Machine Learning Basics for Social Scientists

Yongjun Zhang, Ph.D.

Department of Sociology
Institute for Advanced Computational Science
Stony Brook University
Research Affiliate at New York University

July 14, 2022

Table of Contents

Artificial Intellegence, Machine Learning, and Deep Learning

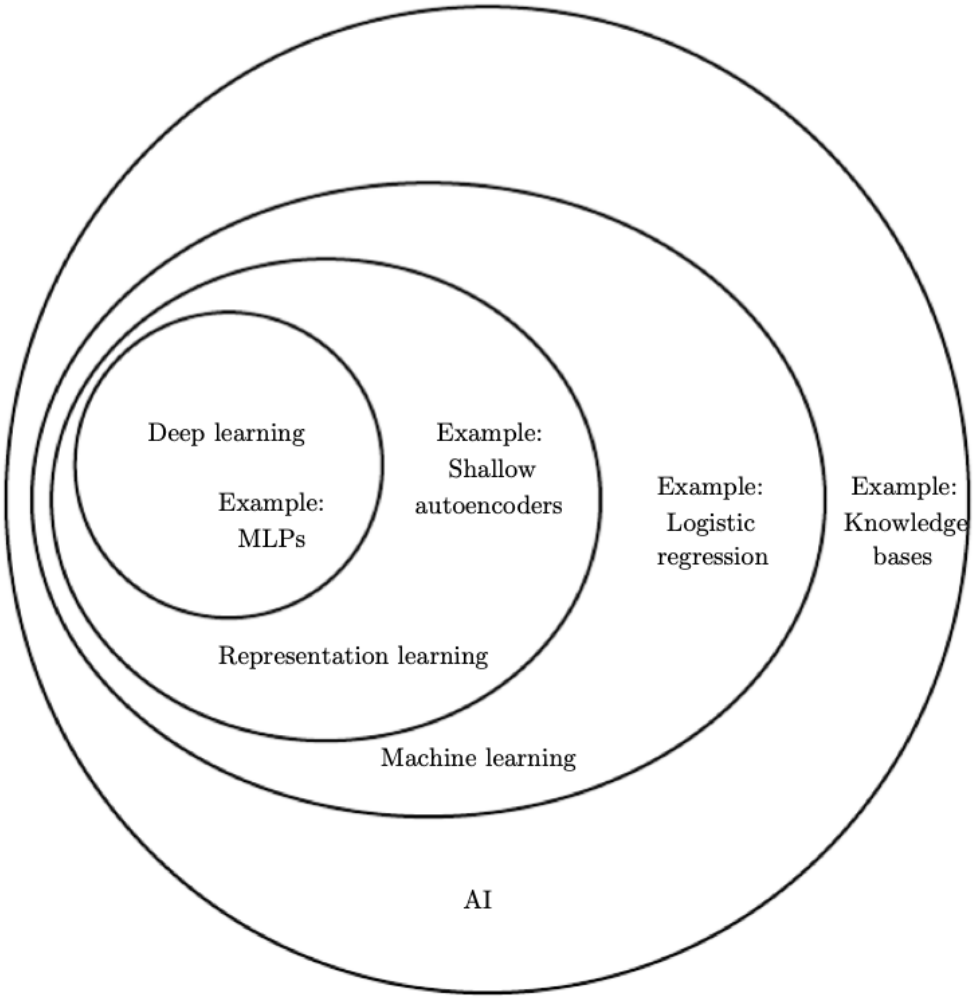


Figure: Goodfellow 2016. The AI Paradigm Shifting

The Difference between Traditonal AI and ML AI

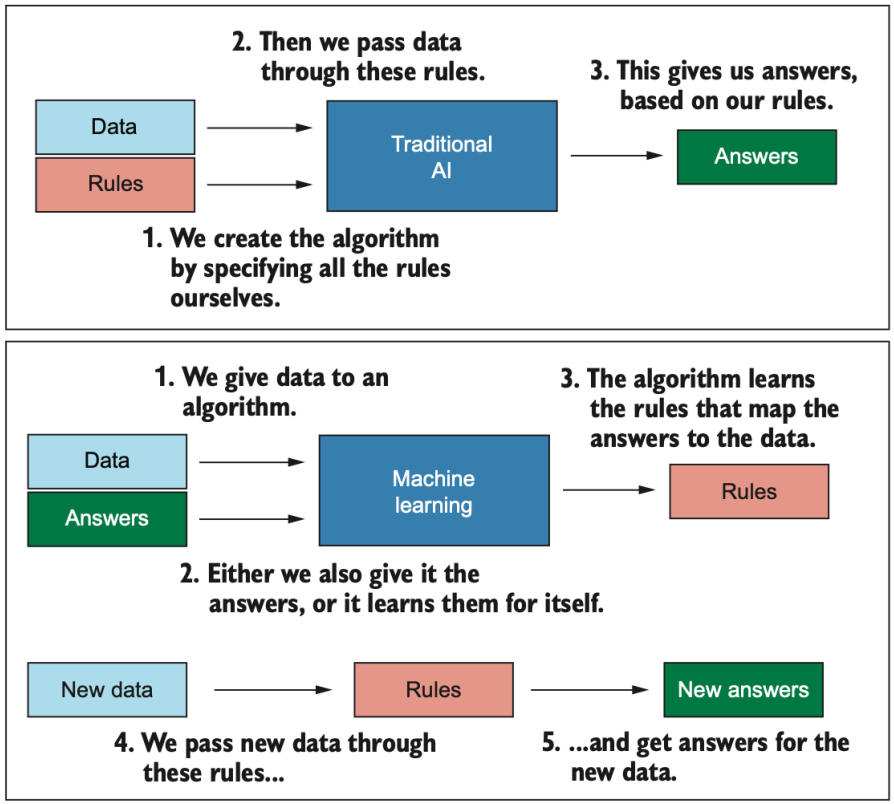


Figure: Rhys 2020. Traditional AI vs. ML AI.

What is Machine Learning?

- Arthur Samuel, a scientist at IBM, coined the term of Machine Learning in 1959 when training an algorithm to play checkers.
- Machine learning is a field at the intersection of statistics and computer science that uses algorithms to extract information and knowledge from data (Monila and Garip 2019).
- Machine learning is a class of flexible algorithmic and statistical techniques for prediction and dimension reduction (Grimmer, Roberts, and Stewart, 2021).

ML discovers rules for executing a data processing task, given examples of what's expected (Chott 2020).

A machine learning model transforms its input data into meaningful outputs, a process that is learned from exposure to known examples of inputs and outputs.

- Input data points: $\mathcal{X} = \{x_1, ..., x_n, ..., x_N\}$
- Examples of the expected output: $\mathcal{Y} = \{y_1, ..., y_n, ..., y_N\}$
- Finding an algorithm that minimizes the distance between the predictions $\hat{\mathcal{Y}} = f(\mathcal{X}, \hat{\theta})$ and true values $\mathcal{Y}$
- A way to measure whether the algorithm is doing a good job. The distance between $\hat{\mathcal{Y}} = f(\mathcal{X}, \hat{\theta})$ and $\mathcal{Y}$ is measured by a loss function, $\mathcal{L}(\mathcal{Y}, f(\mathcal{X}, \hat{\theta}))$

Learning is essentially a two-step processes (supervised ML):

- Learning the representations of data
- Learning mappings from these representations of the data to the output.

