Final Exam Review

Aaron J. Masino, PhD
Associate Professor

Exam logistics

- Location: Academic Success Center, Room 118
- Date / time: May 2nd, 7:00-9:30 PM
- Mode: Canvas (Bring your laptop!!!)
- The test is closed-book. Internet and AI resources are not allowed.
- Python terminal (IPython), calculator, calculator app may be used.

Coding?

- There are NO coding problems (i.e., you will NOT need to write code)
- There ARE code interpretation problems
- Review the labs

Example: What is the value of *z* in the code below?

```
x = numpy.array([1, 2, 3])
y = numpy.array([5, 1, 0])
z = x + y
```

A. [1, 2, 3, 5, 1, 0]
B. [[1, 2, 3], [5, 1, 0]]
C. [6, 3, 3]

Python programming

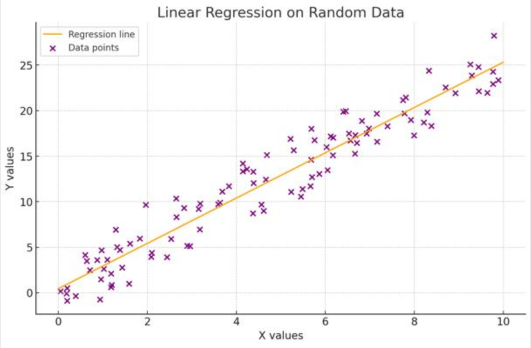
- Data structures
  - Core Python: lists, sets, dictionaries
  - Numpy Arrays
  - PyTorch Tensors
  - Pandas DataFrame
- Index slicing for lists, numpy arrays, PyTorch Tensor
  - How many elements in `l[x:x+7]`
- Numpy and PyTorch broadcasting
- PyTorch matrix operations: transpose, multiplication

Machine learning concepts

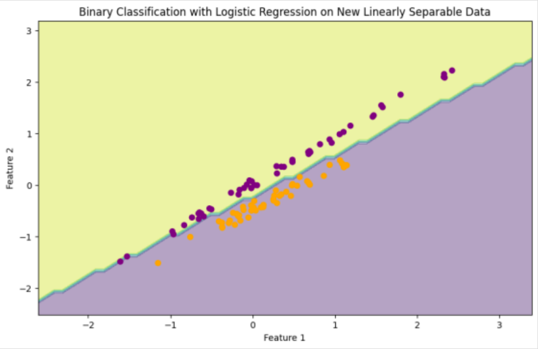
- Supervised vs. unsupervised learning.
  - Which one requires labeled outcomes for training?
  - Tasks

