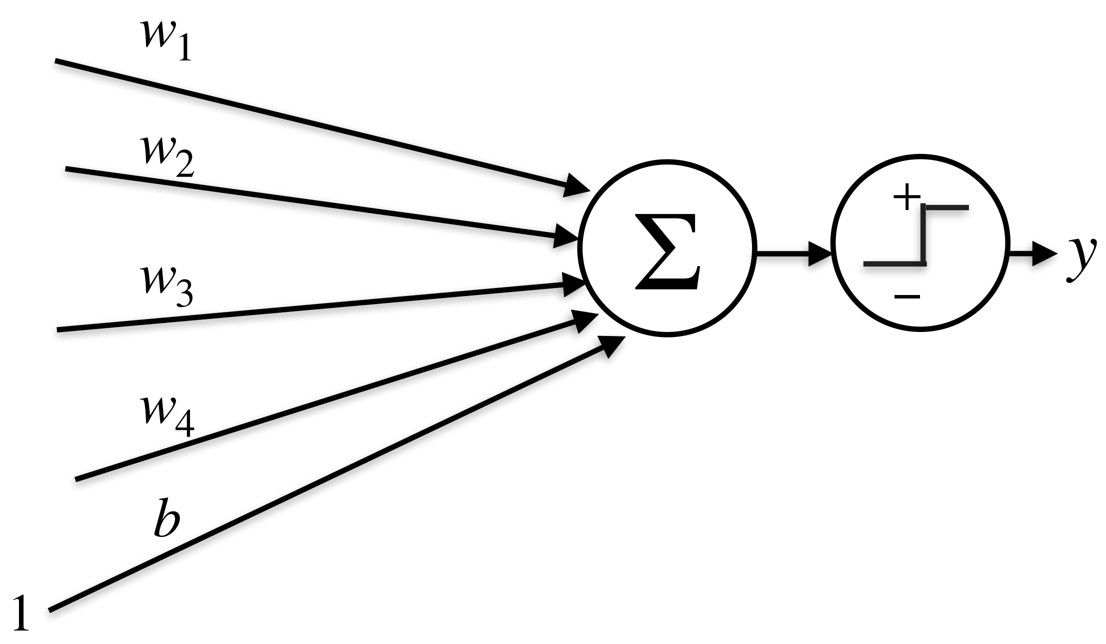
x_2

x_3

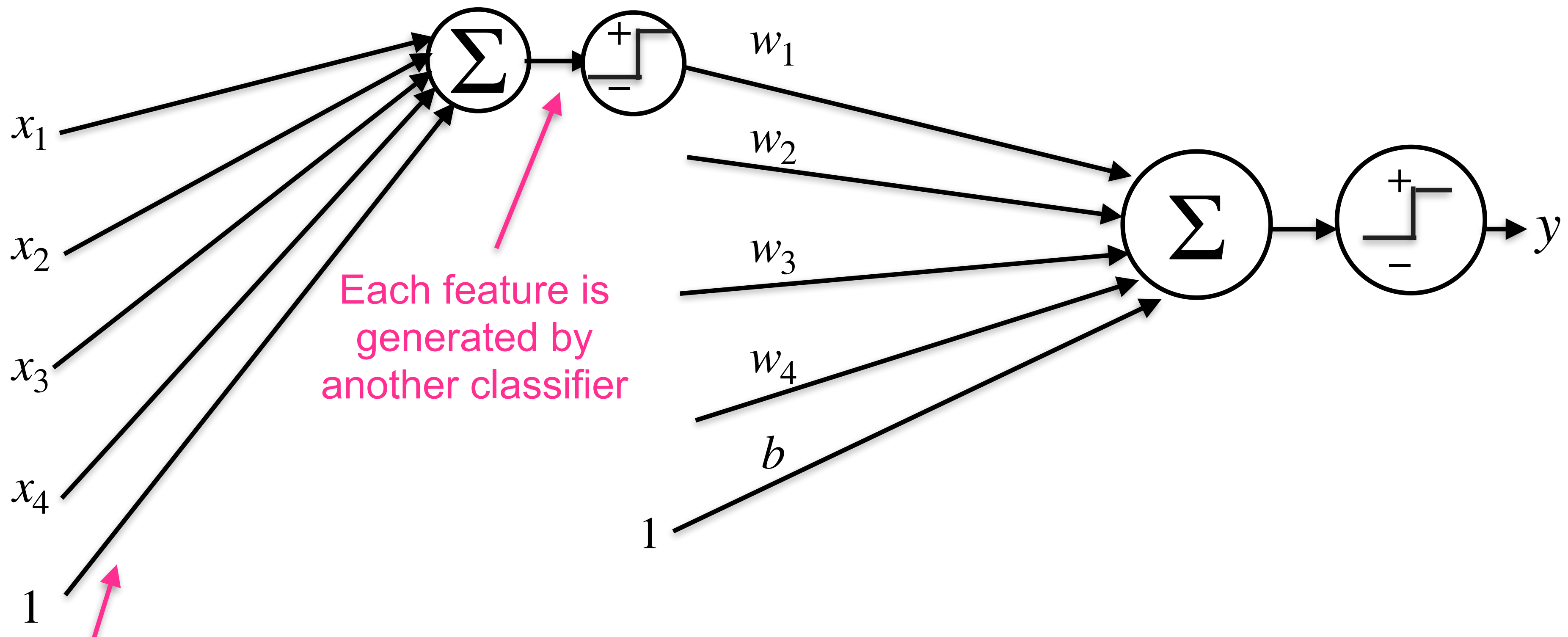
x_4

1

Separate final decision function from raw inputs



Features from classifiers