Conventional Machine Learning focuses on the second step.

Raw input text examples (e.g., yelp review):

- I don't like this restaurant (Negative)
- This restaurant is fantastic (Positive)

Feature engineering (document-feature matrix):

$$\mathcal{X} = \begin{pmatrix} I & fantastic & \dots & restaurant \\ 1 & 0 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Find the function $\mathcal{F}$ that maps representation to output: $\mathcal{X} \to \mathcal{Y}$

This is similar to quantitative social science?

Think about you are running an OLS regression in stata...

- $Y_i|\boldsymbol{X}_i \sim \mathcal{N}(\alpha + \boldsymbol{\beta}^\top \boldsymbol{X}_i, \sigma^2)$
- We can estimate our parameters $\theta = (\alpha, \boldsymbol{\beta})$ by ordinary least squares: $(\hat{\alpha}, \hat{\boldsymbol{\beta}}) = arg\ min_{\alpha,\beta} \sum_{i=1}^{N} (Y_i - \alpha - \boldsymbol{\beta}^\top \boldsymbol{X_i})^2$
- For social scientists, we are interested in the coefficients... $Income_i = \alpha + \beta_1 edu_i + \beta_2 SES_i + \epsilon$
- But in ML setting, we are predicting $\hat{Y}_{N+1}$

The Difference between ML and Conventional Social Science

- The focus of $\hat{y}$ vs. the focus of $\hat{\beta}$
- Out-of-sample prediction vs. Explanation
- terminology: features vs. independent vars; outcomes/target vs dept var
- Any others?

Different Types of Machine Learning Algorithms

ML algorithms can be broadly categorized into:

- Supervised ML (Classification, Regression)
- Unsupervised ML (Dimension Reduction, Clustering)

The term supervised learning originates from the view of the target y being provided by an instructor or teacher who shows the machine learning system what to do. In unsupervised learning, there is no instructor or teacher, and the algorithm must learn to make sense of the data without this guide. We do have other variants like semi-supervised ML.

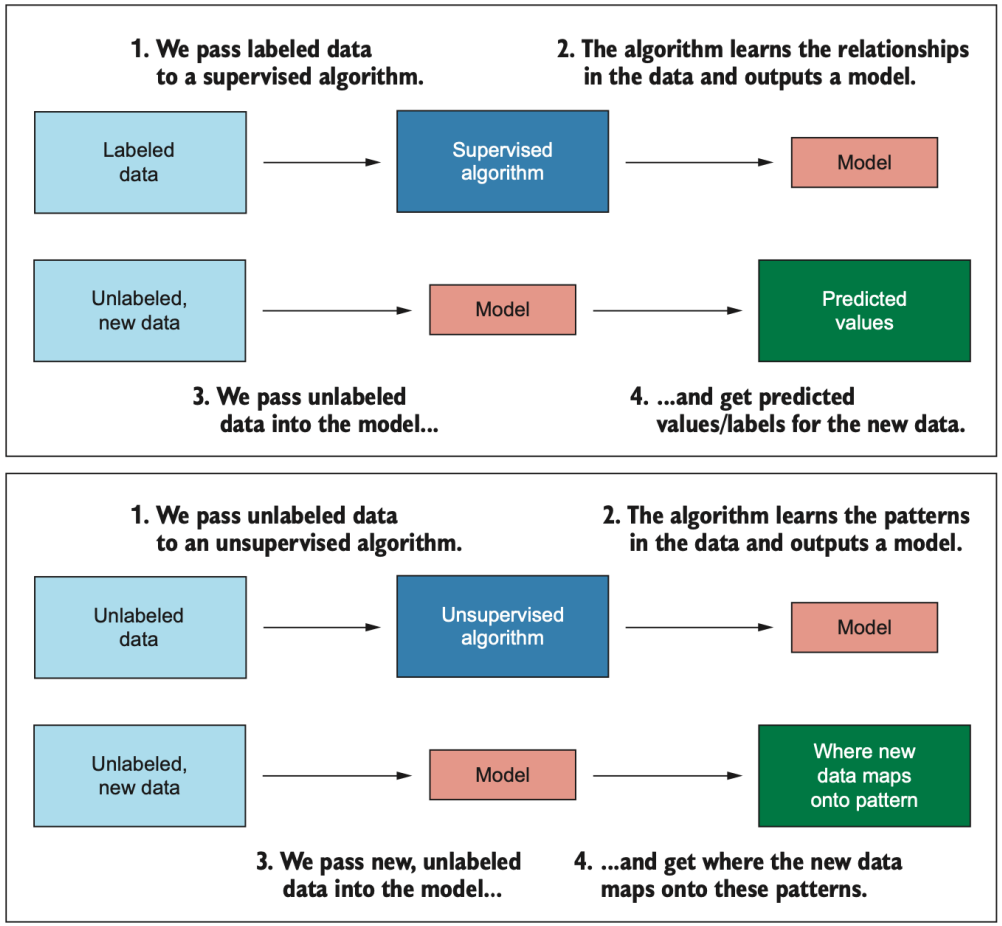


Figure: Rhys 2020. Supervised ML vs. Unsupervised ML

Supervised Machine Learning

- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target.
- Supervised ML Algorithms: logistic regression, support vector machine, regression Trees, Random Forests, causal trees, etc.

How to Train a Supervised ML Algorithm based on Training Data: The Practical Guide

Let us predict outcome of interest using a set of features.

- We have collected a dataset: $(\mathcal{X}, \mathcal{Y}) = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n), ..., (\boldsymbol{x}_N, y_N)\}$. Note that a dataset is a collection of many examples and we call them data points. We call x as a feature and y as a label or target.
- We want to learn a function that maps x and y: $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta})$; Note that the activation function varies based on our outcomes.
- We define a cost function (a.k.a, loss function) to measure the performance of the prediction: $\mathcal{J}(\boldsymbol{\theta}) = \sum \mathcal{L}(y - f(\boldsymbol{x}, \boldsymbol{\theta}))$
- We find the set of parameters $\boldsymbol{\theta}$ to minimize the $\mathcal{J}(\boldsymbol{\theta})$

The goal is to find the set of parameters $\boldsymbol{\theta}$ to minimize the $\mathcal{J}(\boldsymbol{\theta})$.

