COMP 5630/6630:Machine Learning

Lecture 6: Model Selection, Neural Network

Slide Adapted: Dr. Dhruv Batra, Aakur, Kian Katanforoosh

Regularized Least Squares

- How do you prevent overfitting?

Regularized Least Squares

- How do you prevent overfitting?
- ***Regularization!***
- As model complexity increases, e.g., degree of polynomial or no. of basis functions, then it is likely that we overfit

Regularized Least Squares

- How do you prevent overfitting?
- ***Regularization!***
- As model complexity increases, e.g., degree of polynomial or no. of basis functions, then it is likely that we overfit
- One way to control overfitting is not to limit complexity but to add a regularization term to the error function E_D

$$E(\mathrm{w}) = E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})$$

- where λ is the regularization coefficient that controls relative importance of data-dependent error $E_D(w)$ and regularization term $E_W(w)$

Regularized Least Squares

- Regularized least squares i $E(\mathrm{w}) = E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})$

- Simple form of regularization term is $E_W(\mathrm{w}) = \frac{1}{2}\mathrm{w}^T\mathrm{w}$

- Hence total error is given by $E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(x_n) \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}$

- This regularization function is called *weight decay*
    - Weight values decay towards zero unless supported by training data examples

Regularized Least Squares

- Error function with weight decay (quadratic) regularizer is

$$\boxed{E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\ t_n - \mathrm{w}^T\phi(x_n)\ \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}}$$

Regularized Least Squares

- Error function with weight decay (quadratic) regularizer is

$$\boxed{E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\ t_n - \mathrm{w}^T\phi(x_n)\ \}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}}$$

- Its exact minimizer can be found in closed form and is given by

$$\boxed{\mathrm{w} = (\lambda \mathrm{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathrm{t}}$$

Regularized Least Squares

- Error function with weight decay (quadratic) regularizer is

$$\boxed{E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\, t_n - \mathrm{w}^T\phi(x_n)\,\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}}$$

- Its exact minimizer can be found in closed form and is given by

$$\boxed{\mathrm{w} = (\lambda \mathrm{I} + \Phi^T\Phi)^{-1}\Phi^T\mathrm{t}}$$

Simple extension of ordinary least squares solution

$$\boxed{\mathrm{w}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{t}}$$

A General Regularizer

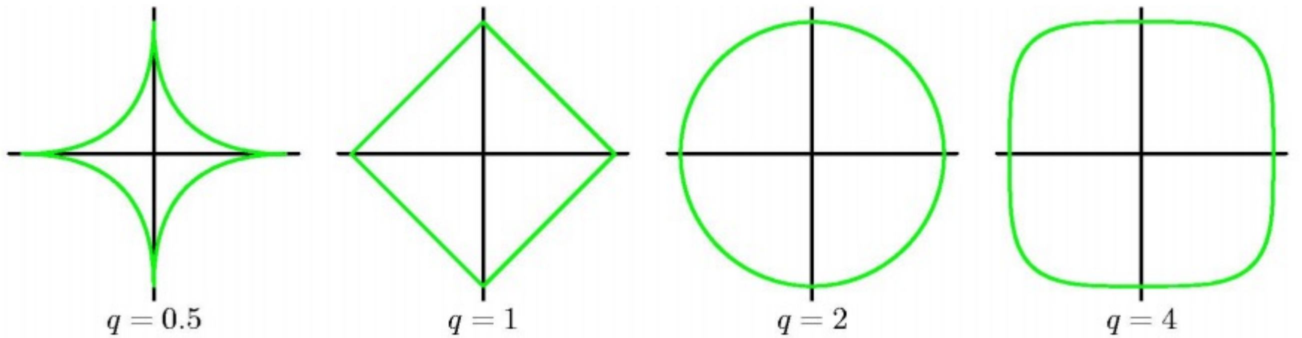
$$\frac{1}{2}\sum_{n=1}^{N}\{\ t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\ \}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|\ w_j\ |^q$$

- q=2 corresponds to the quadratic regularizer
- q=1 is known as lasso
  - Lasso has the property that if λ is sufficiently large some of the coefficients w_j are driven to zero
    - Leads to a sparse model in which the corresponding basis functions play no role

A General Regularizer

$$\frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(\mathrm{x}_n) \right\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|\, w_j \,|^q$$

$\sqrt{w_1} + \sqrt{w_2} = const$ $\quad w_1 + w_2 = const$ $\quad w_1^{\,2} + w_2^{\,2} = const$ $\quad w_1^{\,4} + w_2^{\,4} = const$