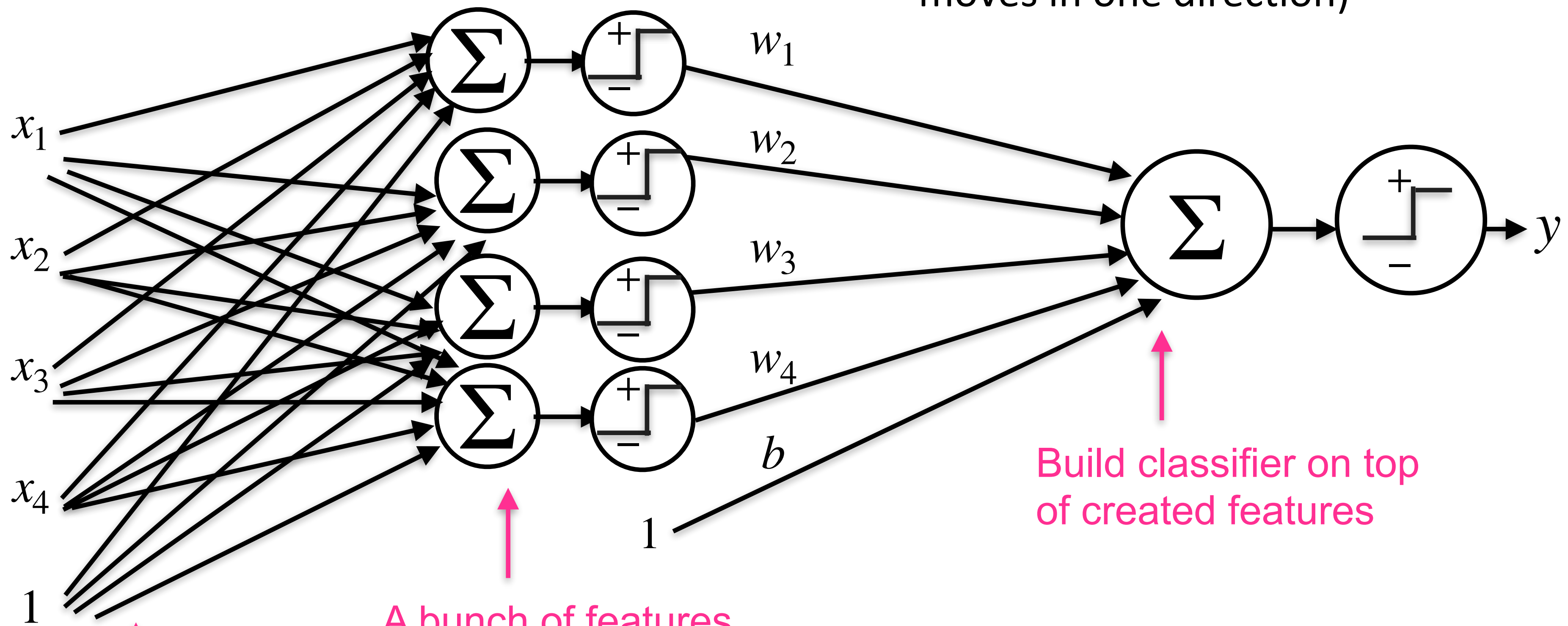


Each feature is generated by another classifier

Each of these connections has its own weight as well

Features from classifiers