Here is a list of functions we used a lot:`https://keras.io/api/losses/`

- Regression Loss Functions
  1. Mean Squared Error Loss
  2. Mean Squared Logarithmic Error Loss
  3. Mean Absolute Error Loss
- Binary Classification Loss Functions
  1. Binary Cross-Entropy
  2. Hinge Loss
  3. Squared Hinge Loss
- Multi-Class Classification Loss Functions
  1. Multi-Class Cross-Entropy Loss
  2. Sparse Multiclass Cross-Entropy Loss
  3. Kullback Leibler Divergence Loss

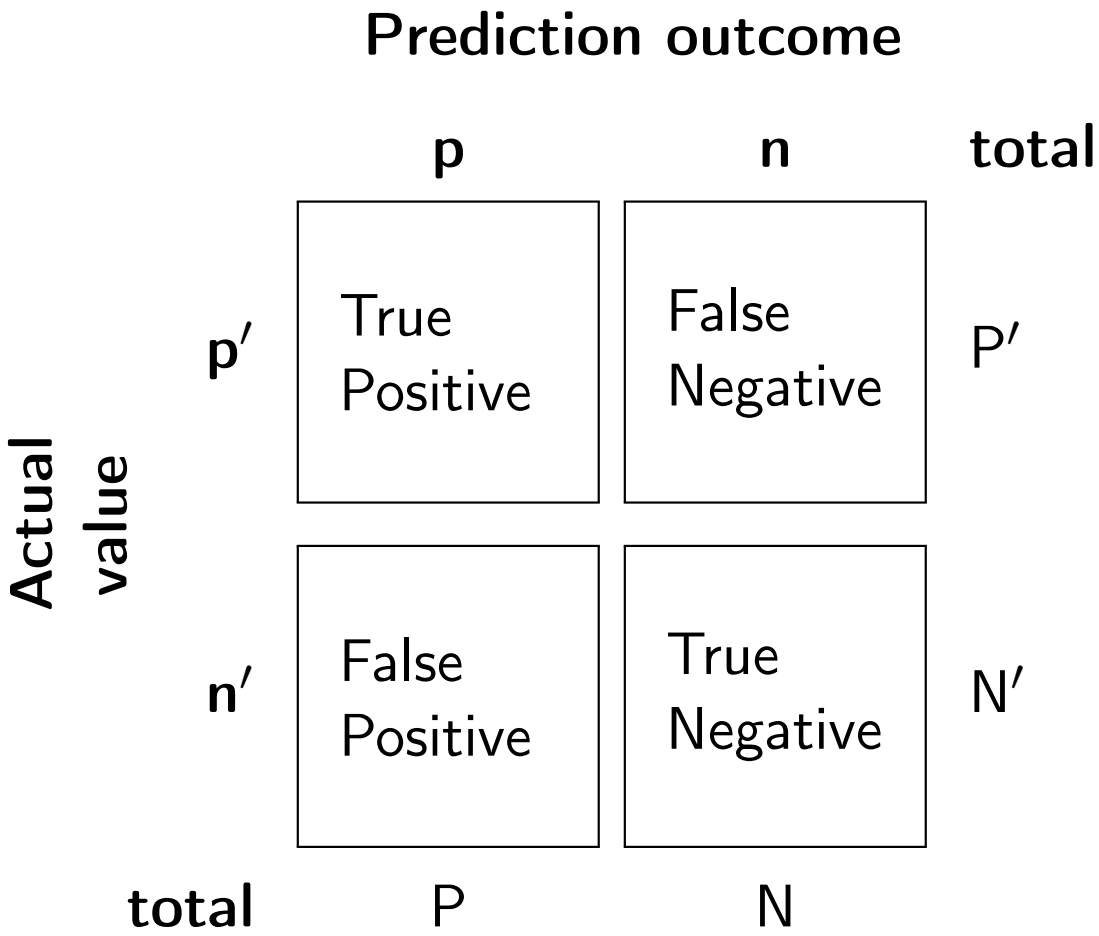
In order to train an algorithm to predict y using x, we need to:

- Visualize and then preprocess your data (centering, scaling, etc.)
- Split the data into training and test (e.g.,80-20)
- Model training and tuning with cross validation

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

Evaluate Model Performance: Confusion Matrix

Confusion matrix, also known as error matrix, visualizes the performance of an algorithm (i.e., contingency table)