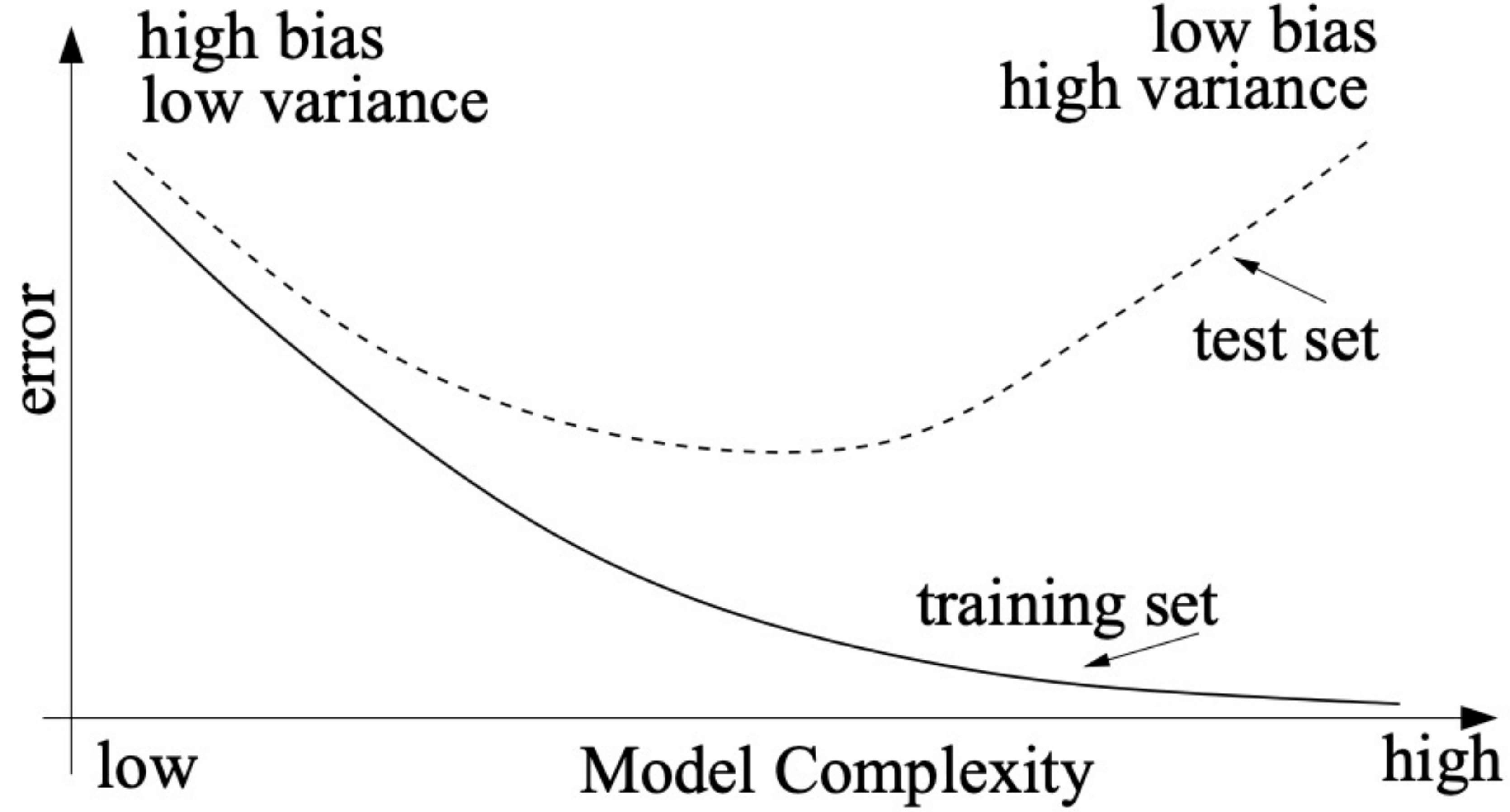


Summary

- Regularization allows
  - complex models to be trained on small data sets
  - without severe over-fitting
- It limits model complexity
  - i.e., how many basis functions to use?
- Problem of limiting complexity is shifted to
  - one of determining suitable value of regularization coefficient

Model Selection Recap

Typical Behaviour



What is Bias and Variance?

- The ***bias error*** is an error from ***erroneous assumptions*** in the learning algorithm. ***High bias*** can cause an algorithm to miss the relevant relations between features and target outputs (***underfitting***).
- The ***variance*** is an error from ***sensitivity to small fluctuations*** in the training set. ***High variance*** can cause an algorithm to ***model*** the ***random noise*** in the training data, rather than the intended outputs (***overfitting***).

Mitigation Techniques

- Model selection
  - Use model assessment to pick the best model
  - Use validation data if available. Otherwise, K-Fold Cross validation
- Add more training data
- Regularization
- Parameter sharing
- Locally weighted or non-parametric models to use less data
- Early stopping

Bias-Variance vs Bayesian

- Bias-Variance decomposition provides insight into model complexity issue
- Limited practical value since it is based on ensembles of data sets
  - In practice there is only a single observed data set
  - If there are many training samples then combine them
    - which would reduce over-fitting for a given model complexity
- Bayesian approach gives useful insights into over-fitting and is also practical

Remember: Curve Fitting

- Regression using basis functions and MSE:

$$y(\mathrm{x}, \mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x}) = \mathrm{w}^T \phi(\mathrm{x})$$

$$E_D(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^{N} \{ t_n - \mathrm{w}^T \phi(\mathrm{x}_n) \}^2$$

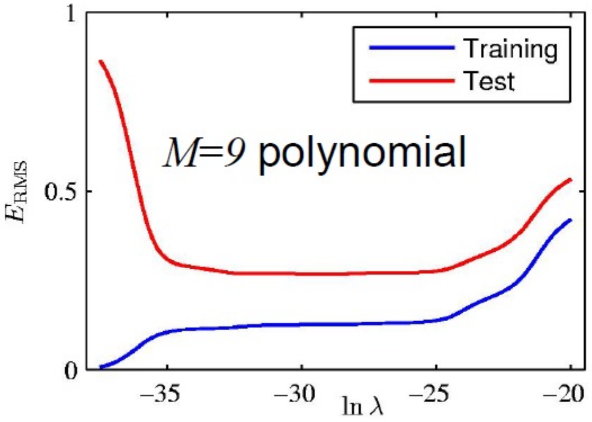
- Need an M that gives best generalization
  - M = No. of free parameters in model or model complexity
- With regularized least squares

$$E(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^{N} \{ t_n - \mathrm{w}^T \phi(x_n) \}^2 + \frac{\lambda}{2} \mathrm{w}^T \mathrm{w}$$

- λ also controls model complexity (and hence degree of over-fitting)

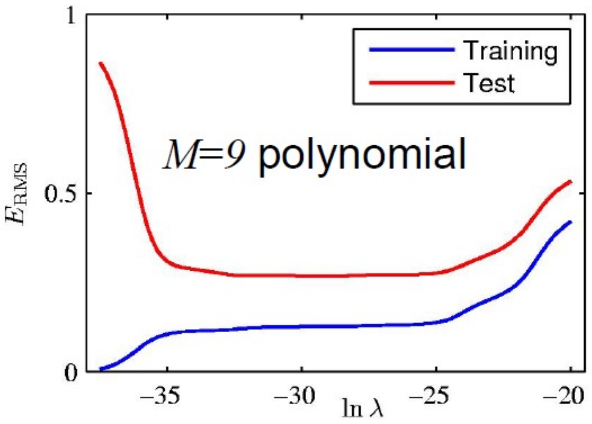
Choosing a model using data

- λ controls model complexity (similar to choice of M)
- Frequentist Approach:
  - Training set
    - To determine coefficients w for different values of (M or λ)
  - Validation set (holdout)
    - to optimize model complexity (M or λ)



Choosing a model using data

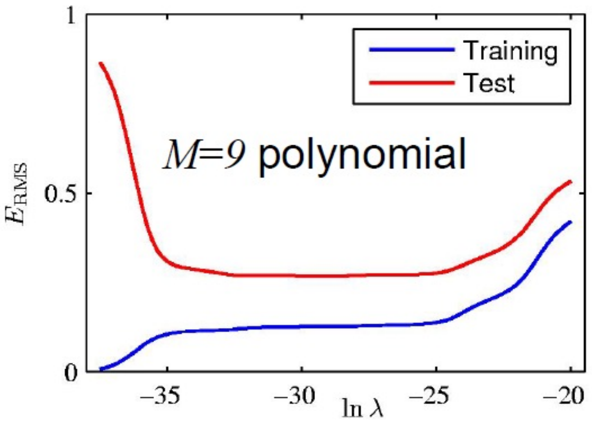
- λ controls model complexity (similar to choice of M)
- Frequentist Approach:
  - Training set
    - To determine coefficients w for different values of (M or λ)
  - Validation set (holdout)
    - to optimize model complexity (M or λ)



What value of λ minimizes error?
λ = -38 : low error on training, high error on testing set
λ = -30 is best for testing set

Choosing a model using data

- λ controls model complexity (similar to choice of M)
- Frequentist Approach:
  - Training set
    - To determine coefficients w for different values of (M or λ)
  - Validation set (holdout)
    - to optimize model complexity (M or λ)



What value of λ minimizes error?
λ = -38 : low error on training, high error on testing set
λ = -30 is best for testing set

$$E_{RMS} = \sqrt{2E(\mathrm{w}^*)/N}$$

Division by N allows different data sizes to be compared since E is a sum over N Sqrt (of squared error) measures on same scale as t

Use the Validation Set!

