IAI, TCG CREST

Avisek Gupta (avisek003@gmail.com)

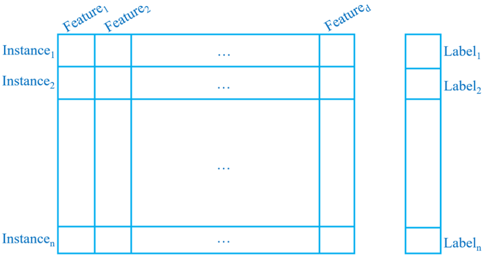
Machine Learning

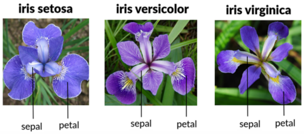
20 – RNNs, Imbalanced Classification

November 25, 2022

Data Matrix

Data Matrix view of a data set: The data instances $x_i \in \mathbb{R}^d$ are present in the rows of the data matrix $X = [x_1, ..., x_n]^T$, the features are present along the columns.





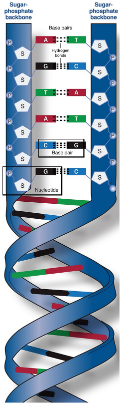
	Petal Length	Petal Width	Sepal Length	Sepal Width		
Iris Instance$_1$	5.1	3.5	1.4	0.2	0	Iris Species$_1$
Iris Instance$_2$	4.9	3.0	1.4	0.2	0	Iris Species$_2$
Iris Instance$_3$	4.7	3.2	1.3	0.2	0	Iris Species$_3$
	...				...	

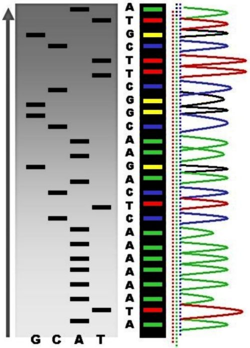
Sequential Data

For **Sequential Data**, there may be dependencies between a data instance x_i and previous instances $x_{i-1}, x_{i-2}, \ldots, x_{i-k}$, which we should also try to model.

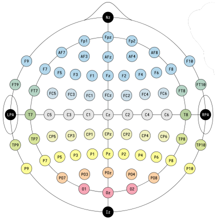
Some examples:

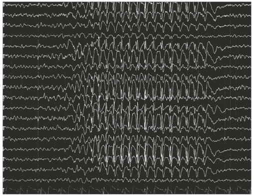
1. Modeling DNA sequences





2. Modeling brain EEG signals





Sequential Data

3. Modeling audio and natural language

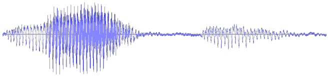
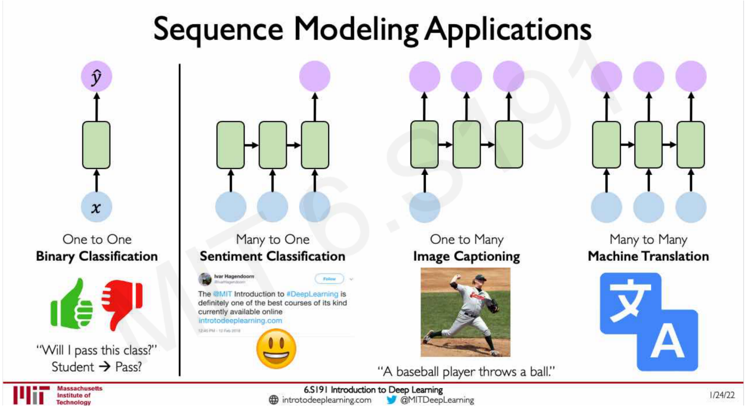


Figure: A speech signal



Sequential Data

Modeling natural language

The boy ate a ______

- pizza
- cake
- apple
- banana
- car
- laptop
- biryani
- desk
- ...