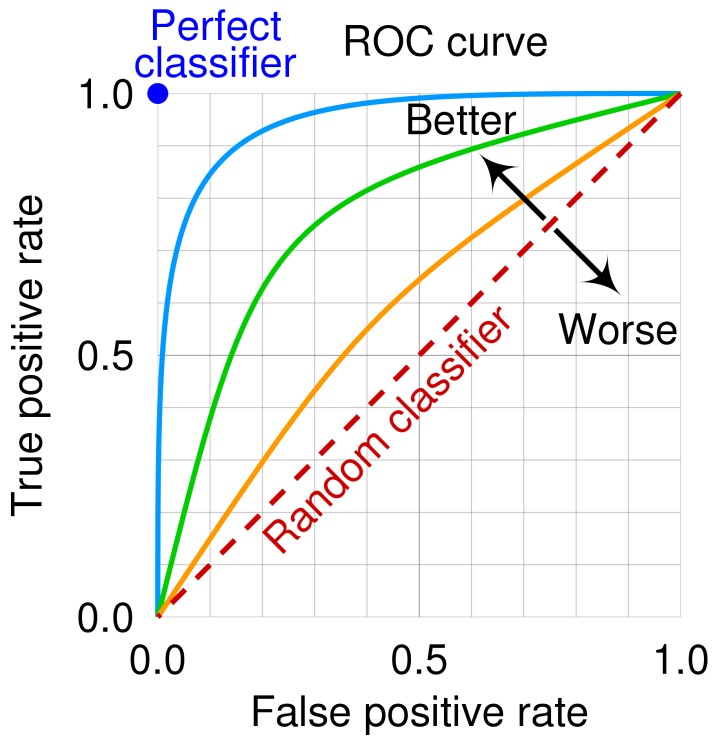


Metrics from the Confusion Matrix

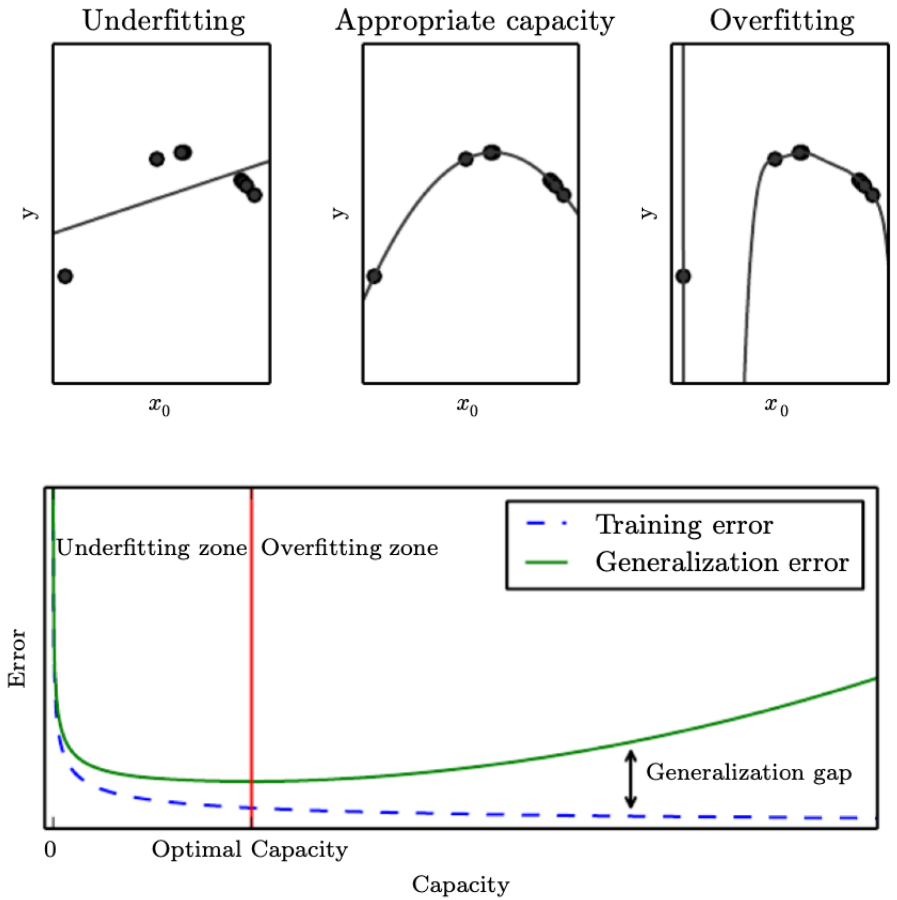
- Precision (positive predictive rate): $\mathrm{PPV} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$
- Recall (sensitivity, hit rate,true positive rate):
  $\mathrm{TPR} = \dfrac{\mathrm{TP}}{\mathrm{P}} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$
- Specificity (selectivity, true negative rate): $\mathrm{TNR} = \dfrac{\mathrm{TN}}{\mathrm{N}} = \dfrac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$
- F1 Score (the harmonic mean of precision and sensitivity):
  $\mathrm{F}_1 = 2 \times \dfrac{\mathrm{PPV} \times \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \dfrac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$
- Accuracy: $\mathrm{ACC} = \dfrac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \dfrac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$

Metrics from the Confusion Matrix

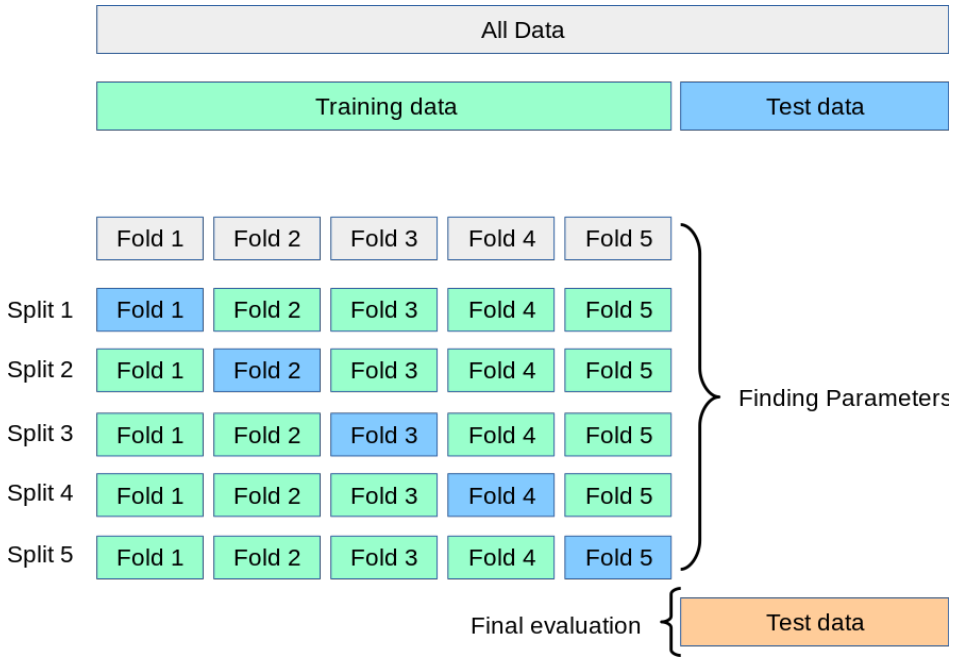
Receiver operating characteristic curve, or ROC curve, is created by plotting the true positive rate (TPR) against the false positive rate (FPR). Sometimes we are also interested in Areas under the Curve (AUC).



Fitting the Model: Underfitting and Overfitting problem



Fitting the Model: Cross-Validation



Fitting the Model: Regularization

Regularization discourages learning a more complex model to avoid overfitting problem.

- $arg\ min_{\alpha,\beta} \sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{p} \beta_{ij} X_i)^2$
- L2 Norm (Ridge Regression):
  $arg\ min_{\alpha,\beta} \sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{p} \beta_{ij} X_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$
- L1 Norm (Lasso regression):
  $arg\ min_{\alpha,\beta} \sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{p} \beta_{ij} X_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$
- λ is the tuning parameter that decides how much we want to penalize the flexibility of our model. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance.
- We often use LASSO to do feature selection since it forces some coefficients to zeros.

Supervised ML for Classification and Regression Problems

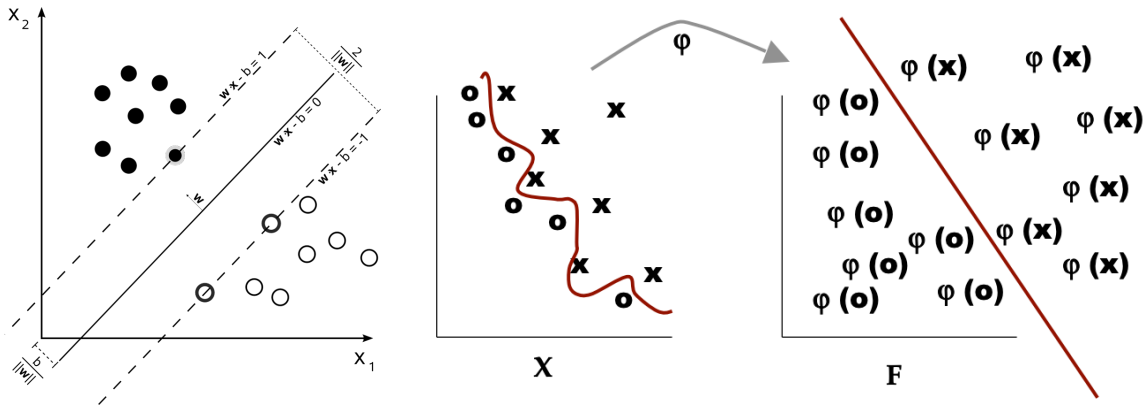
- Classification/regression algorithms take labeled data and learn patterns in the data that can be used to predict a categorical or continous output variable.