- Performance on training set is not a good indicator of predictive performance
- If there is plenty of data,
  - use some of the data to train a range of models Or a given model with a range of values for its parameters
  - Compare them on an independent set, called validation set
  - Select one having best predictive performance
- If data set is small then some over-fitting can occur and it is necessary to keep aside a test set

K-fold Cross Validation

- Supply of data is limited
- All available data is partitioned into K groups
- K-1 groups are used to train and evaluated on remaining group
- Repeat for all K choices of held-out group
- Performance scores from K runs are averaged

K=4

If K=N, then it is called the leave-one-out cross validation

Disadvantage of Cross-Validation

- No. of training runs is increased by factor of K
- Problematic if training itself is expensive
- Different data sets can yield different complexity parameters for a single model
  - E.g., for given M several values of λ
- Combinations of parameters is exponential
- Need a better approach
  - Ideally one that depends only on a single run with training data and should allow multiple hyperparameters and models to be compared in a single run

ML Model Selection & Training : Mitigation Technique

ML Model Selection & Training : Mitigation Technique

- Model selection
  - Use model assessment to pick the best model
  - Use validation data if available. Otherwise, K-Fold Cross validation
  - Analytic approach: AIC, Bayesian model selection
- Add more training data
- Regularization
- Parameter sharing
- Locally weighted or non-parametric models to use less data
- Early stopping

ML Training Challenge: Mitigation Technique

- Model selection
  - Analytic approach: Bayesian model selection, AIC
    - Apply AIC and Bayesian model selection in practice
- Model Selection Principle
  - No Free Lunch Theorem
  - Ocham’s Razor
- Model Assessment of Classification
  - Confusion Matrix
  - Performance Measure using a confusion matrix: Accuracy, precision, recall, misclassification rate

ML Training Challenge: Mitigation Technique

- Model selection
  - Analytic approach: Bayesian model selection, AIC
    - Apply AIC and Bayesian model selection in practice

- Model Selection Principle
  - No Free Lunch Theorem
  - Ocham's Razor

- Model Assessment of Classification
  - Confusion Matrix
  - Performance Measure using a confusion matrix: Accuracy, precision, recall, misclassification rate

Bayesian Model Selection: Concepts

- Likelihood
  - How much does a certain hypothesis explain the data?
- Prior
  - What do we believe before seeing any data?
- Posterior
  - What do we believe after seeing the data?

- Bayes Rule = P(A|B) = $\frac{P(A \wedge B)}{P\ (B)} = \frac{P(B|A)P(A)}{P\ (B)}$

Bayes theorem

- It is more intuitive if we use the notation H for hypothesis, and O for observed event.
- Upon the **observance** of O, one goes back and assesses the probability of the causal **hypothesis** H.

$$P(H|O) = \frac{P(H) \cdot P(O|H)}{P(O)}$$

Bayes theorem

- Furthermore, the probability of the observed event P(O) can be written as:

$$P(O) = P\big((O \cap H) \cup (O \cap nonH)\big)$$

$$\mathrm{P}(O) = P(O \cap H) + P(O \cap nonH)$$

$$\mathrm{P(O)} = P(H) \cdot P(O|H) + P(nonH) \cdot P(O|nonH)$$

- **O** may happen if the hypothesis **H** is **true** (first term) or if the hypothesis **H** is **not true** (second term)

Bayes theorem

- Using the value of P(O), we get

$$P(H|O) = \frac{P(H) \cdot P(O|H)}{P(H) \cdot P(O|H) + P(nonH) \cdot P(O|nonH)}$$

Bayes theorem

Example

Let us assume a disease such as cervical cancer, with an prevalence of about 1 case in 5000 women, is to be screened for using a testing procedure that provides positive results for 90% of the women who have the disease (true positives) but also for 0.5% of the women who do not have the disease (false positives). Let **O** be the event of a positive test outcome and **H** the event corresponding of having cervical cancer. What is the probability that a woman who tested positive actually has the disease?

$$P(H|O) = \frac{P(H) \cdot P(O|H)}{P(H) \cdot P(O|H) + P(nonH) \cdot P(O|nonH)}$$

Bayes theorem

- The probability that a woman having a cancer:

$$P(H) = \frac{1}{5000} = 0.0002$$

- The probability of a positive test given that the person is having cancer:

$$P(O|H) = 0.9$$

- The term P(nonH) is the probability of a woman not having cancer:

$$P(nonH) = 1 - P(H) = 0.9998$$

- The term P(O|nonH) is the probability of having a positive test given that there is no cancer:

$$P(O|nonH) = 0.005$$

- The probability of having a positive test:

$$P(0) = P(H) \cdot P(O|H) + P(nonH) \cdot P(O|nonH)$$

- The probability for a woman who tested positive actually to have the disease is:

$$P(H|O) = \frac{0.0002 \cdot 0.9}{0.0002 \cdot 0.9 + 0.9998 \cdot 0.005} = 0.034$$

How do you choose a model?

- Based on its evidence!
- A model is a probability distribution over data D – E.g., a polynomial model is a distribution over target values *t* when input x is known, i.e., p(t|x,D)
- Uncertainty in model itself can be represented by probabilities
- Compare a set of models Mi , i=1,…,L
- Given training set, wish to evaluate posterior

$$p(M_i|D) \quad \alpha \quad \underbrace{p(M_i)}\ \underbrace{p(D|M_i)}$$

Prior expresses preference
for different models

We assume equal priors

Model Evidence
(or Marginal Likelihood)

Bayesian Model Selection: Steps

- Given two models, M_1 and M_2, choose the best one based on observed data, D.
- Compute Bayes factor, K

$$K(M_1, M_2) = \frac{P(D|M_1)}{P(D|M_2)} = \frac{P(M_1|D)P(D)}{P(M_1)} / \frac{P(M_2|D)P(D)}{P(M_2)} = \frac{P(M_1|D)}{P(M_2|D)}\frac{P(M_2)}{P(M_1)}$$

- K measure the degree to which the data alter our belief to support M_1 over M_2

Akaike Information Criterion (AIC)

- AIC $= 2k - 2(\log likelihood)$, k = number of estimated parameters (Θ)
- Choose the model with lowest AIC

- AIC Formula Interpretation
  - *If model complexity increases (e.g., k increases), how does the AIC change?*
  - *If k increases, how does $2(log\ likelihood)$ change?*

Akaike Information Criterion (AIC)

- AIC $= 2k - 2(\log likelihood)$, k = number of estimated parameters (Θ)
- Choose the model with lowest AIC
- AIC Formula Interpretation
  - If model complexity increases (e.g., k increase), how does the AIC change? AIC Increases
  - If k increases, how does $2(\log likelihood)$ change?
    - Model fits better, $2(\log likelihood)$ increases, - $2(\log likelihood)$will get smaller
- Bayesian Information Criterion (BIC) is a variant of this quantity
- Disadvantages:
  - need runs with each model, prefers overly simple models

ML Training Challenge: Mitigation Technique

- Model selection
  - Analytic approach: Bayesian model selection, AIC
    - Apply AIC and Bayesian model selection in practice
- Model Selection Principle
  - No Free Lunch Theorem
  - Ocham's Razor
- Model Assessment of Classification
  - Confusion Matrix
  - Performance Measure using a confusion matrix: Accuracy, precision, recall, and misclassification rate

Model Selection: No Free Lunch Theorem

No Free Lunch

- David Wolpert and others have proven a series of theorems, known as the "no free lunch" theorems which, roughly speaking, say that *unless you make some assumptions* about the nature of the functions or densities you are modeling, no one learning algorithm can *a priori* be expected to do better than any other algorithm.
- In particular, this lack of clear advantage includes any algorithm and any meta-learning procedure applied to that algorithm. In fact, "anti-cross-validation" (i.e. picking the regularization parameters that give the *worst* performance on the CV samples) is a priori just as likely to do well as cross-validation. Without assumptions, random guessing is no worse than any other algorithm.
- So capacity control, regularlization, validation tricks and meta-learning (next class) cannot *always* be successful.