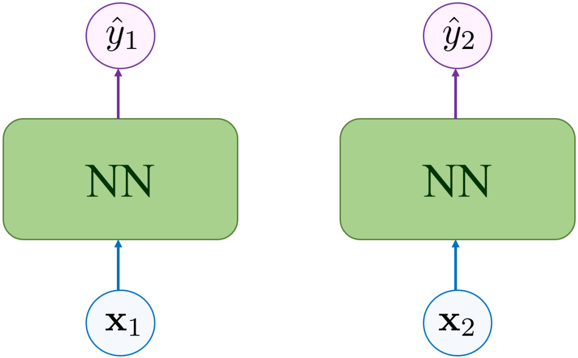
Modeling Sequential Data

The problems with modeling with conventional machine learning models (like MLPs):

- A model is defined to work on an input of fixed size $\mathbf{x}_i \in \mathbb{R}^d$.
- If a sequence is broken down into d-sized sub-sequences, conventional models still do not consider the dependencies between sequence instances.

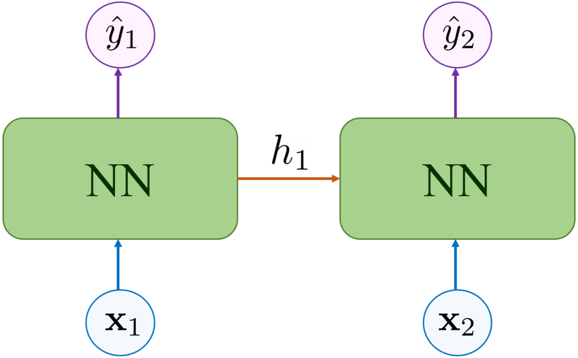
Modeling Sequential Data

If a sequence is broken down into d-sized sub-sequences, conventional models still do not consider the dependencies between sequence instances.



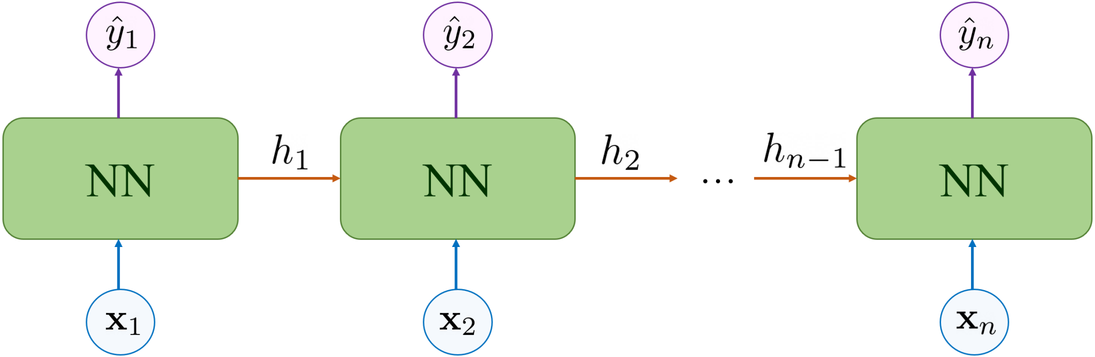
Modeling Sequential Data

Consider h_i which carries previous information from x_i.



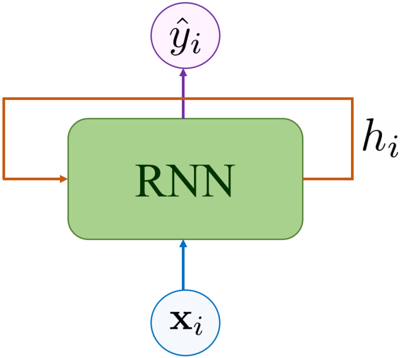
Modeling Sequential Data

Consider h_i which carries previous information from x_i. A sequence of n terms can then be modeled with the help of information carried over by h_i.