Probabilistic ML: Naive Bayes

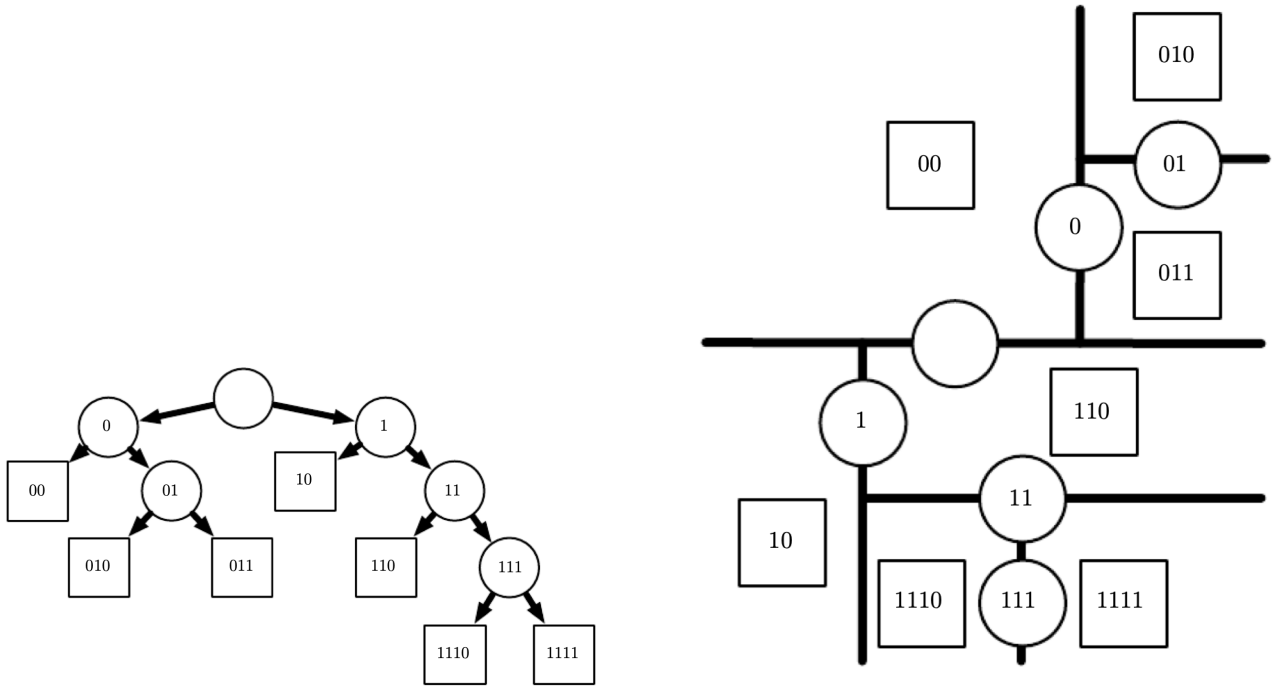
- Bayes Theorem: $\mathcal{P}(\mathcal{A}|\mathcal{B}) = \frac{\mathcal{P}(\mathcal{B}|\mathcal{A})\mathcal{P}(\mathcal{A})}{\mathcal{P}(\mathcal{B})}$
- Assume we have data: $(\mathcal{X}, \mathcal{Y}) = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n), ..., (\boldsymbol{x}_N, y_N)\}$
- $\mathcal{P}(y|x_1, ..., x_n) = \frac{\mathcal{P}(x_1|y)...\mathcal{P}(x_n|y)\mathcal{P}(y)}{\mathcal{P}(x_1)...\mathcal{P}(x_n)}$
- $\mathcal{P}(y|x_1, ..., x_n) \propto \mathcal{P}(y)\prod_{i=1}^{n}\mathcal{P}(x_i|y)$
- $arg\ max_y \mathcal{P}(y)\prod_{i=1}^{n}\mathcal{P}(x_i|y)$
- Multinomial NB, Bernoulli NB, Gaussian NB, etc.

Support Vector Machine

- Apply a kernel function (e.g., gaussian, polynomial,radial-basis) to data and map them to a new space.
- Support vectors are the data points that lie closest to the decision surface (or hyperplane).
- Find the hyperplane that maximizes the margin around the separating hyperplane.