This is a two-layer feed forward neural network (i.e., information moves in one direction)



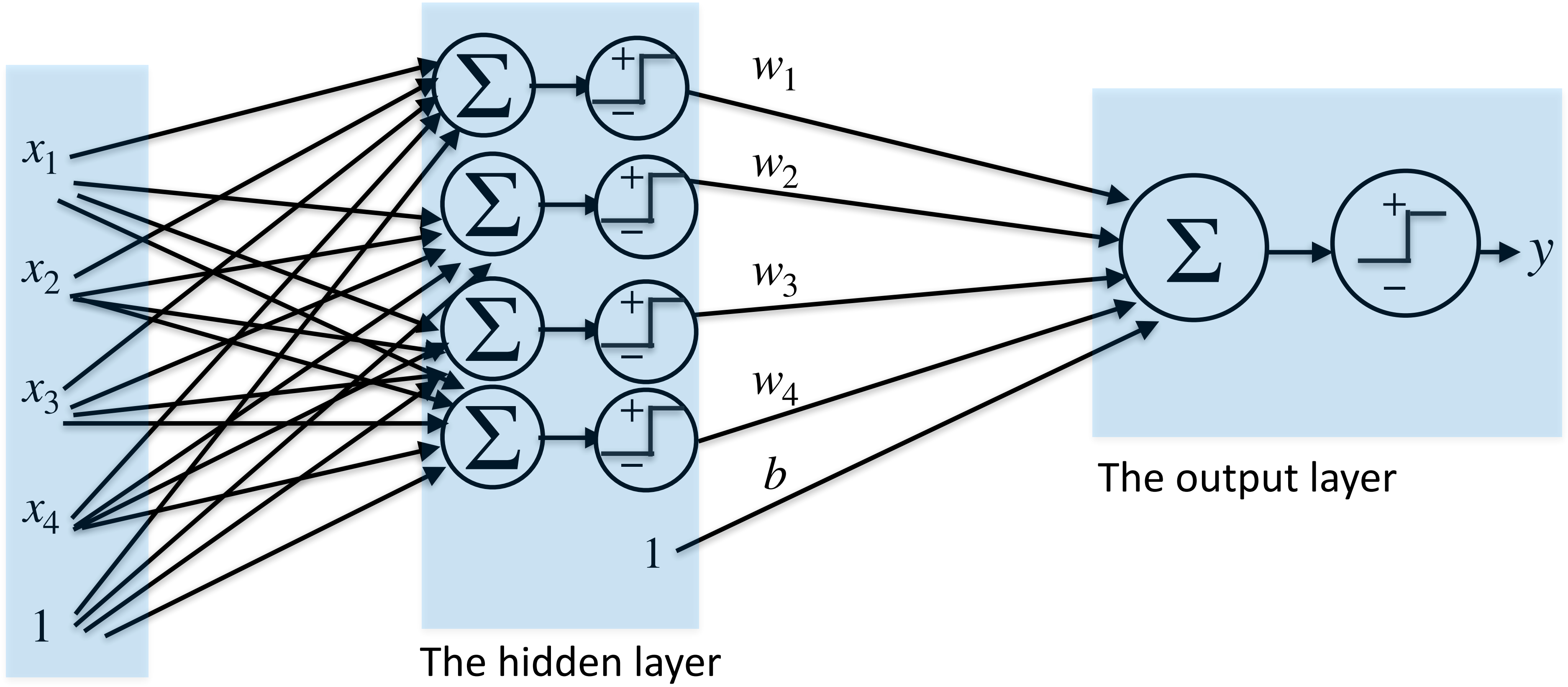
A bunch of features have now been created!