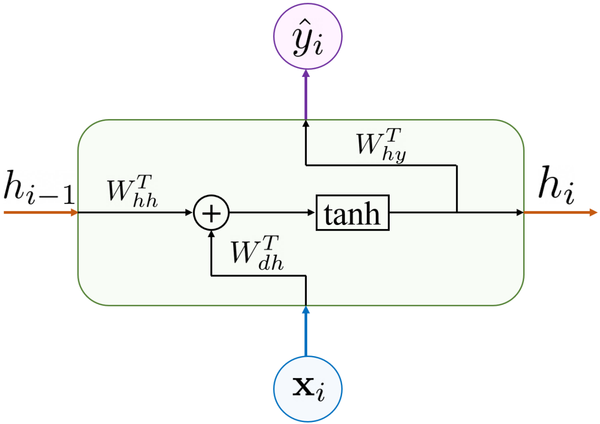
Recurrent Neural Network

A Recurrent Neural Network (RNN) cell takes as input $\mathbf{x}_i \in \mathbb{R}^d$, and produces two outputs: a cell state h_i which takes into account past information, and the output of the cell $\hat{y}_i$.



Recurrent Neural Network

A Recurrent Neural Network (RNN) cell takes as input $\mathbf{x}_i \in \mathbb{R}^d$, and produces two outputs: a cell state h_i which takes into account past information, and the output of the cell $\hat{y}_i$.