Model Selection: Ocham’s Razor

- Pick the simplest model that explain the data well

ML Training Challenge: Mitigation Technique

- Model selection
  - Analytic approach: Bayesian model selection, AIC
    - Apply AIC and Bayesian model selection in practice
- Model Selection Principle
  - No Free Lunch Theorem
  - Ocham’s Razor
- Model Assessment of Classification
  - Confusion Matrix
  - Performance Measure using a confusion matrix: Accuracy, precision, recall, and misclassification rate

Confusion Matrix: Measuring Classifier Performance

- Used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.

Confusion Matrix Elements

- Used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion Matrix: Element Definition

- Used to **describe the performance of a classification model** (or "classifier") on a set test data for which the true values are known.
- Models four major characteristics
  - **true positives (TP)**
  - **true negatives (TN)**
  - **false positives (FP)**
  - **false negatives (FN)**

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

- When the total of TP is not close to the total of TN, the dataset is *imbalanced*

Perf. Measures from Confusion Matrix

- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total
- **Recall:** When it's actually yes, how often does it predict yes?
  - also known as "Sensitivity"

$$\textit{Recall} = \frac{TP}{TP + FN}$$

- **Precision:** When it predicts yes, how often is it correct?

$$\textit{Precision} = \frac{TP}{TP + FP}$$

Design Question

- You are designing a software to assist college admission system. One criteria is predicting the dropout rate of candidate based on the candidate's profile. You are testing our predictive system on previous students' records.

  - Which metric is a desirable characteristic of your predictive system system and why?
    - High Precision
    - High Recall

Introduction to Neural Networks

Introduction to Neural Network

- The Perceptron algorithm for binary classification
  - How perceptron is different from logistic regression (LR)
  - Training steps a perceptron
  - Extending LR to multiclass LR
    - Extending perceptron to multilayer perceptron

- Forward Pass
  - Defining propagation equations
  - Defining loss function

- How to train
- Backward propagation

Introduction to Neural Network

- The Perceptron algorithm for binary classification
  - How perceptron is different from logistic regression (LR)
  - Training steps a perceptron
  - Extending LR to multiclass LR
    - Extending perceptron to multilayer perceptron

  - Forward Pass
    - Defining propagation equations
    - Defining loss function

  - How to train
  - Backward propagation

The Perceptron: Supervised Binary Classifier

- Task: Predict whether an input image contains a cat (1) or not cat (0)
- Consider modifying the logistic regression method to "force" it to output values that are either 0 or 1 or exactly.
  - Change the definition of g to be the threshold function of logistic regression:

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- Let $h(x) = g(\theta^T x)$ as before but using this modified definition of g, and if we use the update rule

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

The Perceptron Algorithm: Difference w.r.t LR

- In the 1960s, this "perceptron" was argued to be a rough model for how individual neurons in the brain work.
- Although the perceptron may be similar to the other algorithms we have seen, it is actually a very different type of algorithm than logistic regression.
  - The hypothesis in logistic regression provides a measure of uncertainty in the occurrence of a binary outcome based on a linear model.
  - The output from a *step function* can of course not be interpreted as any kind of probability.
  - Since a step function is *not differentiable*, it is not possible to train a perceptron using the same algorithms that are used for logistic regression.

Introduction to Neural Network

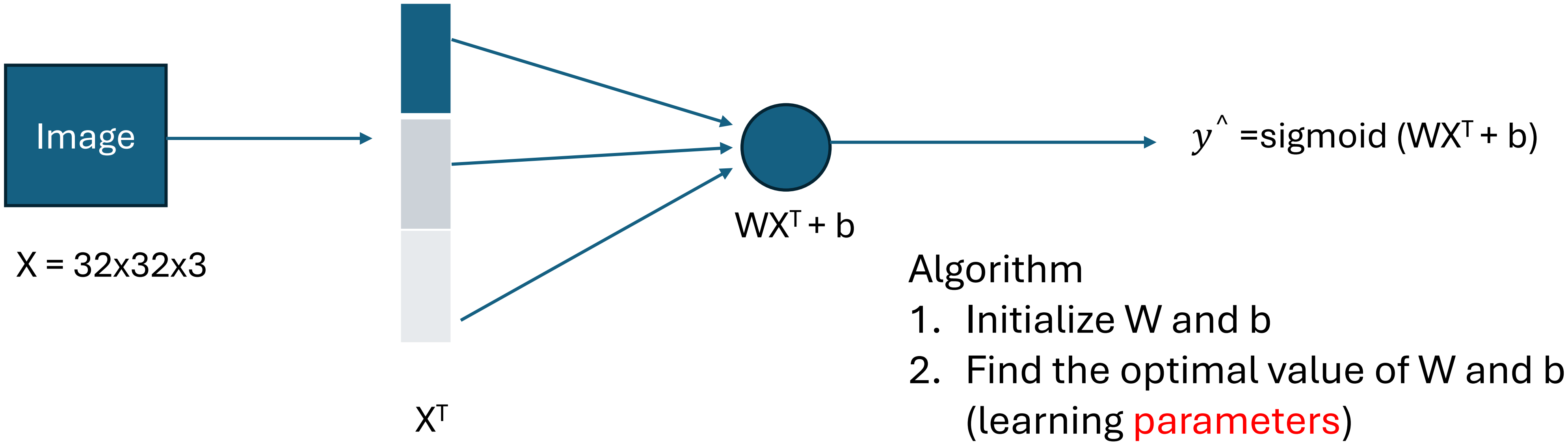
- The Perceptron algorithm for binary classification
  - How perceptron is different from logistic regression (LR)
  - Training steps a perceptron
  - Extending LR to multiclass LR
    - Extending perceptron to multilayer perceptron

  - Forward Pass
    - Defining propagation equations
    - Defining loss function

  - How to train
  - Backward propagation

The Perceptron Algorithm: Training Idea

- Task: Predict whether an input image contains a cat (1) or not (0)
- Image representation from Lecture 2: Linear Classifier