Supervised

Regression

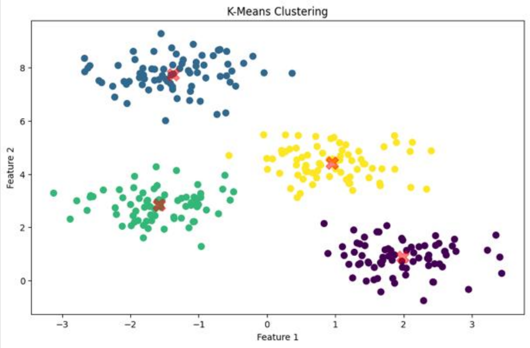


Classification

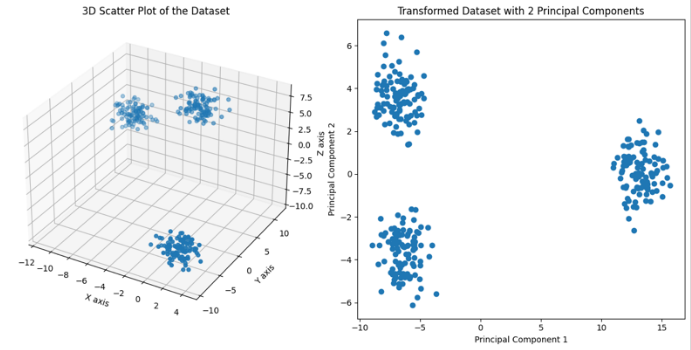


Unsupervised

Clustering

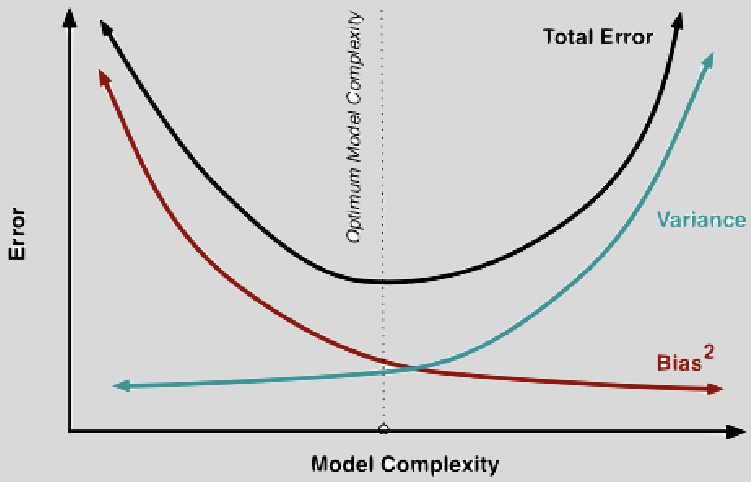


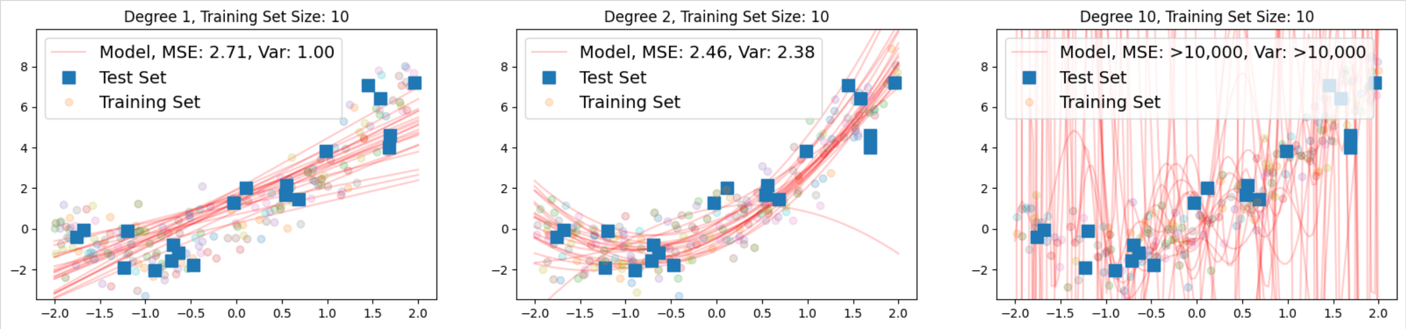
Dimensionality Reduction



Machine learning concepts

- Bias vs variance





- Model selection
    - K-fold CV
    - Bootstrapping
    - Train / validation / test

Linear Regression

- Model form

$$\hat{y} = \beta_0 + \beta_1 * age + \beta_2 * height + \beta_3 * weight$$

- Assessing model fit
  - R^2 – the proportion of explained variance. Range [0, 1]. Higher is better. What happens if we add predictor variables?
  - **F-statistic** – used to assess likelihood of the null hypothesis

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

  - **Coefficient standard error** – average deviation of coefficient estimate from the actual coefficient
  - **t-statistic** - For a given coefficient, it is a test of the null hypothesis that the individual coefficient is zero

Linear Regression

Python StatsModels library

Summary data for multiple linear regression:

OLS Regression Results

Dep. Variable:	PickupCount	R-squared:	0.382
Model:	OLS	Adj. R-squared:	0.380
Method:	Least Squares	F-statistic:	205.0
Date:	Fri, 27 Sep 2024	Prob (F-statistic):	1.58e-103
Time:	14:57:21	Log-Likelihood:	-4131.3
No. Observations:	1000	AIC:	8271.
Df Residuals:	996	BIC:	8290.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	41.3965	1.990	20.801	0.000	37.491	45.302
x1	-9.2296	0.714	-12.928	0.000	-10.631	-7.829
x2	1.0189	0.069	14.735	0.000	0.883	1.155
x3	-0.0267	0.002	-14.090	0.000	-0.030	-0.023

What do these numbers mean?

Our model:

$$\hat{y} \approx 41.40 - 9.23 * \mathrm{x1} + 1.02 * \mathrm{x2} - 0.03 * \mathrm{x3}$$

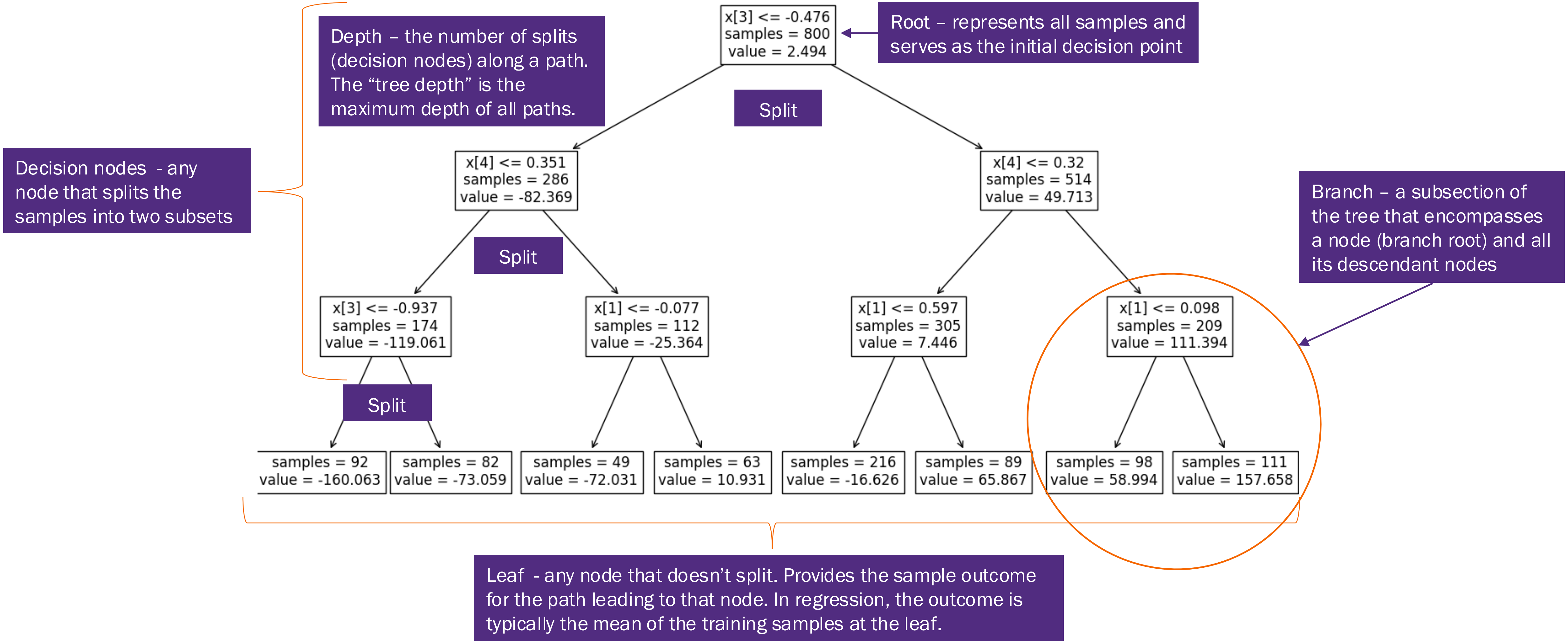
Logistic Regression

- Binary classification model

$$P(Y=1) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

- We classify all observations for which $\hat{P}(Y=1) \geq \gamma$ to be in the group associated with $Y=1$ and all observations for which $\hat{P}(Y=0) < \gamma$ to be in the group associated with $Y=0$. We select $0 < \gamma < 1$.