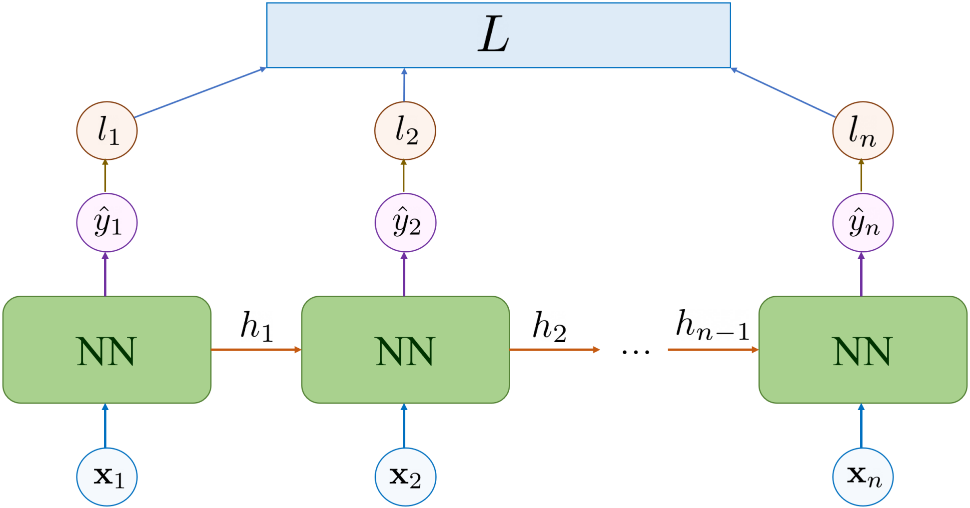


$$h_i = \tanh(W_{hh}^T h_{i-1} + W_{dh}^T x_i)$$

$$\hat{y}_i = W_{hy}^T h_i$$

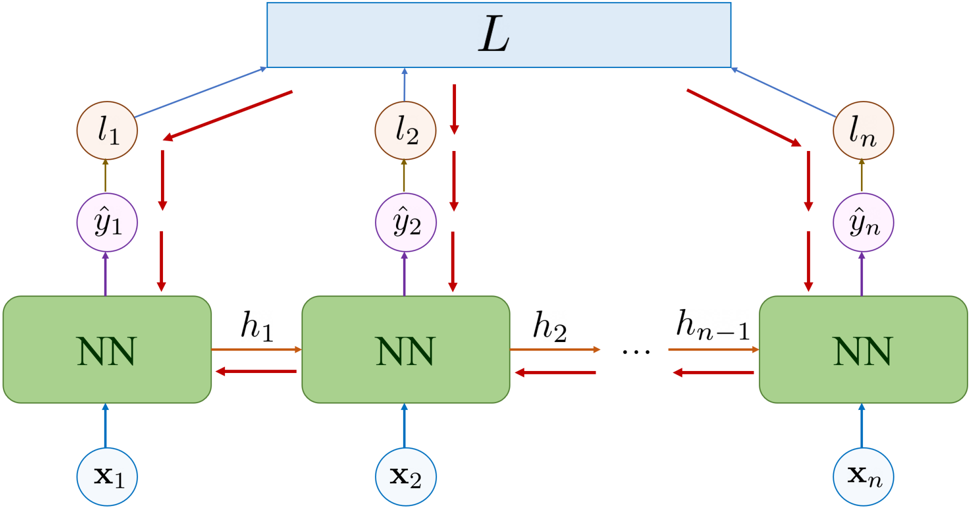
RNN Forward Propagation

From the predicted $\hat{y}_i$, losses can be calculated l_i, all of which combined form a total loss L.

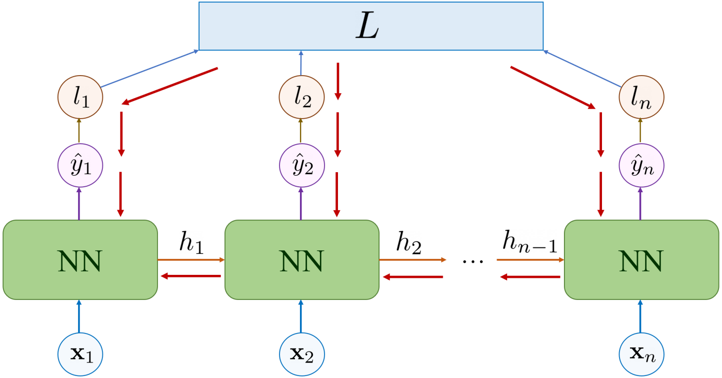


RNN Backpropagation Through Time

Through backpropagation, the network parameters W_{dh}, W_{hh}, W_{hy} can be updated.



RNN Backpropagation Through Time Challenges

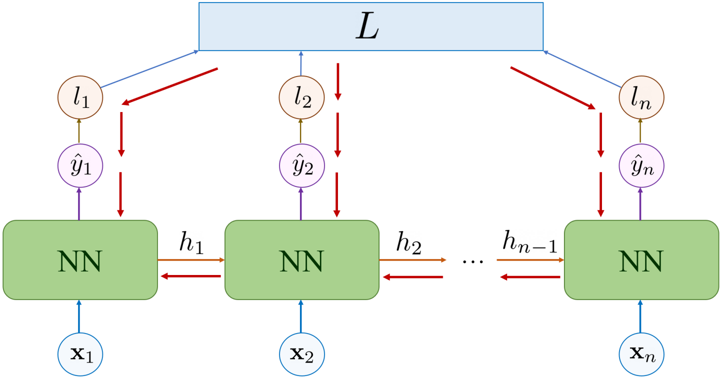


Updating W_{hy} can be done with ease.

$$\hat{y}_i = W_{hy}^T h_i,\ l_i = f(\hat{y}_i),\ L = \sum_i f(l_i)$$

The computation of $\nabla_{W_{hy}} L$ involves the sum of $\nabla_{W_{hy}} \hat{y}_i$.