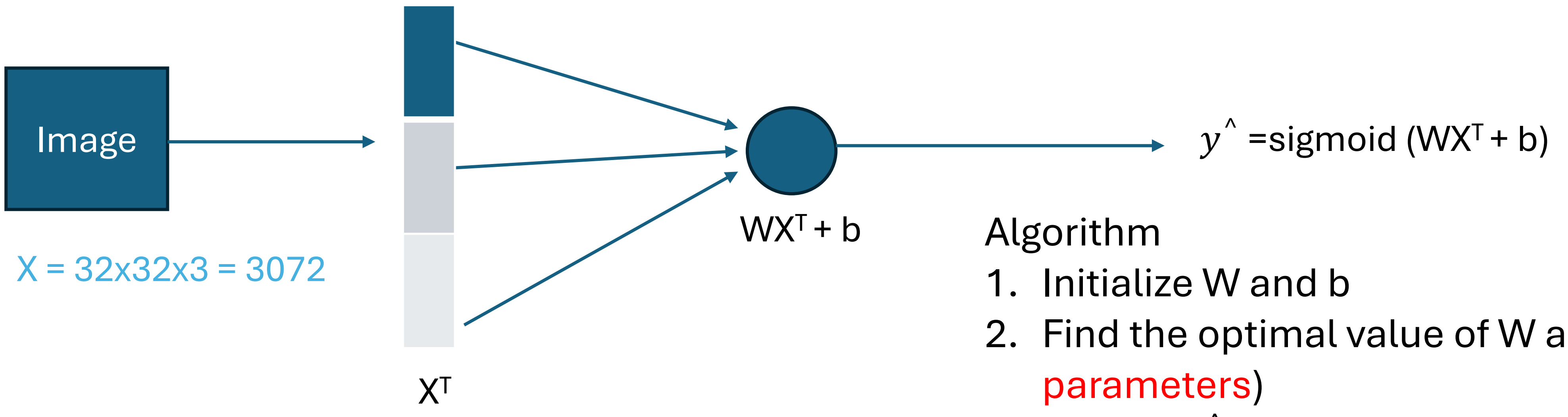
Algorithm

1. Initialize W and b
2. Find the optimal value of W and b (learning parameters)
3. Predict $\hat{y}$

How many parameters are in this model?

The Perceptron Algorithm

- Task: Predict whether an input image contains a cat (1) or not cat (0)
- Image representation from Lecture 2: Linear Classifier



Algorithm

1. Initialize W and b
2. Find the optimal value of W and b (learn parameters)
3. Predict $\hat{y}$

How many parameters are in this model?

$W = 1x\ 3072$, $X^T = 3072x\ 1$, $b = 1$

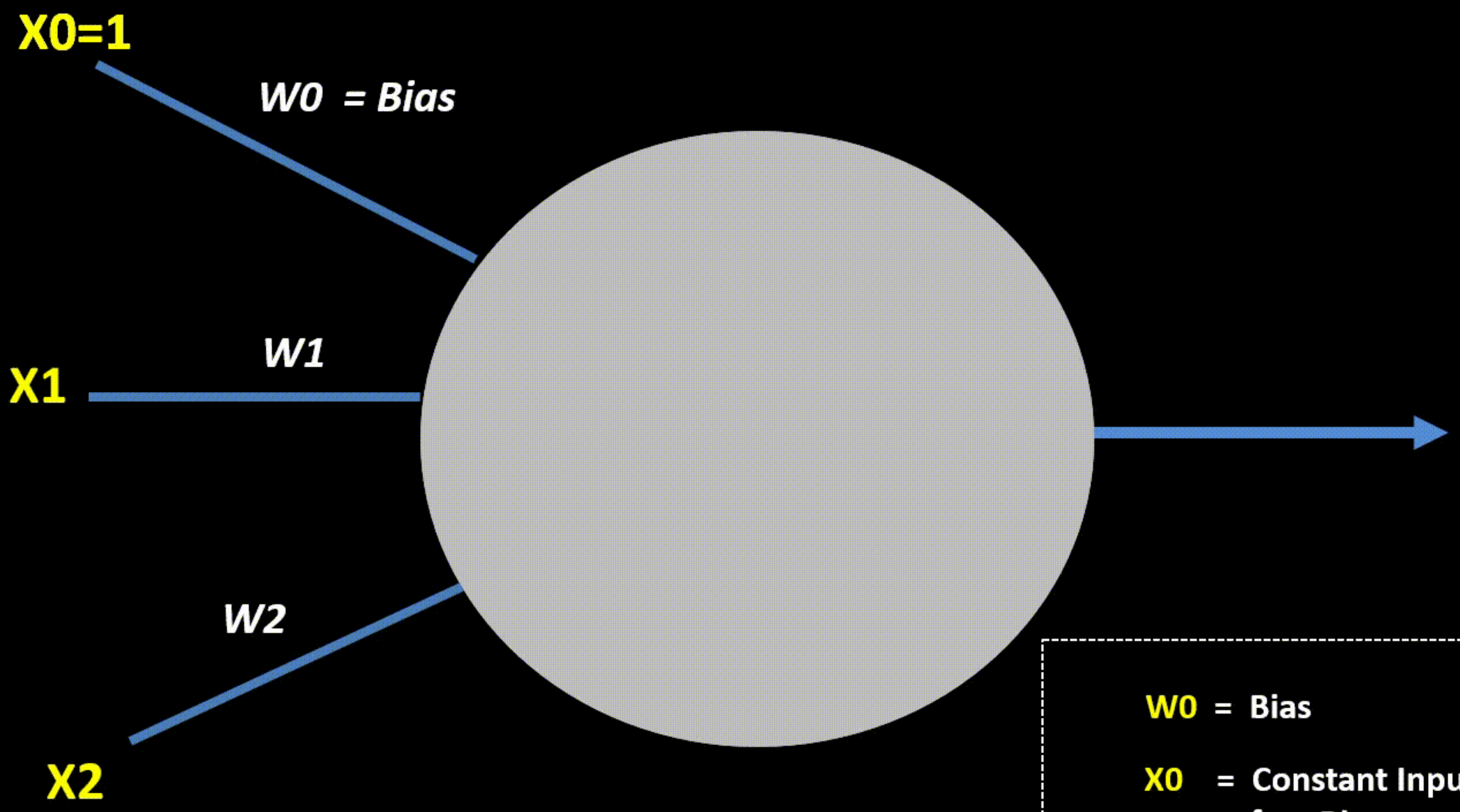
Parameters = 3072 + 1

Vocabulary of Neural Network

- Neuron = linear + activation
  - In our example
    - Linear= $WX^T + b$
    - Activation= sigmoid on the linear output

- Model: architecture + parameter
  - In our example
    - Model = logistic regression
    - Parameter = 3073

Artificial Neuron



W0 = Bias

X0 = Constant Input 1 for Bias

X1,X2 = Attribute Inputs

W1,W2 = Weights of Inputs X1,X2

machinelearningknowledge.ai

Vocabulary of Neural Network: Activation Functions

- Neuron = linear + activation (non-linear)
- The z_j is a linear component, corresponds to outputs of inputs from previous layer
  - or input layer of network
- Each activation a_j transforms z_j using differentiable nonlinear activation functions

- Three examples of activation functions:

1. Logistic sigmoid
2. Hyperbolic tangent
3. Rectified Linear unit

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

$$f(x) = \max(0, x)$$

Activation Functions: Examples

- Task: Predict the age of the animal of the image
  - *What changes will you make in the previous network and why?*

Activation Functions: Examples

- Task: Predict the age of the animal of the image

  - What changes will you make in the previous network?