Decision tree terminology



Tree Ensembles

Bagging Decision Trees

- How can bagging be used with decision trees for regression?

1. Construct *B* bootstrapped datasets from available training data
2. For each bootstrapped dataset, construct a deep, unpruned tree
3. For a given test sample, the final output is the average of the *B* trees for that sample

ISSUE: trees are highly correlated

Random Forests

- Extension of bagging that decorrelates the trees
- Follows the same procedure as bagging, except that for each split, only $m < p$ predictors are considered
- As with normal bagging, the number of trees (bootstrapped samples) does not affect variance

Boosting Trees

- Similar to bagging in that multiple trees are created
- No bootstrapping
- Instead, we fit a sequence of small trees to the residuals $r_i = y_i - \hat{y}_i$ (initially all $r_i = y_i$)
- Unlike bagging, boosting variance is sensitive to the value of B. Creating too many trees can lead to overfitting.

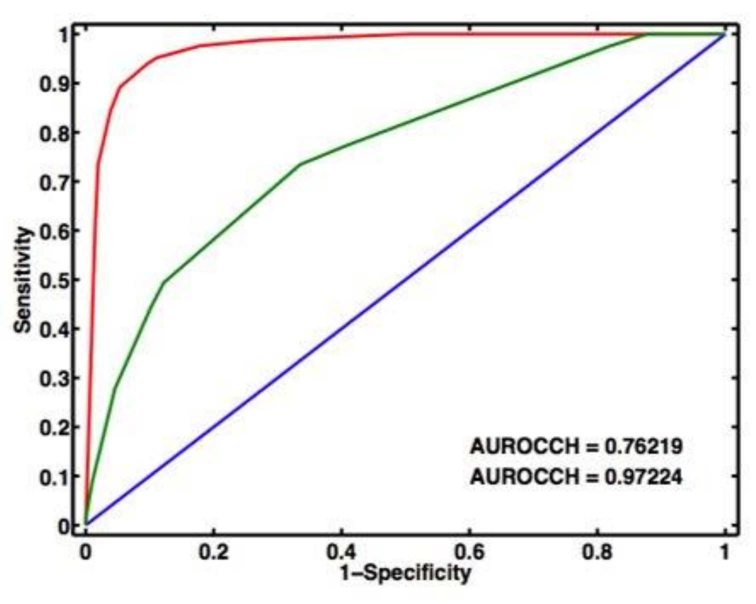
Classification Performance Metrics

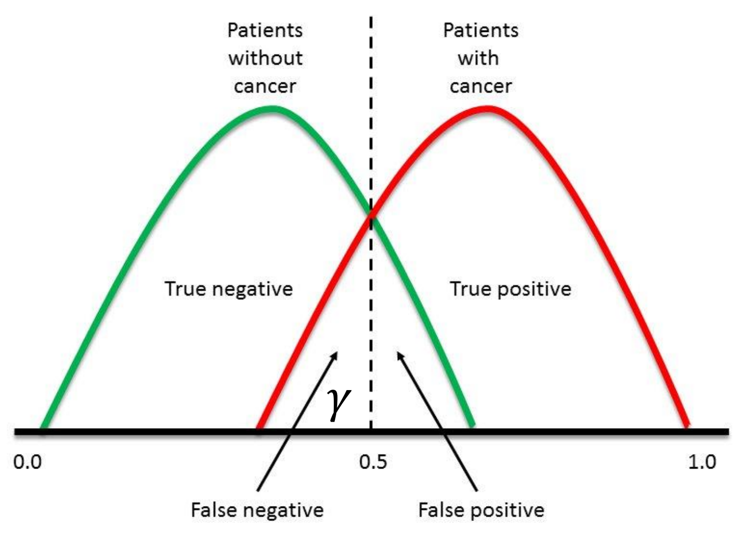
<table>
<tr><td colspan="2"></td><td colspan="2">Actual Class</td></tr>
<tr><td colspan="2"></td><td>Positive</td><td>Negative</td></tr>
<tr><td rowspan="2">Predicted Class</td><td>Positive</td><td>True Positive (TP)</td><td>False Positive (FP)
False Alarm
Type I Error</td></tr>
<tr><td>Negative</td><td>False Negative (FN)
Type II Error</td><td>True Negative (TN)</td></tr>
</table>

- Accuracy = (TP+TN)/(TP+TN+FP+FN)
- Sensitivity (a.k.a Recall, True Positive Rate) = TP/(TP+FN) = TP/P
  - The fraction of actually positive samples predicted positive by the model
- Specificity (a.k.a. True Negative Rate) = TN/(TN+FP) = TN/N
  - The fraction of actually negative samples predicted negative by the model
- Precision (a.k.a. Positive Predictive Value) = TP/(TP+FP)
  - The fraction of samples predicted positive by the model that are actually positive
- False Positive Rate (a.k.a. Probability of false alarm) = FP/(FP+TN) = FP/N
- F1 Score = 2(Precision*Recall)/(Precision + Recall)
- Balanced Accuracy = (Sensitivity + Specificity)/2

Classifier performance with ROC and AUC

- Recall, we can select $0 < \gamma < 1$ as a threshold s.t. $\hat{P}(Y = 1) \geq \gamma$ implies $\hat{y} = 1$. What happens if we change γ?
- Receiver Operating Characteristic Curve (ROC)
  - Illustrates tradeoff between FPR and TPR of a binary classifier as the discrimination threshold, γ, is adjusted
  - Axes
    - 1 – Specificity (false positive rate) = 1 – TN/N
    - Sensitivity (true positive rate) = TP/P
- Area under the curve (AUC)
  - Summary statistic
  - AUC = 0.5 indicates no predictive value (random guessing)
  - AUC = 1.0 indicates perfect predictive value