RNN Backpropagation Through Time Challenges

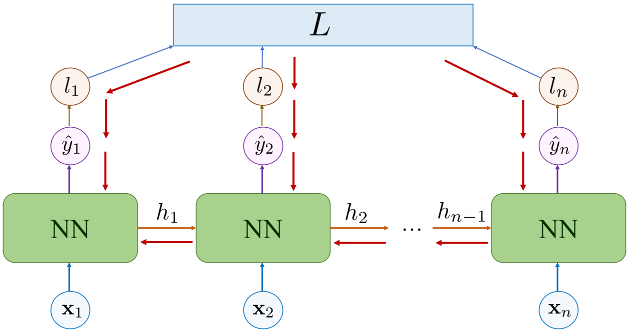


Updating W_{hh} can be difficult. Ignoring the tanh activation,

$$h_n = W_{hh}^T h_{n-1} + W_{dh}^T x_n = (W_{hh}^T \{W_{hh}^T h_{n-2} + W_{dh}^T x_{n-1}\} + W_{dh}^T x_n)$$

$$= (W_{hh}^T)^2 h_{n-2} + W_{hh}^T W_{dh}^T x_{n-1} + W_{dh}^T x_n = (W_{hh}^T)^n h_0 + \ldots$$

RNN Backpropagation Through Time Challenges

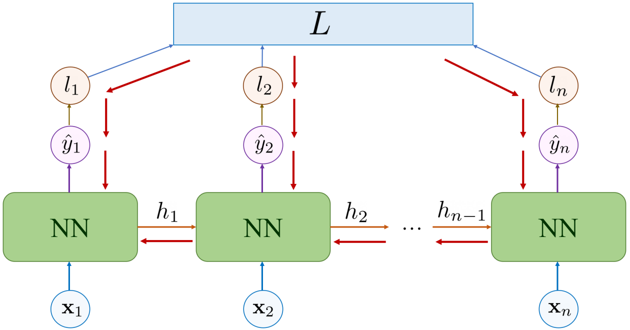


The terms $(W_{hh}^T)^n$ with large n can cause two kinds of problems:

1. **Exploding Gradients**: If W_{hh} has several values > 1, then $(W_{hh}^T)^n$ will have extremely large values.

A solution: Use **Gradient Clipping** to limit the magnitude of the gradients.

RNN Backpropagation Through Time Challenges

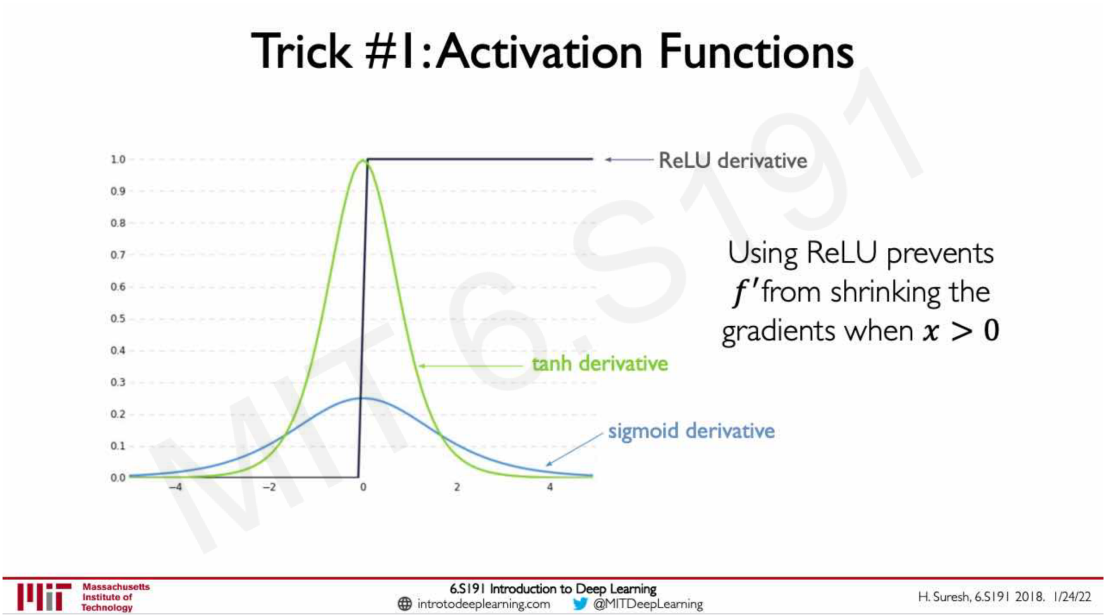


The terms $(W_{hh}^T)^n$ with large n can cause two kinds of problems:

2. **Vanishing Gradients**: If W_{hh} has several values < 1, then gradients that involve computing $(W_{hh}^T)^n$ will become zero.