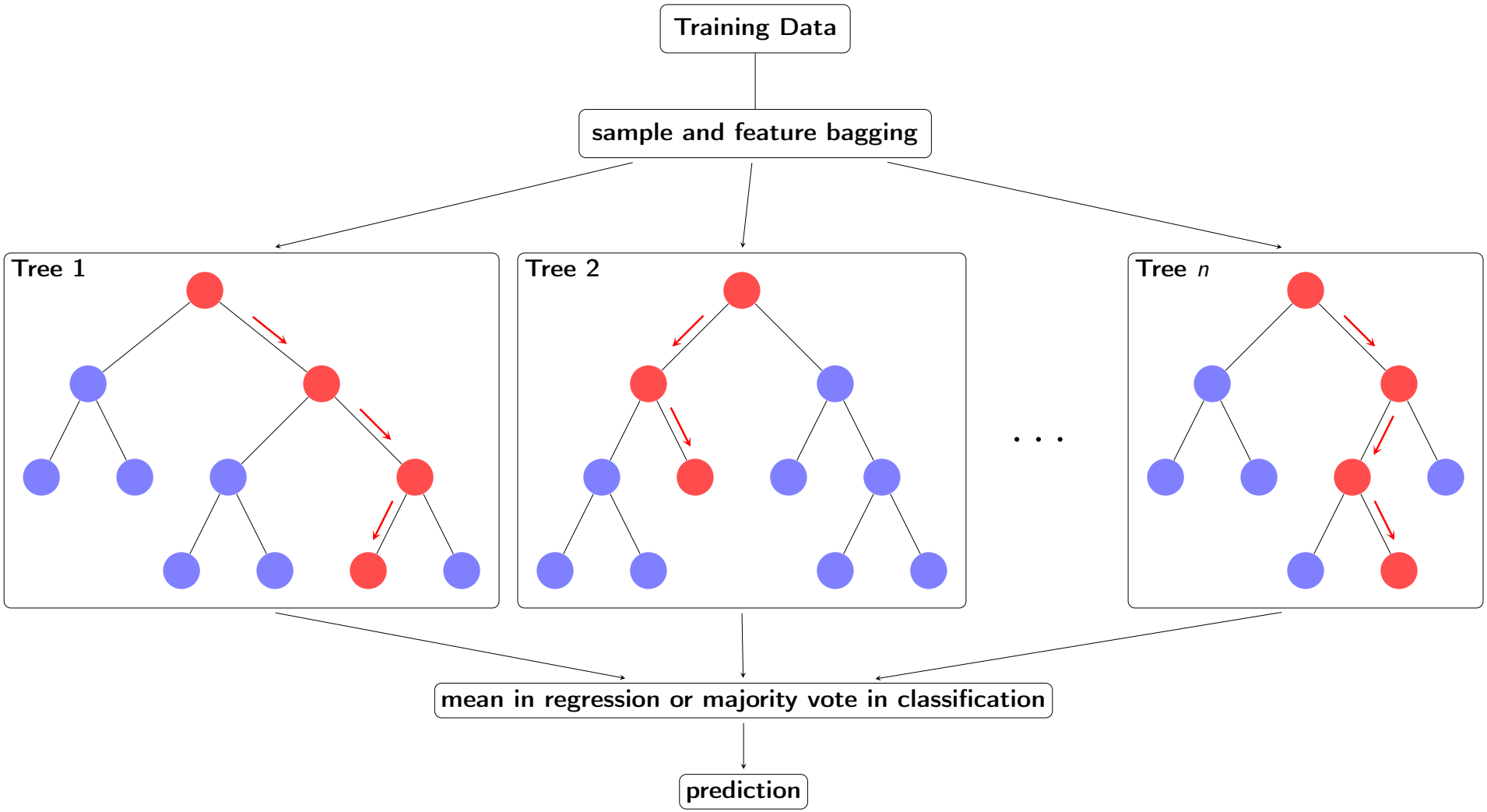


Decision Tree



Diagrams describing how a decision tree works. (Top)Each node of the tree chooses to send the input example to the child node on the left (0) or the child node on the right (1). Internal nodes are drawn as circles and leaf nodes as squares. Each node is displayed with a binary string identifier corresponding to its position in the tree, obtained by appending a bit to its parent identifier (0=choose left or top, 1=choose right or bottom). (Bottom)The tree divides space into regions. The 2D plane shows how a decision tree might divide R2. The nodes of the tree are plotted in this plane, with each internal node drawn along the dividing line it uses to categorize examples, and leaf nodes drawn in the center of the region of examples they receive. The result is a piecewise-constant function, with one piece per leaf. Each leaf requires at least one training example to define, so it is not possible for the decision tree to learn a function that has more local maxima than the number of training examples. (Goodfellow et al. 2016)

Random Forests



An example of using RF

American Political Science Review Vol. 108, No. 3 August 2014
doi:10.1017/S0003055414000306

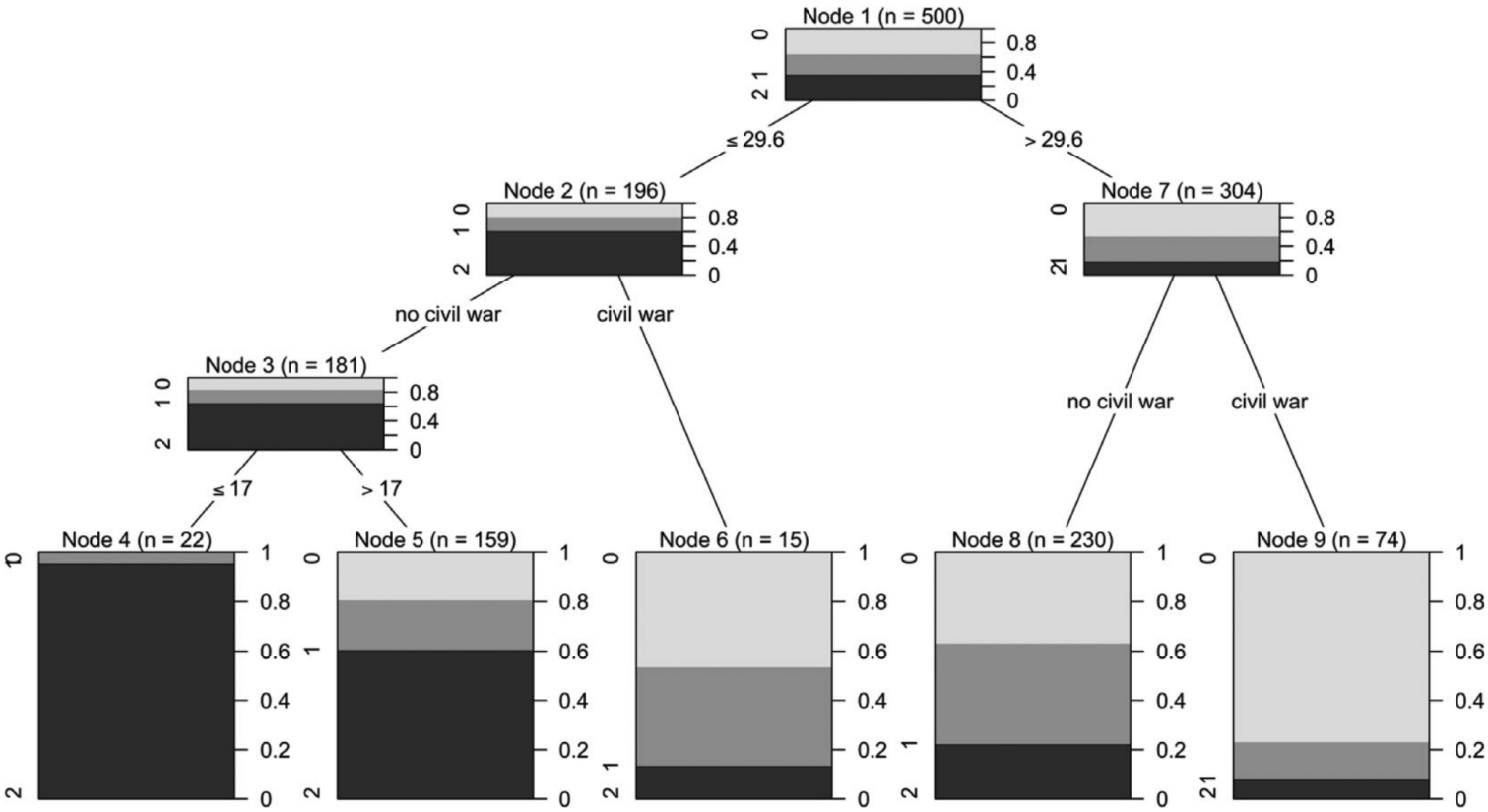
An Empirical Evaluation of Explanations for State Repression

DANIEL W. HILL, JR. *University of Georgia*
ZACHARY M. JONES *Pennsylvania State University*

The empirical literature that examines cross-national patterns of state repression seeks to discover a set of political, economic, and social conditions that are consistently associated with government violations of human rights. Null hypothesis significance testing is the most common way of examining the relationship between repression and concepts of interest, but we argue that it is inadequate for this goal, and has produced potentially misleading results. To remedy this deficiency in the literature we use cross-validation and random forests to determine the predictive *power of measures of concepts the literature identifies as important causes of repression. We find that few of these measures are able to substantially improve the predictive power of statistical models of repression. Further, the most studied concept in the literature, democratic political institutions, predicts certain kinds of repression much more accurately than others. We argue that this is due to conceptual and operational overlap between democracy and certain kinds of state repression. Finally, we argue that the impressive performance of certain features of domestic legal systems, as well as some economic and demographic factors, justifies a stronger focus on these concepts in future studies of repression.*

An example of using RF

FIGURE 1. Results of Using a Decision Tree to Predict the Level of Political Imprisonment using 500 Randomly Sampled Observations and Two Covariates, Civil War and Youth Bulges