Build classifier on top of created features

These connections are computing feature transformations

Features from classifiers

This is a two-layer feed forward neural network



The input layer

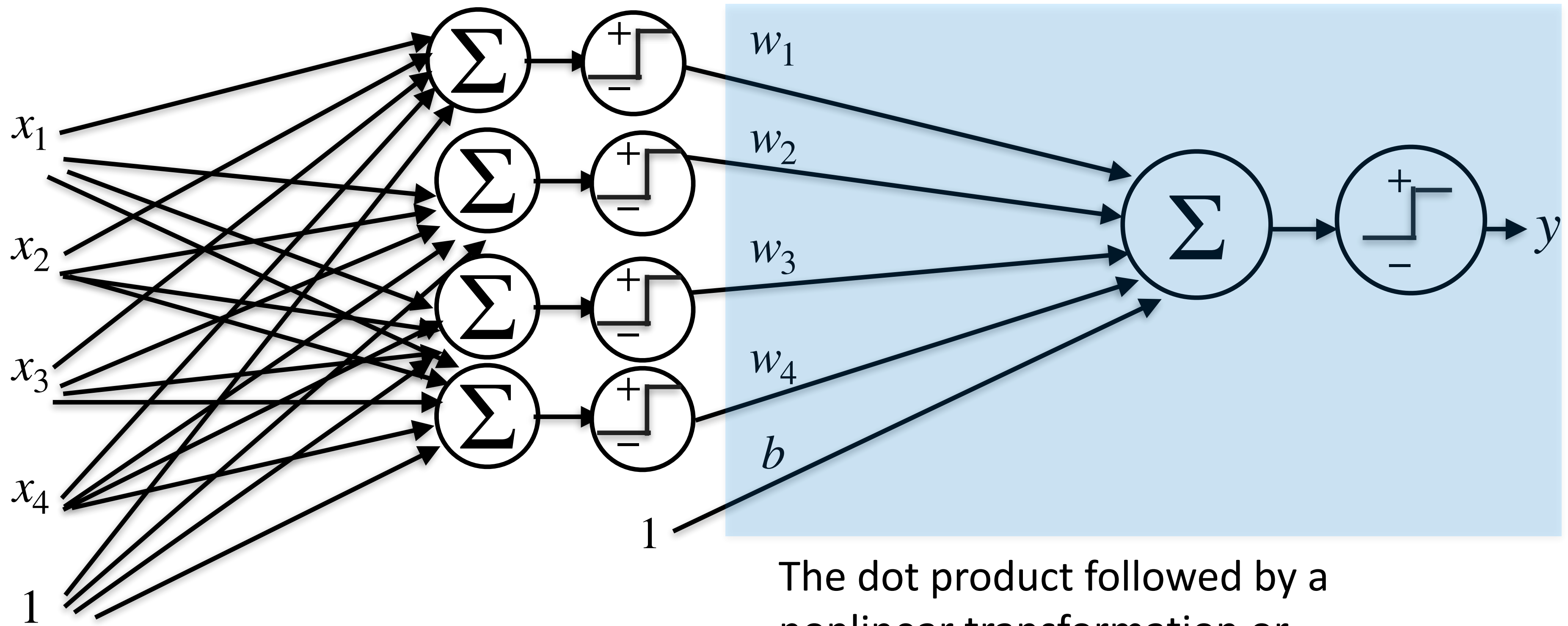
The hidden layer

The output layer

Think of the hidden layer as learning a good representation of the inputs for this particular classification or regression task. For different tasks, hidden layer might be different

Features from classifiers

This is a two-layer feed forward neural network



The dot product followed by a nonlinear transformation or threshold is called a neuron

Five neurons in this picture (four in hidden layer and one output)