  - Sigmoid activation function will not be a valid choice
    - ReLU: the age is non-negative
    - The identity or linear activation function

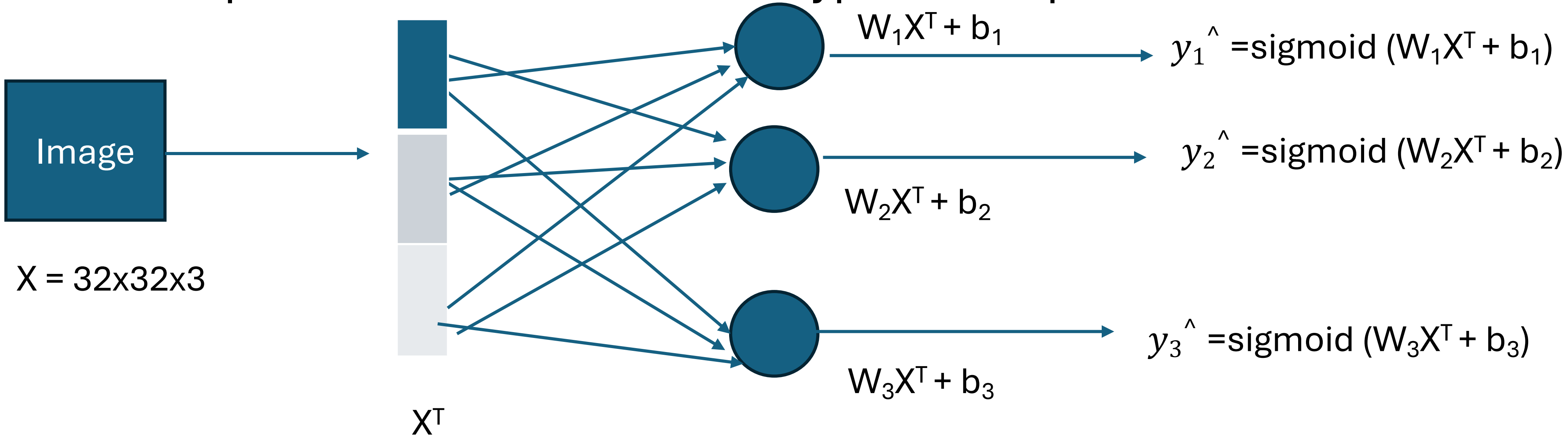
Introduction to Neural Network

- The Perceptron algorithm for binary classification
  - How perceptron is different from logistic regression (LR)
  - Training steps a perceptron
  - Extending LR to multiclass LR
    - Extending perceptron to multilayer perceptron
  - Forward Pass
    - Defining propagation equations
    - Defining loss function
  - How to train
  - Backward propagation

Multiclass Logistic Regression & Neural Network

Task: Predict whether an input image contains a cat/sheep/dog. *Image may contain multiple types of animals.*

Extend the previous network for three types of outputs



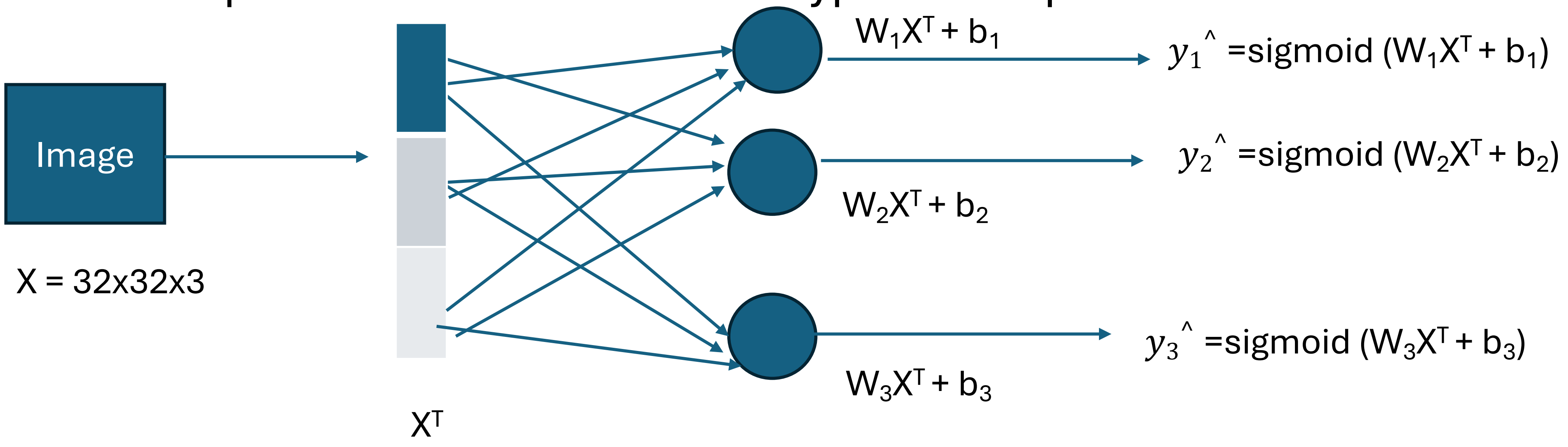
Assignment 1 question answer hint

How many parameters are in this model?
Can this network classify if an image contains BOTH cat and dog?

Multiclass Logistic Regression & Neural Network

Task: Predict whether an input image contains a cat/sheep/dog. *Image may contain multiple types of animals.*

Extend the previous network for three types of outputs



Assignment 1 question answer hint

How many parameters are in this model? = 3 x prev. model

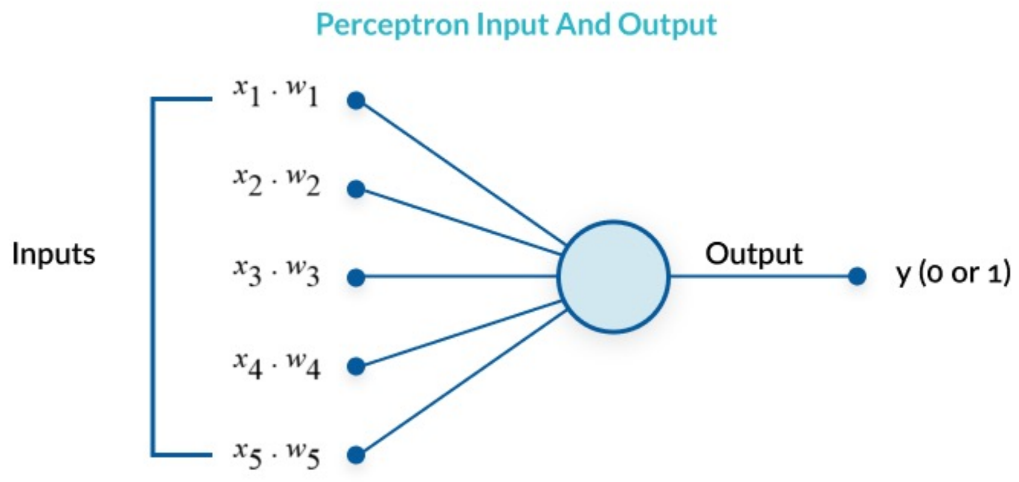
Can this network classify if an image contains BOTH cat and dog? YES. As each neuron is independent