K-means clustering

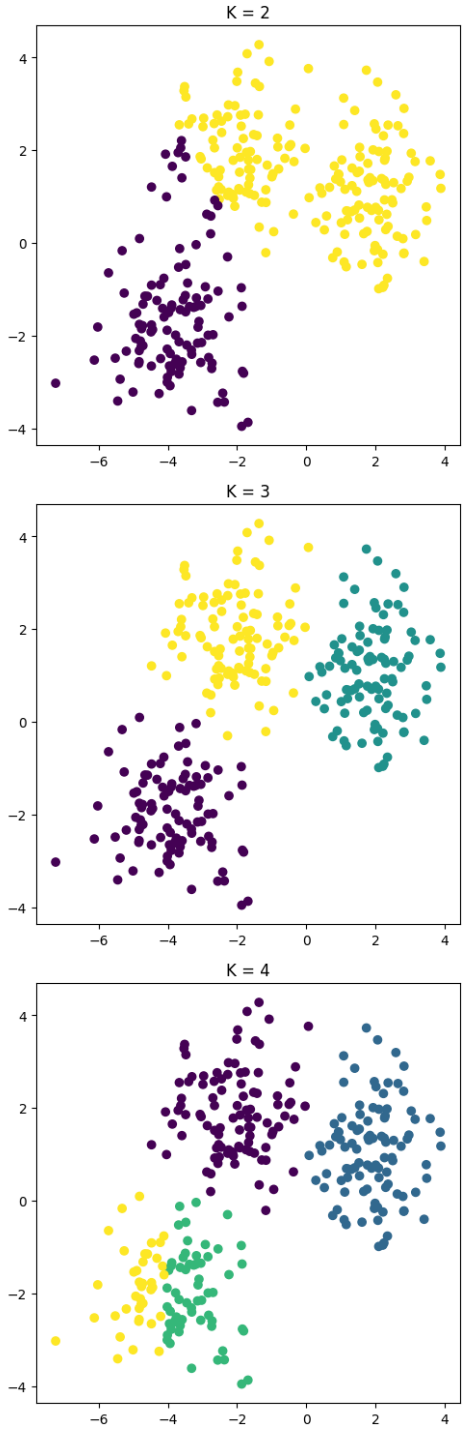
- Given n samples partition the samples into K distinct groups, $C_1, \cdots, C_k$
- Each sample is assigned to exactly one group (cluster)
- Conceptual approach is to assign cluster membership that minimizes

$$\sum_{k=1}^{K} W(C_k)$$

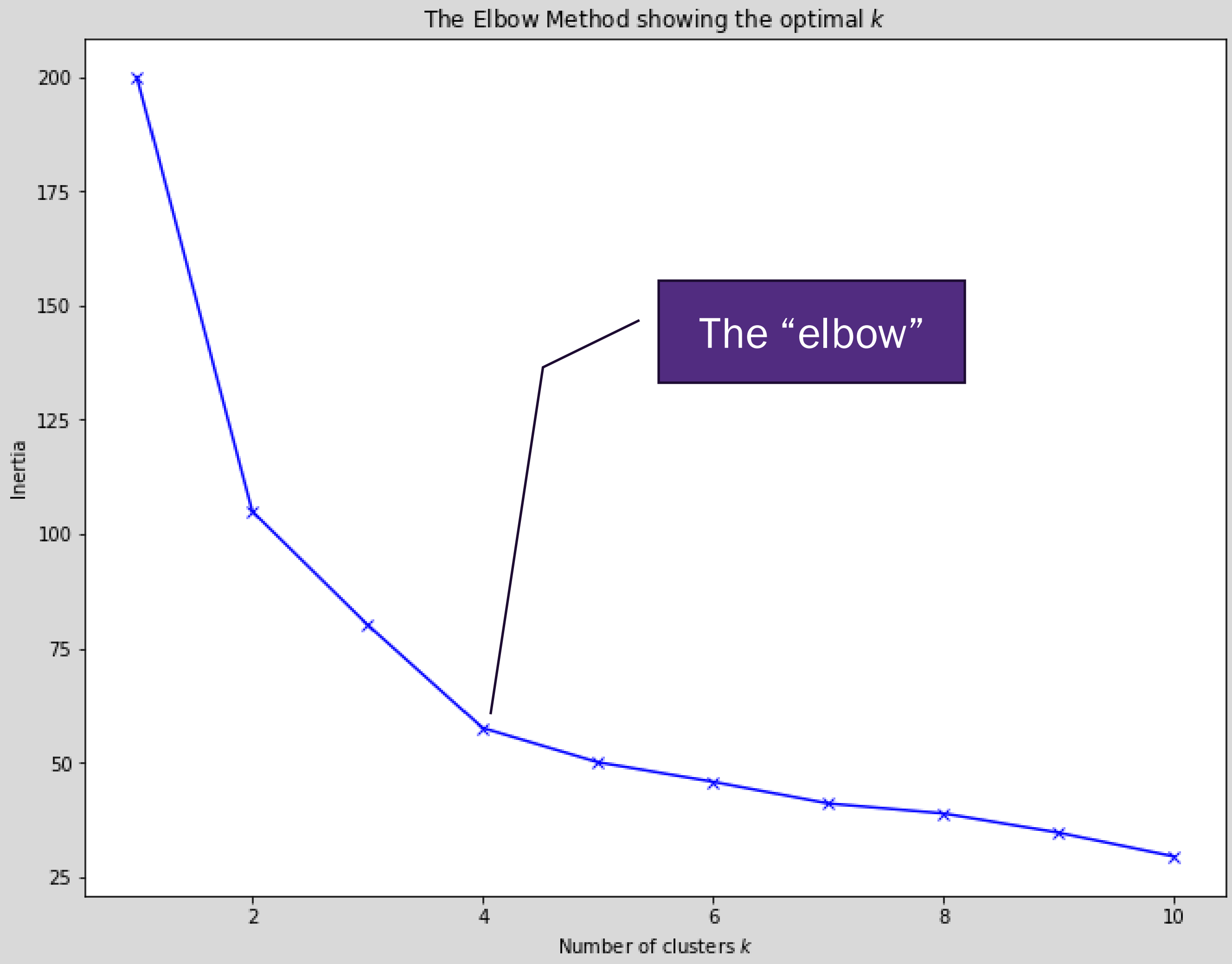
  where $W(C_k)$ is a measure of *intra-cluster variation*
- Letting $W(C_k)$ be the *squared Euclidean distance,* we seek $C_1, \cdots, C_k$ that minimize