Solutions: Find suitable (i) Activation Functions (ii) Weight initializations (iii) Network Architectures.

Avoiding Vanishing Gradients



Avoiding Vanishing Gradients

Trick #2: Parameter Initialization

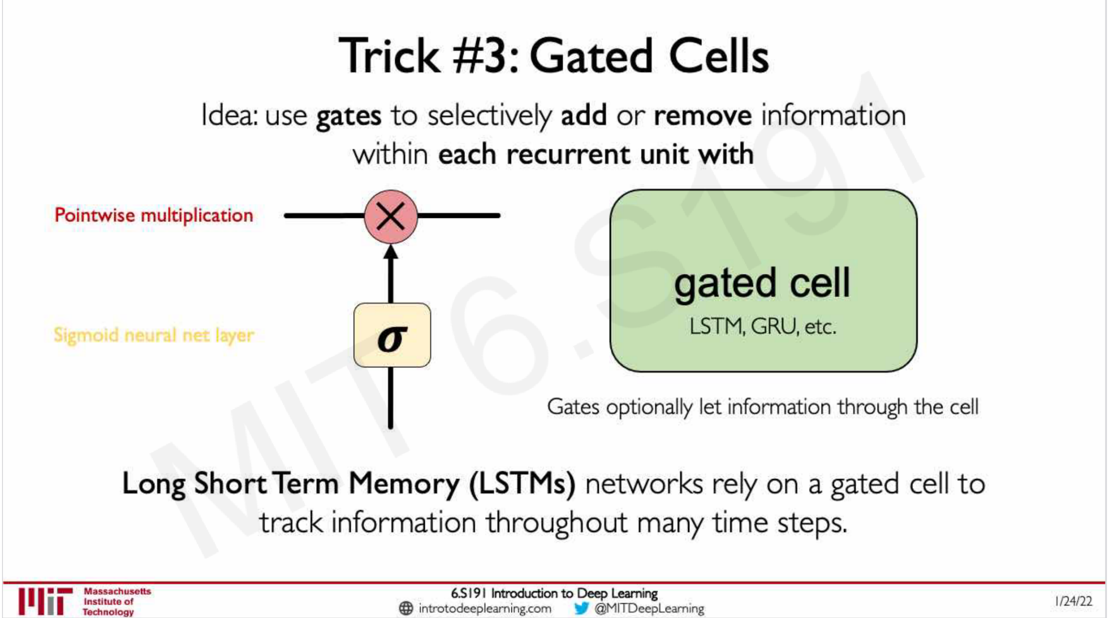
Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

Avoiding Vanishing Gradients



Networks for Sequence Modeling

Subsequent networks for sequence modeling:

- Long Short Term Memory (LSTM) networks
- Gated Reccurent Unit (GRU) networks
- Transformers

Imbalanced Classification

Imbalanced Classification

Let us consider a contingency table for a binary classification problem.

Size of class 1 is 10000.

Size of class 2 is 100.

	R_1	R_2
D_1	9990	10
D_2	90	10

It may be beneficial for the classifier to consider misclassifying the minority class as more severe than misclassifying the majority class.

Cost-Sensitive Learning

Cost-Sensitive Learning: In the cost function of a classifier, weigh the cost of misclassification of each class j by a weight $w_j = \frac{n}{2n_j}$, where n is the total number of instances, and n_j is the number of instances of class j.