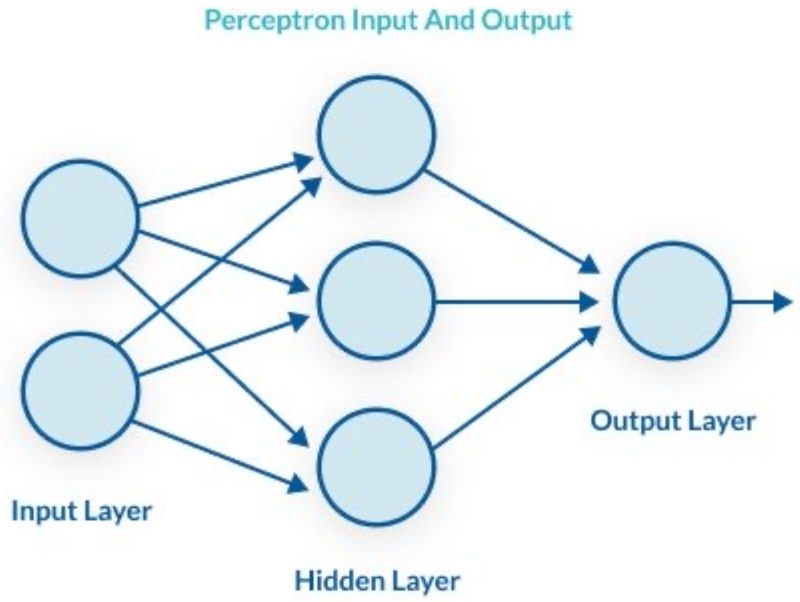
Activation Function Selection

- Determined by the nature of the data and the assumed distribution of the target variables
- For standard regression problems the activation function is the identity function so that $y_k = a_k$
  - *(e.g. predicting the age of the animal in an input image)*
- For multiple binary classification problems, each output unit activation is transformed using a logistic sigmoid function so that $y_k = \sigma(a_k)$
  - *(e.g. predicting whether an input image contains a cat/sheep/dog. Image may contain multiple types of animals)*
- For multiclass problems, a softmax activation function of the form:

$$\boxed{\frac{\exp(a_k)}{\sum_j \exp(a_j)}}$$

Perceptron vs Multilayer Perceptron (MLP)

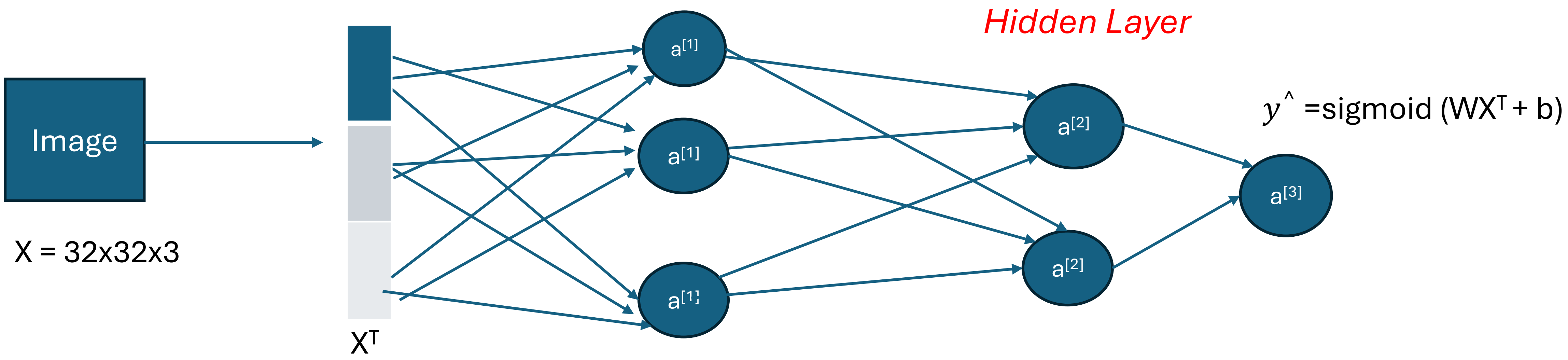




Why do we need more layers?

Multilayer Perceptron

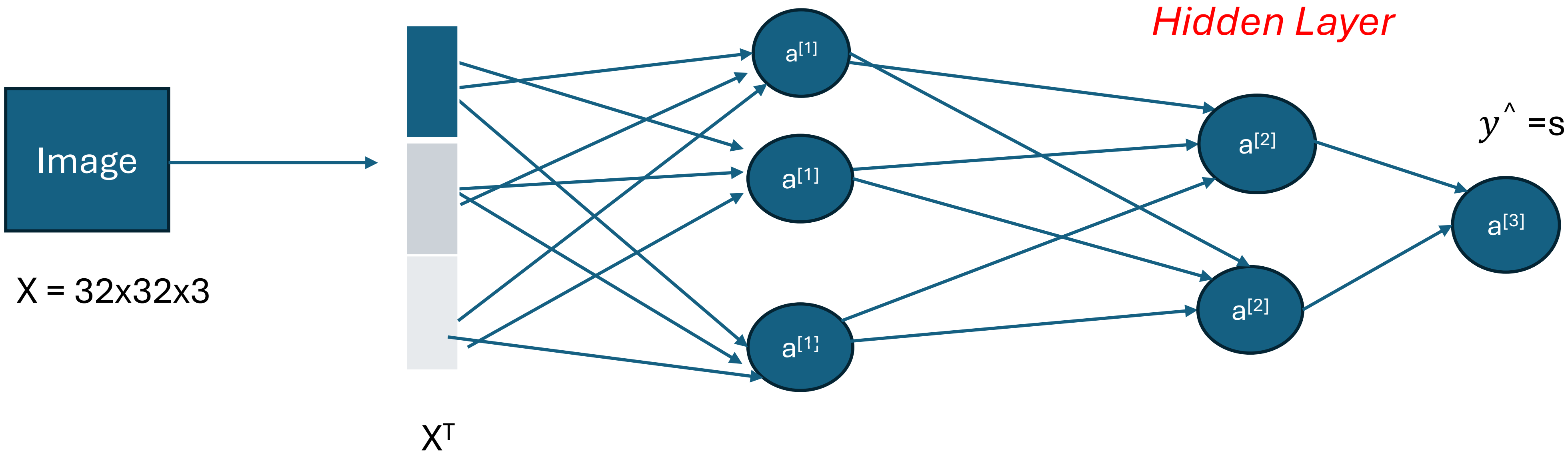
Task: Predict whether an input image contains a cat or not



- The first layer processes the input.
- The two neuron in the hidden layer may identify two different features of a cat, such as eye color or size of the head. We don’t know how the layers operate (thus called hidden layer).
- Assumption: given enough data and adding more layers, the network can accurately identify the image

Multilayer Perceptron

Task: Predict whether an input image contains a cat or not

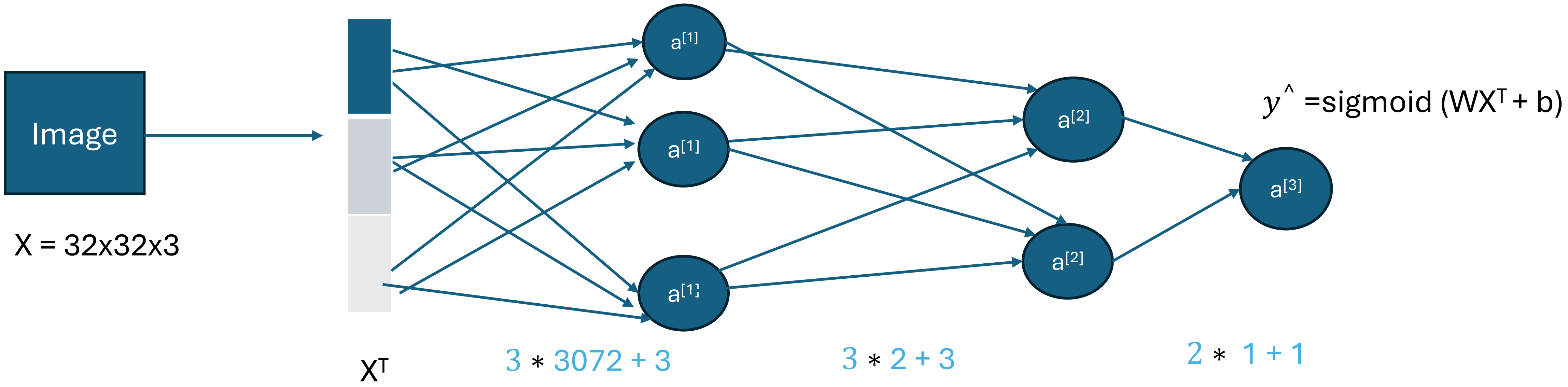


$\hat{y}$ =sigmoid (WX^T + b)

How many parameters are in this model?