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{i,j} - x_{i',j}\right)^2$$

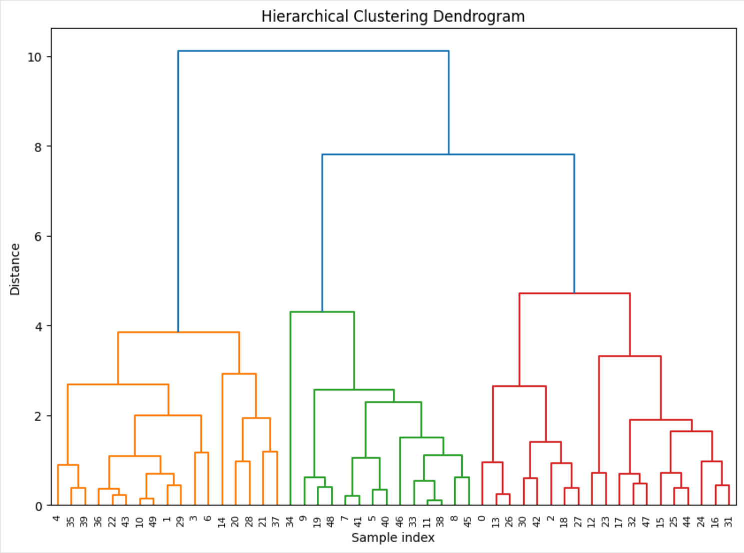
- Other metrics: Hamming (binary vectors), Manhattan (integer vectors), Gower's (combined binary, numerical, categorical)



Elbow Method – How many clusters?



Agglomerative Clustering



The algorithm:

- Each observation starts as its own cluster
- At each step of the algorithm, two **clusters** that are most "similar" are combined into a new larger cluster
- This process of combining clusters is repeated until all observations are members of one single large cluster

Linkage functions

- *Complete (or maximum) Linkage Clustering:* For two clusters, determine the maximum dissimilarity between any observation in the first cluster and any observation in the second cluster.
- *Single Linkage Clustering:* For two clusters, determine the minimum dissimilarity between any observation in the first cluster and any observation in the second cluster.
- *Average Linkage Clustering:* Compute all pairwise dissimilarities between observations in the first and second cluster and calculate the average.

Feed forward neural networks

How many parameters are in this model?