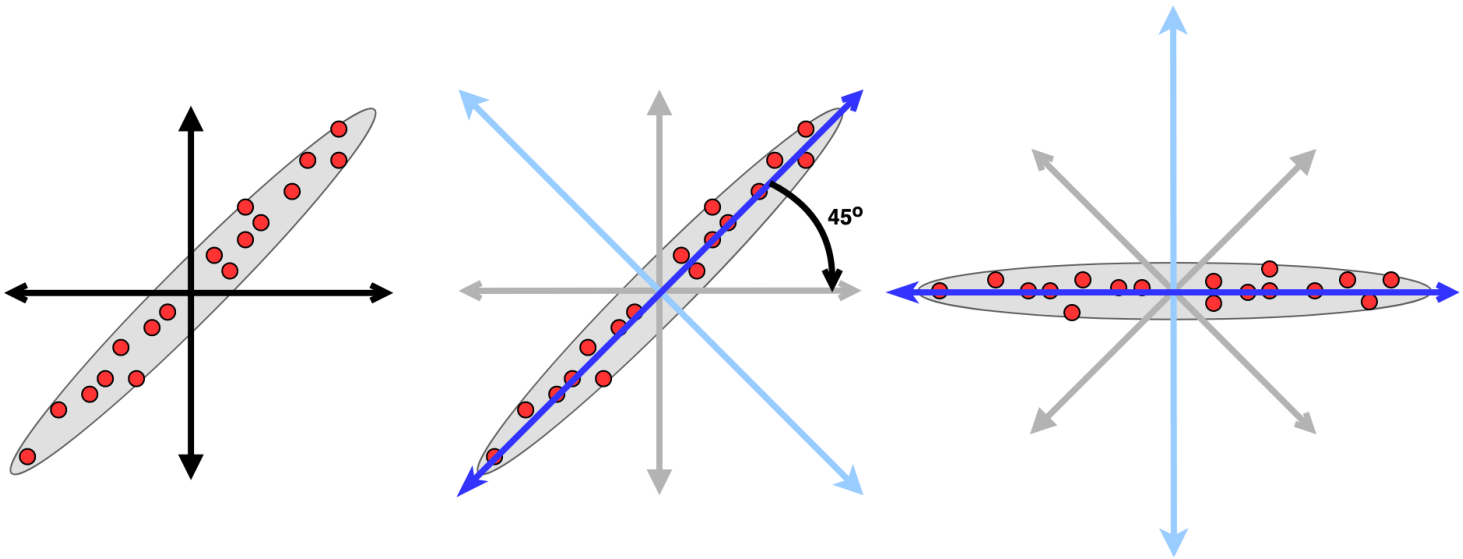


Notes: The number of observations at each node (or partition) is indicated next to the node's number, and the bar plots indicate the distribution of the values of the dependent variable at the node. At node 1 (the parent node) the youth bulges variable is most strongly related to political imprisonment and is selected. The optimal split in youth bulges is 29.6, resulting in two daughter nodes, wherein the process repeats. At each daughter node (2 and 7) civil war is the most strongly related to political imprisonment and is selected, resulting in node 3, where youth bulges is again selected (a new split is found at 17); resulting in terminal nodes (nodes 4, 5, 6, 8, and 9), which are used to predict the dependent variable (the most common category in each terminal node is the predicted value of all observations in that node). Note how the variance of the distribution of dependent variable decreases at each node.

Unsupervised Machine Learning for Dimension Reduction and Clustering

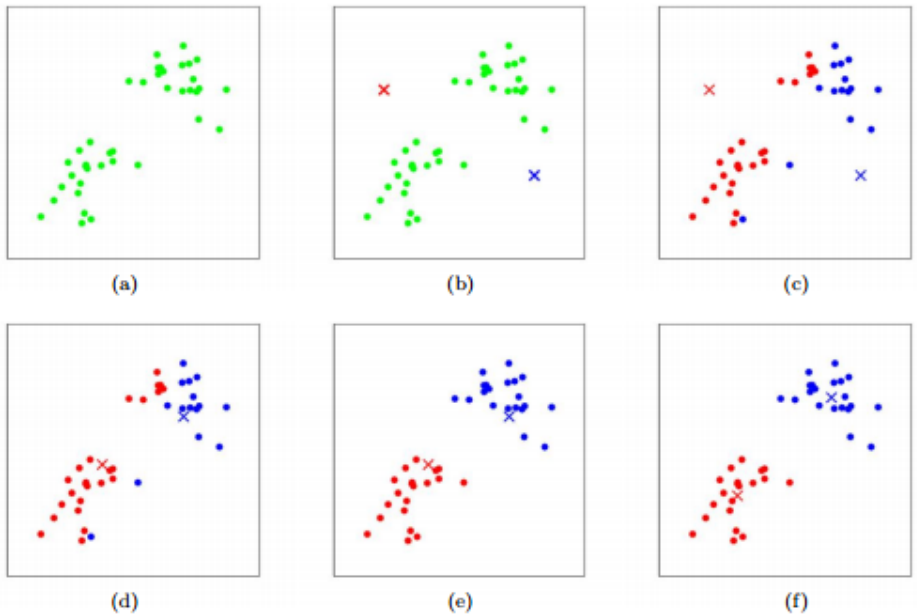
- Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.
- A classic unsupervised learning task is to find the "best" representation of the data.
- Informally, unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate examples.
- Unsupervised ML algorithms: principal component analysis, k-means clustering

Principal Component Analysis



PCA learns a representation that has lower dimensionality than the original input. It also learns a representation whose elements have no linear correlation with each other. We can use PCA as a simple and effective dimensionality reduction method that preserves as much of the information in the data as possible.

Clustering



The k-means clustering algorithm divides the training set into k different clusters of examples that are near each other. K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by "painting" the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it.

Most Common ML Tasks (Goodfellow et al. 2016)

- Classification (binary, multiclass, etc.)
- Regression (predict a numerical value given some input)
- Transcription (e.g., OCR)
- Machine Translation
- Structured Output (e.g., output a vector, like parsing)
- Anomaly Detection
- Synthesis and sampling (e.g., generating new examples similar to training data)
- ...

Table of Contents

What is deep learning?

- ML: searching for useful representations and rules over some input data, within a predefined space of possibilities, using guidance from a feedback signal.
- Deep Learning: learn representations of the data and a function mapping data representations to the output. Here deep refers to the successive layers of representations. Layered representations are learned via models called neural networks, structured in literal layers stacked on top of each other.