Multilayer Perceptron

Task: Predict whether an input image contains a cat or not



How many parameters are in this model?

Introduction to Neural Network

- The Perceptron algorithm for binary classification
  - How perceptron is different from logistic regression (LR)
  - Training steps a perceptron
  - Extending LR to multiclass LR
    - Extending perceptron to multilayer perceptron

  - Forward Pass
    - Defining propagation equations
    - Defining loss function

  - How to train
  - Backward propagation

Notations and Forward Pass

- Output of a layer l is given by $$z^{[\ell]} = W^{[\ell]}a^{[\ell-1]} + b^{[\ell]}$$

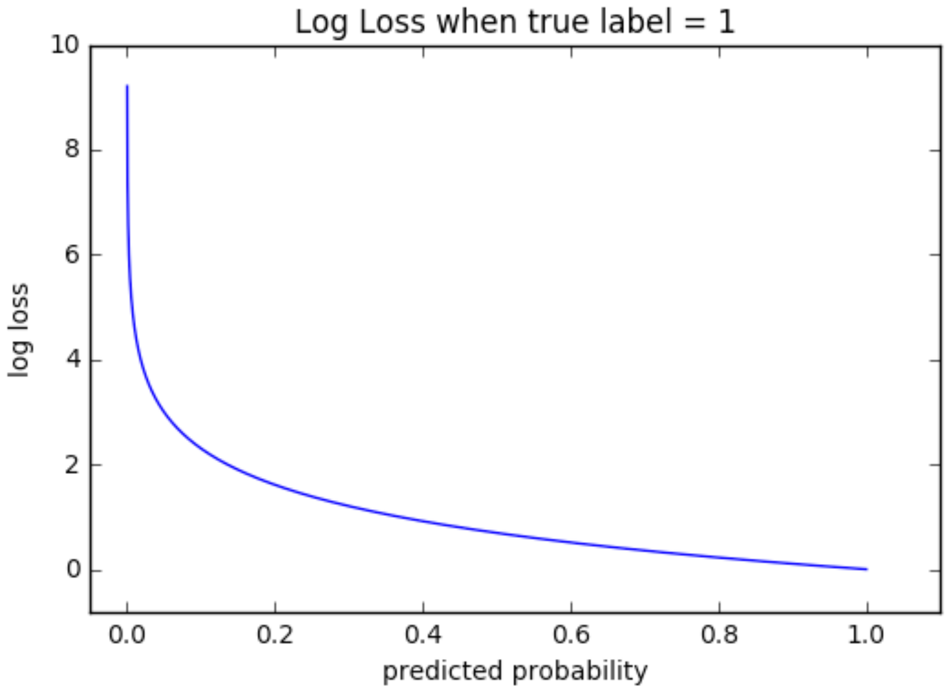
- Where W^l is the weights of the layer, b^l is the bias and a^l is the activation $$a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$$

- Neuron = linear ($z^{[l]}$)+ activation ($a^{[l]}$)

Loss Function: Log loss

- Measures the performance of a classification model whose output is a probability value between 0 and 1.

$$-(y \log(p) + (1 - y) \log(1 - p))$$



- Cross-entropy loss increases

as the predicted probability

diverges from the actual label.

How to Train this MLP Network?

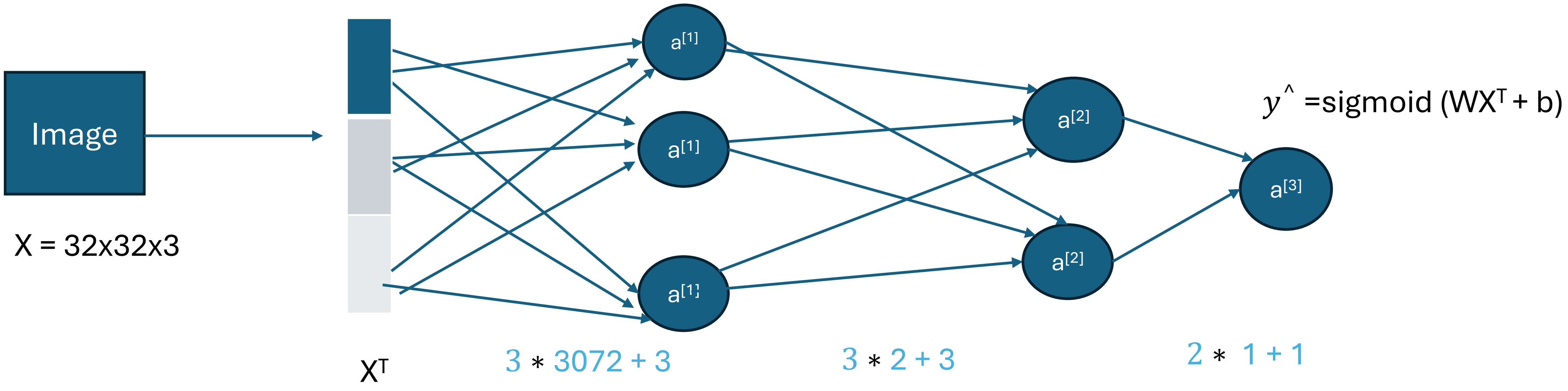
- Gradient Descent + Backpropagation!

- What is Backpropagation

  - Backprop or Backpropagation is a way to train multilayer neural networks using gradients (next class).

Task: How to Train this MLP Network

Task: Predict whether an input image contains a cat or not



Defining Propagation Equations

- $X = [n \times f]$
- $b = [n \times 1]$

- First layer: W_1, bias = b_1
- Second layer : W_2, bias = b_2
- Third layer: W_3, bias = b_3

The Forward Pass

- Output of layer 1: $z^{[1]} = W^{[1]}x + b^{[1]}$

  $a^{[1]} = g(z^{[1]});$ $a^{[1]}$ = activation function

- Output of layer 2: $z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

  $a^{[2]} = g(z^{[2]});$ $a^{[2]}$ = activation function

- Output of layer 3: $z^{[3]} = W^{[3]} a^{[2]} + b^{[3]}$

  $a^{[3]} = g(z^{[3]});$ $a^{[3]}$ = activation function

- Output of neural network: $y^{\wedge} = a^{[3]}$

Defining the Loss

- Task: Binary classification
- Loss: binary cross entropy or log loss
- L = $-[(1-y)\log(1-\ y^{\wedge}\ )\ +\ y.\log y^{\wedge}]$