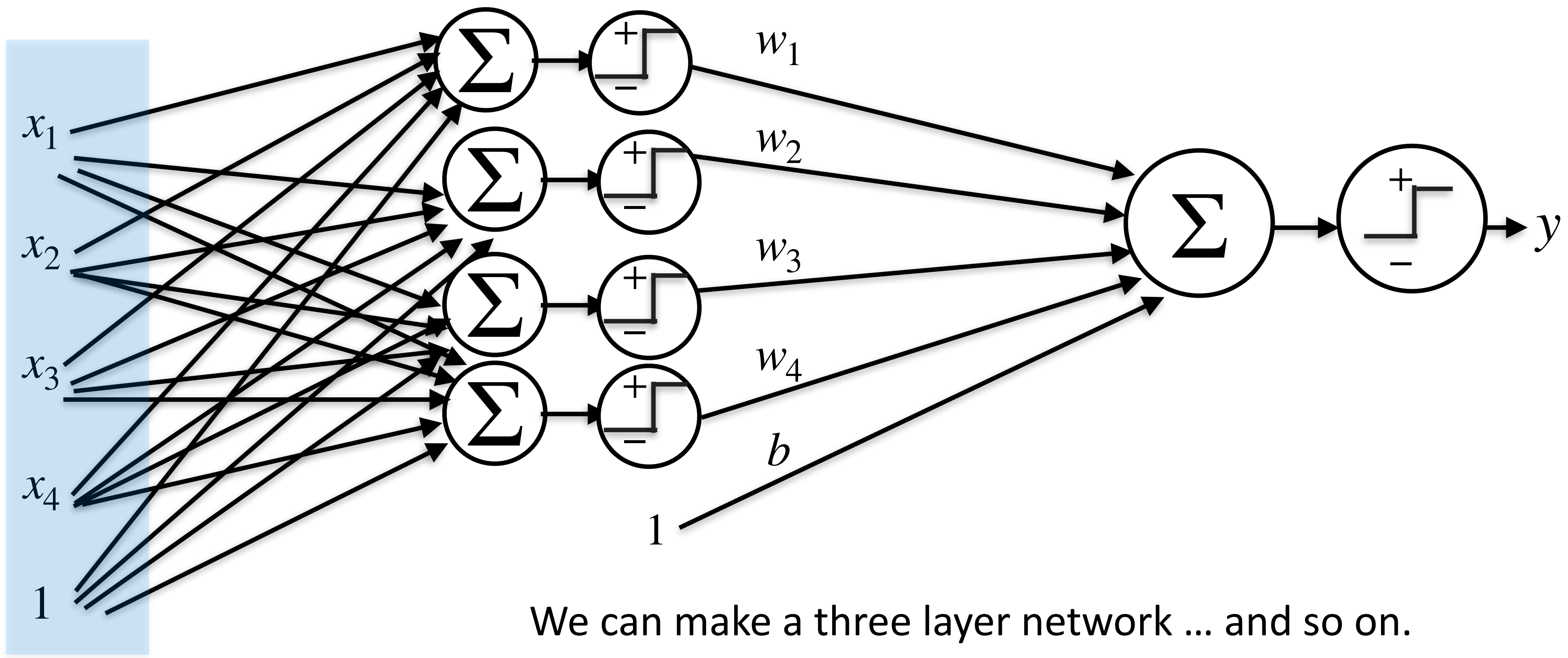
3 min

How do you know what is best hidden layer to learn? That is, how would you learn the best set of weights for edges?

Neural network represents a function. Write down loss function, compute gradient, and minimize using gradient descent as before. May not be convex anymore so no longer guaranteed minima.

Only thing that is complicated here is computing the gradient. Algorithm that does this computation is called backpropagation.

But where do the inputs come from?



The input layer

Question: how do we know these input features are the right ones? Maybe we need another layer

We can make a three layer network … and so on.
Idea: let inputs be as close as possible to raw data and let neural network learn representation of features

OpenAI GPT-3: Neural network with 175 billion parameters

https://arxiv.org/pdf/2005.14165.pdf

Two takeaways about neural networks

- Neural networks can learn feature representations

- Learned feature representations outperform hand-coded representations

… It’s all about the features!