- This layer is fully connected
- Hidden layer of multiple computation nodes

$$z_j^{(1)} = \left[\sum_{j=0}^{n} w_{j,k}^{(1)} x_j\right]$$

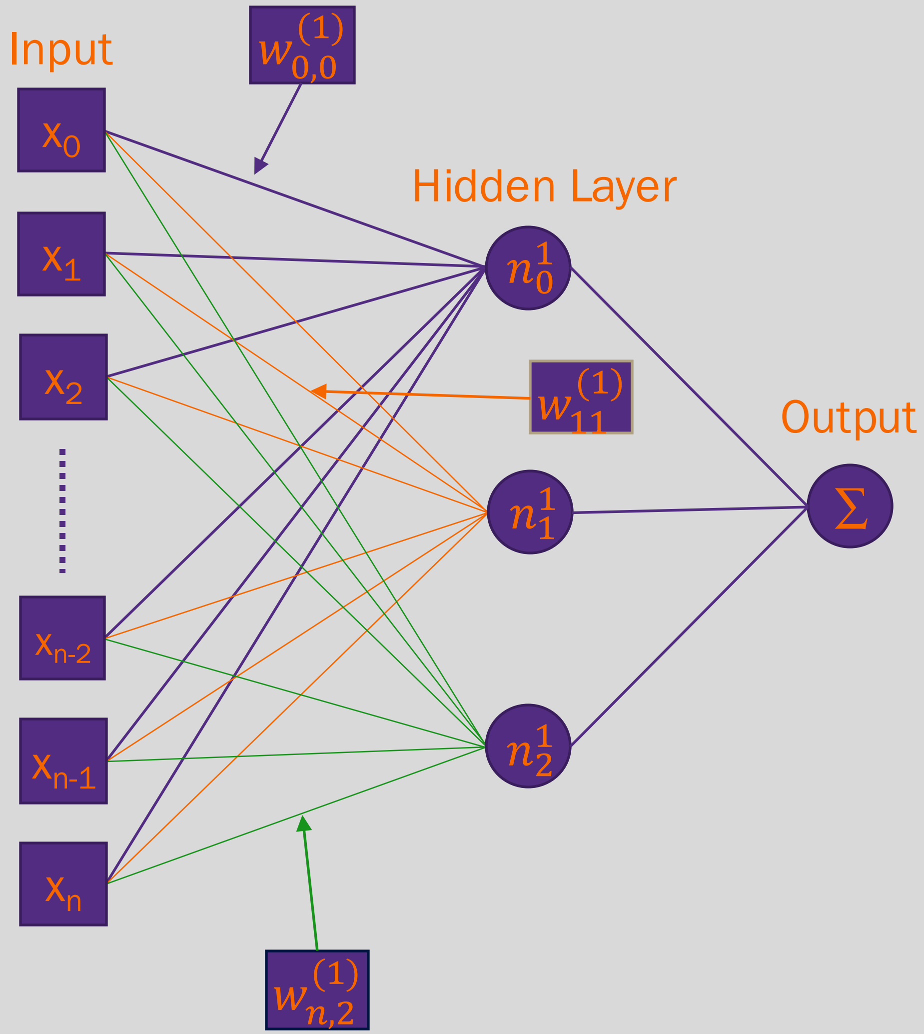
$$a_j^{(1)} = h(z_j^1)$$

- Output layer is a weighted linear sum

$$z^{(out)} = \left[\sum_{j=0}^{k} w_j^{(2)} a_j^{(1)}\right]$$

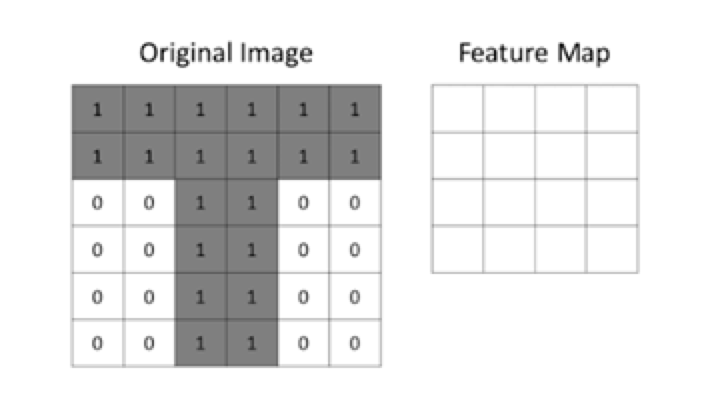
- In classification, sigmoid (binary) or softmax (multiclass) function is applied to output

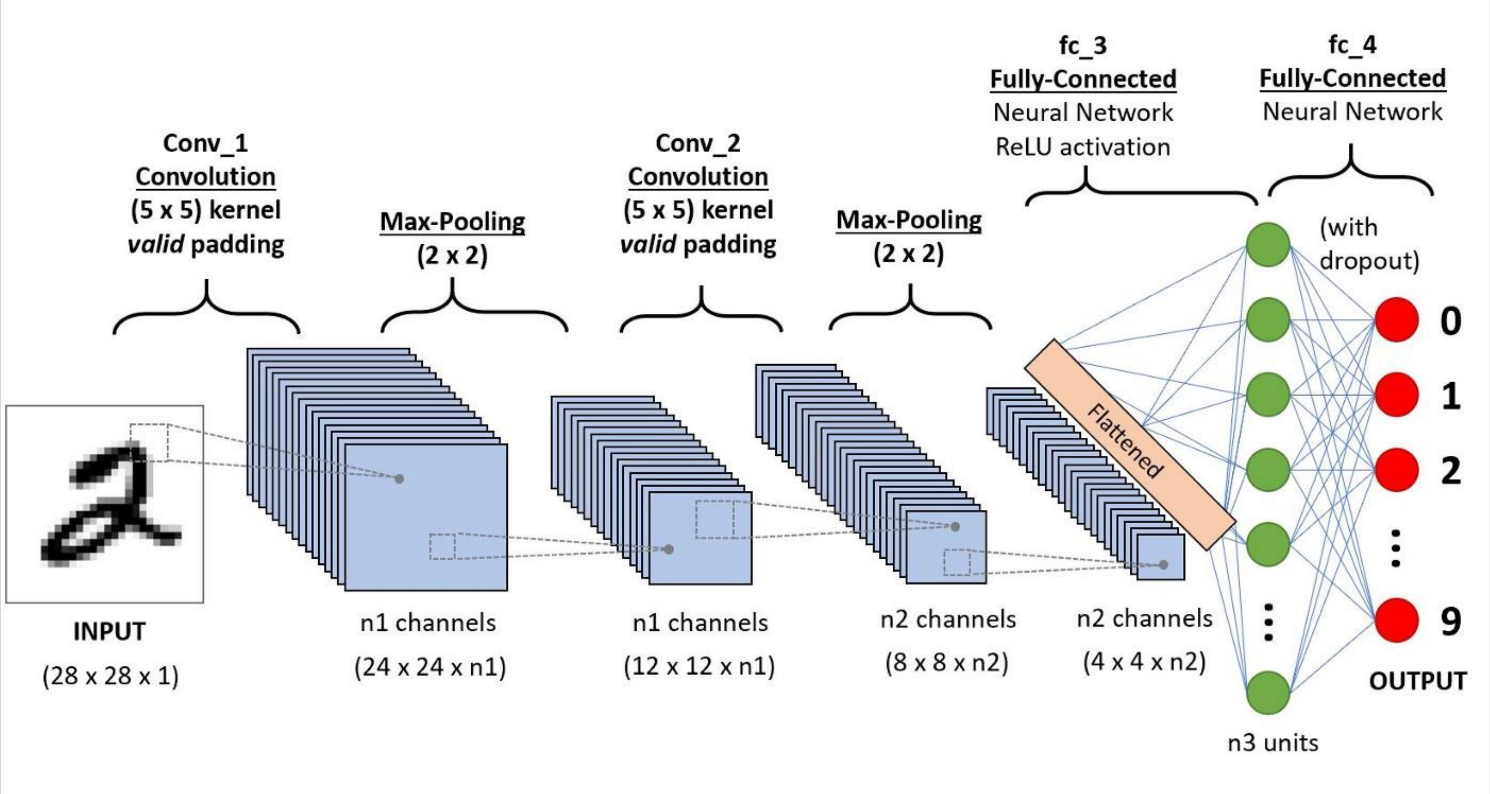
$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad \sigma_{SM}(z_k, \mathbf{z}) = \frac{e^{z_k}}{\sum_{j=1}^{M} e^{z_j}}$$