Example: For Logistic Regression:

Loss of Logistic Regression:

$$-\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)$$

The first term is the cost for the minority class ($y_i = 1$), the second term is the cost for the majority class ($y_i = 0$).

$$-\frac{1}{n}\sum_{i=1}^{n} w_1 y_i \ln(\hat{y}_i) + w_0(1-y_i)\ln(1-\hat{y}_i)$$

Cost-Sensitive Learning

Cost-Sensitive Learning: In the cost function of a classifier, weigh the cost of misclassification of each class j by a weight $w_j = \frac{n}{2n_j}$, where n is the total number of instances, and n_j is the number of instances of class j.

For a balanced class $n_0 = n_1 = \frac{n}{2}$,

$$w_1 = \frac{n}{2\frac{n}{2}} = 1, \quad w_0 = \frac{n}{2\frac{n}{2}} = 1$$

Hence $w_1 = w_0$.

For an imbalanced class $n_1 = \frac{n}{10}, n_0 = \frac{9n}{10}$,

$$w_1 = \frac{n}{2\frac{n}{10}} = 5, \quad w_0 = \frac{n}{2\frac{9n}{10}} = \frac{10}{18} < 1$$

Hence $w_1 > w_0$.

Synthetic Minority Oversampling TEchnique (SMOTE)

Idea: In order to reduce the difference in the sizes of the majority class and the minority class, generate more synthetic minority class data instances.

Synthetic Minority Oversampling TEchnique (SMOTE)

Idea: In order to reduce the difference in the sizes of the majority class and the minority class, generate more synthetic minority class data instances.

SMOTE algorithm:

1. Draw a random instance $\mathbf{x}_i$ from the minority class.
2. Identify the k nearest neighbors of this instance $\mathbf{x}_i$. Randomly select one of these k nearest neighbors (say $\mathbf{x}_j$)
3. Obtain as a new instance, an instance on the vector joining $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. the new instance $\mathbf{x}_k^s$ is,

$$\mathbf{x}_k^s = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j, \quad \lambda \in (0, 1).$$

Synthetic Minority Oversampling TEchnique (SMOTE)

SMOTE algorithm:

1. Draw a random instance $\mathbf{x}_i$ from the minority class.
2. Identify the k nearest neighbors of this instance $\mathbf{x}_i$. Randomly select one of these k nearest neighbors (say $\mathbf{x}_j$)
3. Obtain as a new instance, an instance on the vector joining $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. the new instance $\mathbf{x}_k^s$ is,

$$\mathbf{x}_k^s = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \quad \lambda \in (0,1).$$

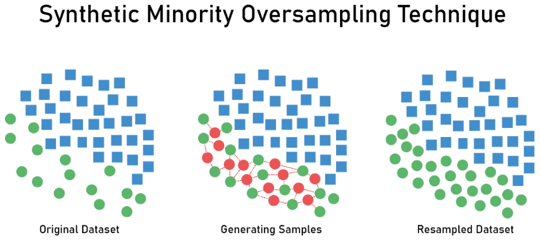


Image Source: `https://emilia-orellana44.medium.com/smote-2acd5dd09948`

References

[1] Introduction to Deep Learning, MIT 6.S191.
http://introtodeeplearning.com/

[2] MIT 6.S191: Recurrent Neural Networks and Transformers.
https://www.youtube.com/watch?v=QvkQ1B3FBqA

[3] Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique". Journal of artificial intelligence research 16 (2002): 321-357.

Image Sources:

https://en.wikipedia.org/wiki/DNA#/media/File:Phosphate_backbone.jpg

https://en.wikipedia.org/wiki/DNA_sequencing#/media/File:Radioactive_Fluorescent_Seq.jpg

https://commons.wikimedia.org/wiki/File:EEG_Absence_seizure.png

https://commons.wikimedia.org/wiki/File:EEG_10-10_system_with_additional_information.svg

https://commons.wikimedia.org/wiki/File:Signal-speech-martin-de.png