CPSC 4300 School of COMPUTING Clemson University

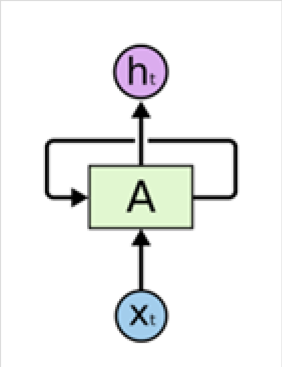
Convolutional Neural Networks



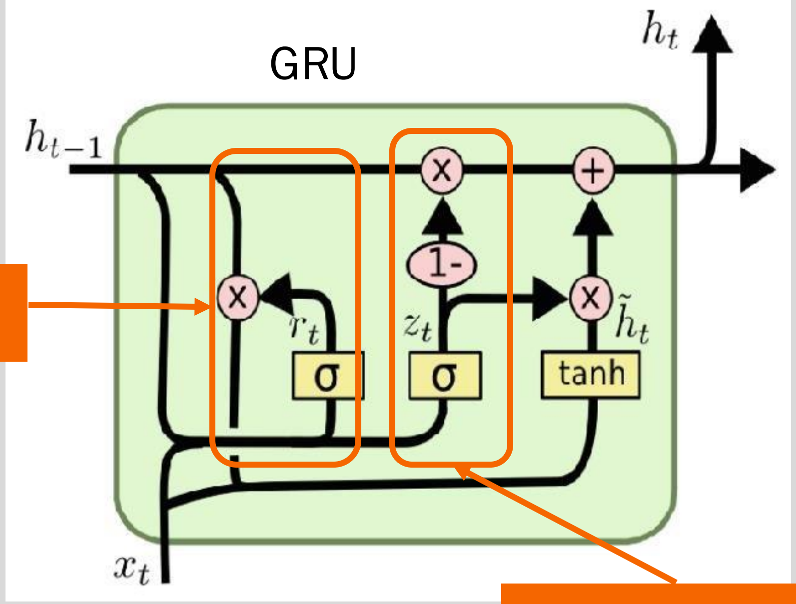


How many inputs are there from the last convolution layer to the fully connected layer?

Recurrent Neural Networks?

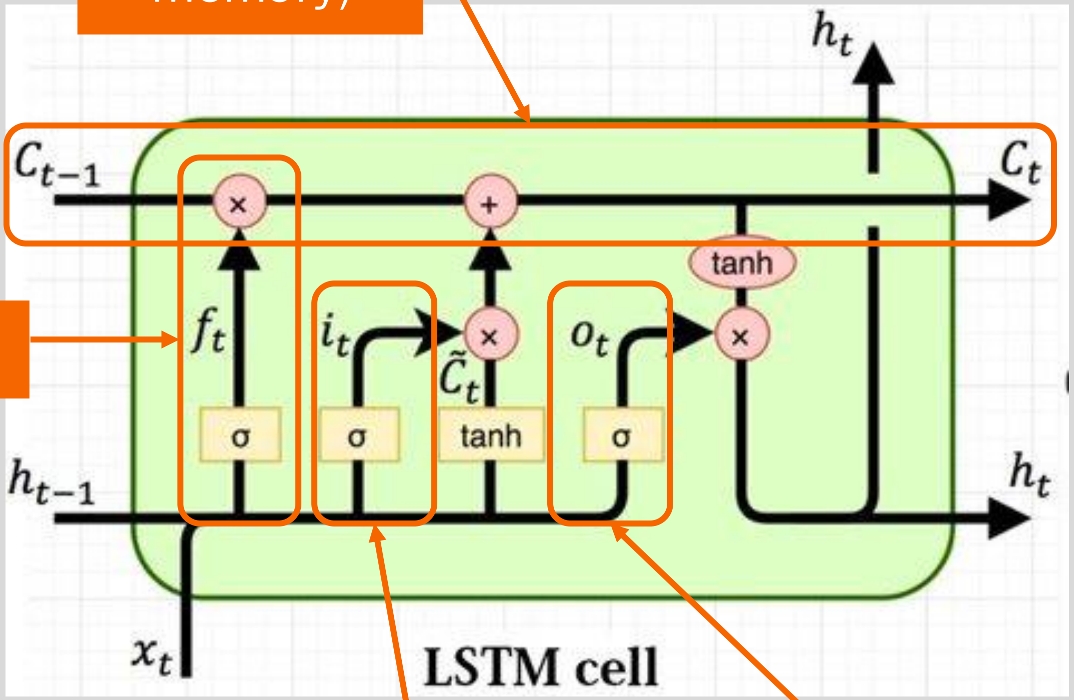


RNN’s have self-loops
What do these gates do?



Reset Gate

Update gate



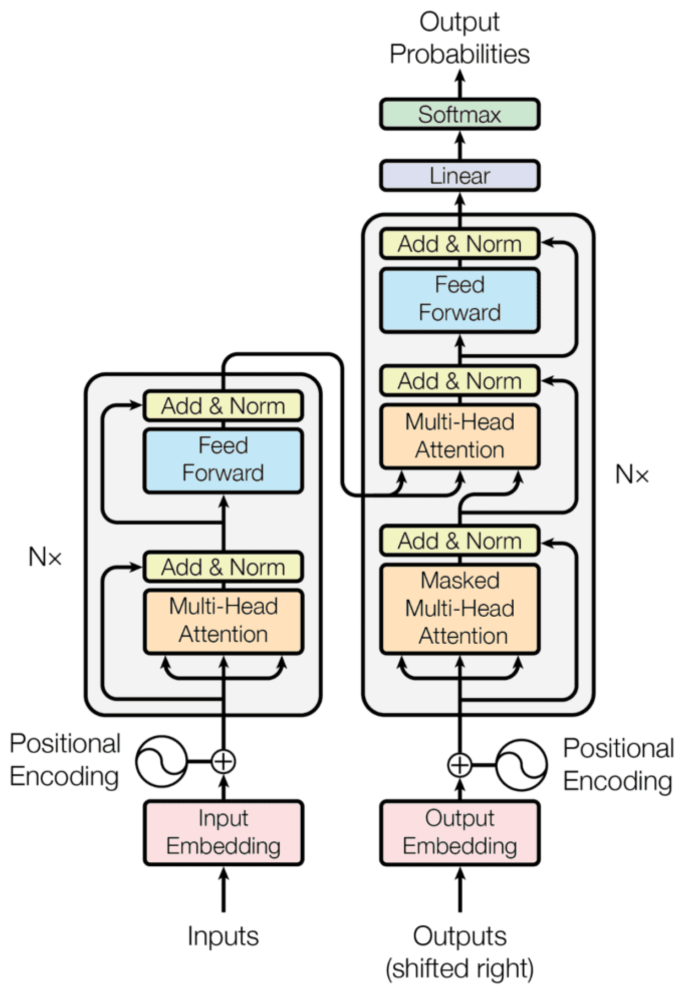
Cell state (long term memory)

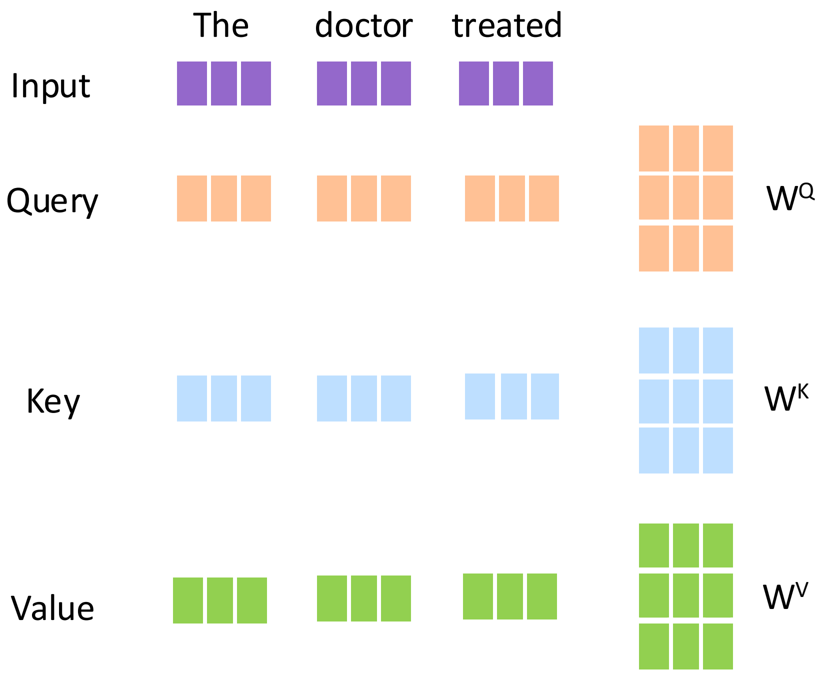
Forget gate

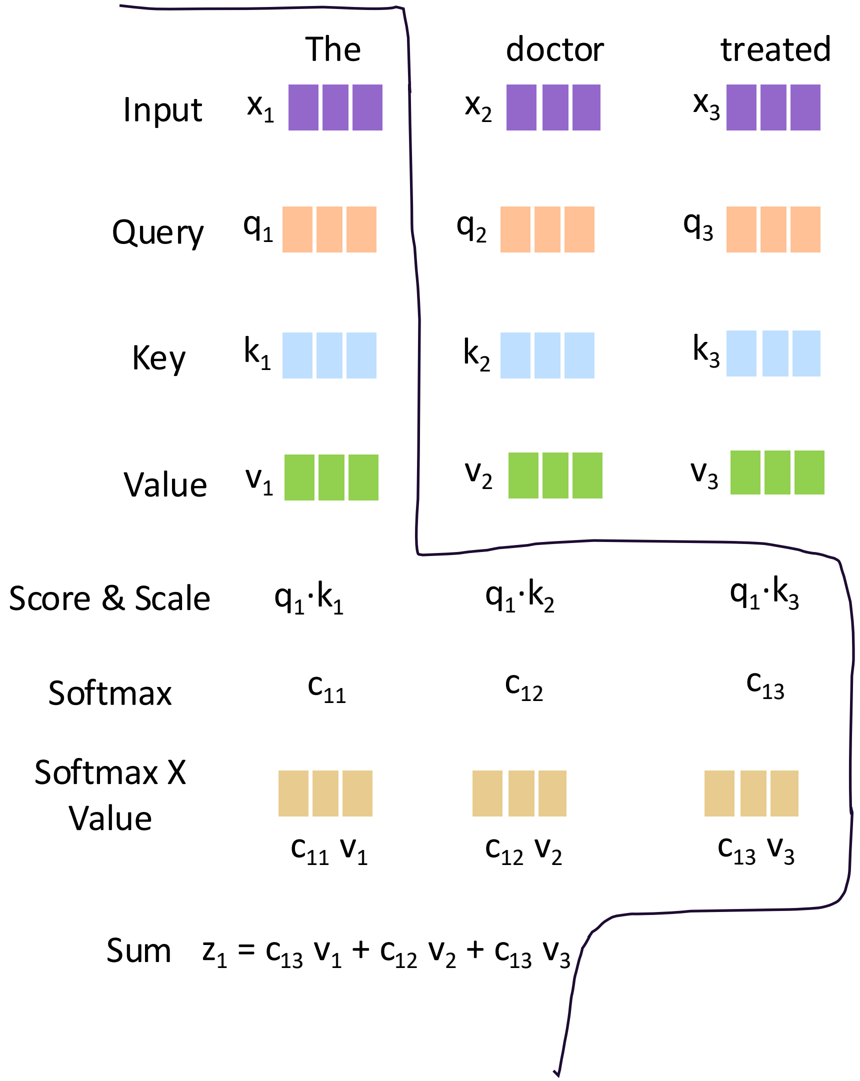
Input gate

Output gate

Transformers and Attention







Deep-learning implementation (PyTorch)

Dimensions

Dimensions

Dimensions

```
class CIFAR10Output(nn.Module):
    def __init__(self, input_dim, num_classes):
        super().__init__()
        self.flatten = nn.Flatten()  # Flatten the input tensor
        self.linear = nn.Linear(input_dim, num_classes)  # Linear layer with 10 output units

    def forward(self, x):
        x = self.flatten(x)  # Flatten the input tensor
        x = self.linear(x)   # Pass through the linear layer
        return x

# Test the module with random input data
decoder = CIFAR10Output(num_kernels*15*15, 10)
input_tensor = torch.randn(5, num_kernels, 15, 15)  # Batch size of 5, input tensor shape [5, 32, 7, 7]
rslt = decoder(input_tensor)
print("Rslt shape:", rslt.shape)  # Expected output shape: [5, 10]
print(rslt.dtype)
```