Moving beyond conventional ML

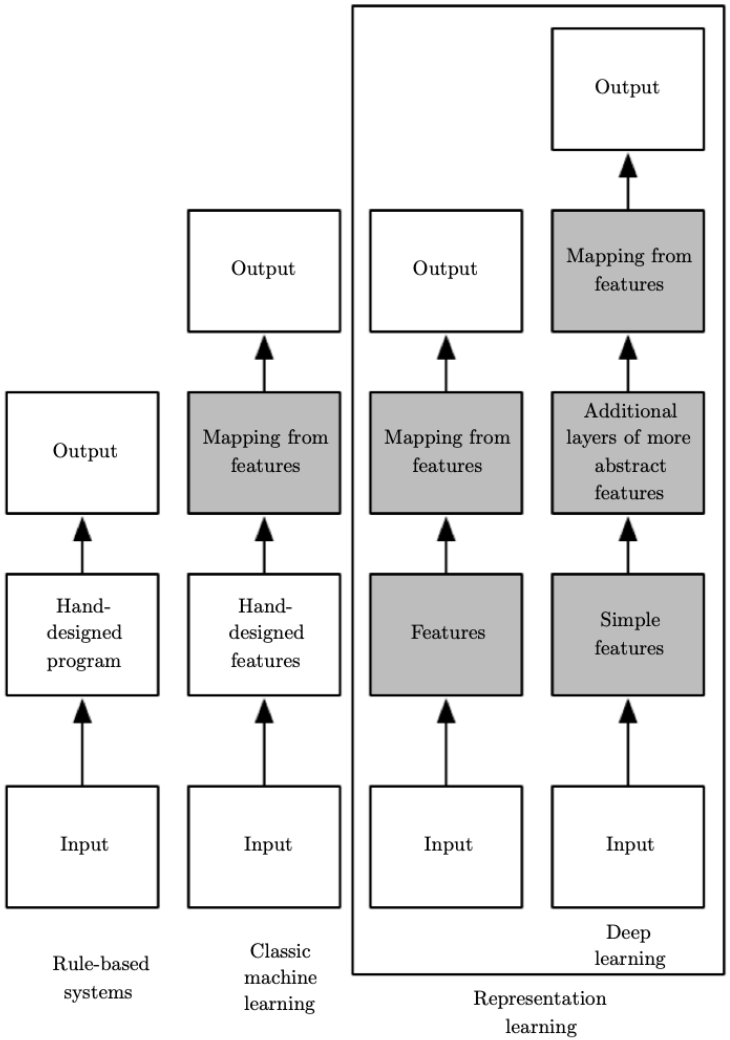
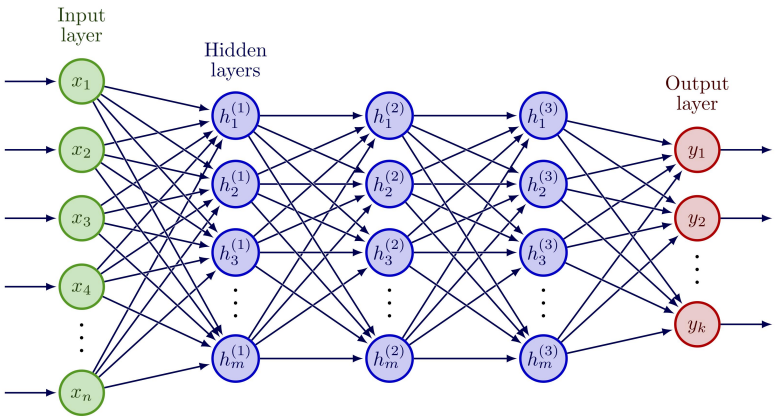
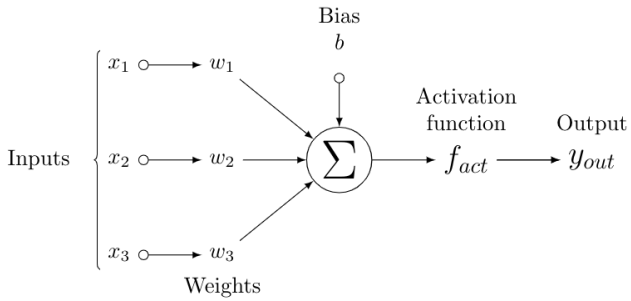


Figure: Goodfellow et al. 2016, Flowcharts showing different parts of AI.

Formal Definition

- In deep learning models, an abstract representation of the data is learned by applying the data to a stack of several simple functions, or learned via neural networks.
- Each function takes as an input the representation of the data created by previous functions and generates a new representation: $\mathrm{f}(\mathcal{X}) = \mathrm{f}^3(\mathrm{f}^2(\mathrm{f}^1(\mathcal{X})))$
- This chain structures the most commonly used structures of neural networks. f^i denotes the ith layer and the overall length of the chain gives the depth of the model (Goodfellow 2016).
- for each neuron, $\mathrm{f}(\boldsymbol{x}; \boldsymbol{w}, b) = \mathrm{g}(\boldsymbol{x}^\top \boldsymbol{w} + b)$.
- The goal of deep learning is to find these parameters including weights and biases that minimize the loss function.


Input layer
Hidden layers
Output layer
x_1
x_2
x_3
x_4
x_n
$h_1^{(1)}$
$h_2^{(1)}$
$h_3^{(1)}$
$h_m^{(1)}$
$h_1^{(2)}$
$h_2^{(2)}$
$h_3^{(2)}$
$h_m^{(2)}$
$h_1^{(3)}$
$h_2^{(3)}$
$h_3^{(3)}$
$h_m^{(3)}$
y_1
y_2
y_k


Bias
b
Inputs
x_1
x_2
x_3
w_1
w_2
w_3
$\sum$
Activation function
f_{act}
Output
y_{out}
Weights

Deep Neural Network as a multistage information distillation process.

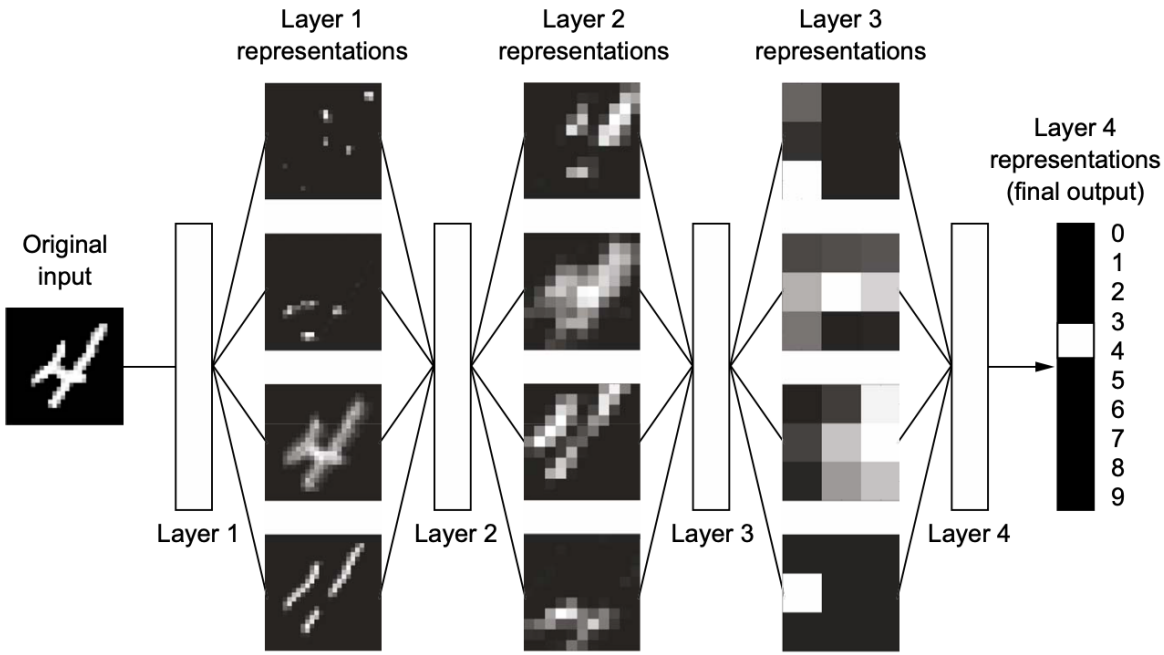


Figure: Chollet 2021. An Illustration of Deep Learning

Understanding How Deep Learning Works

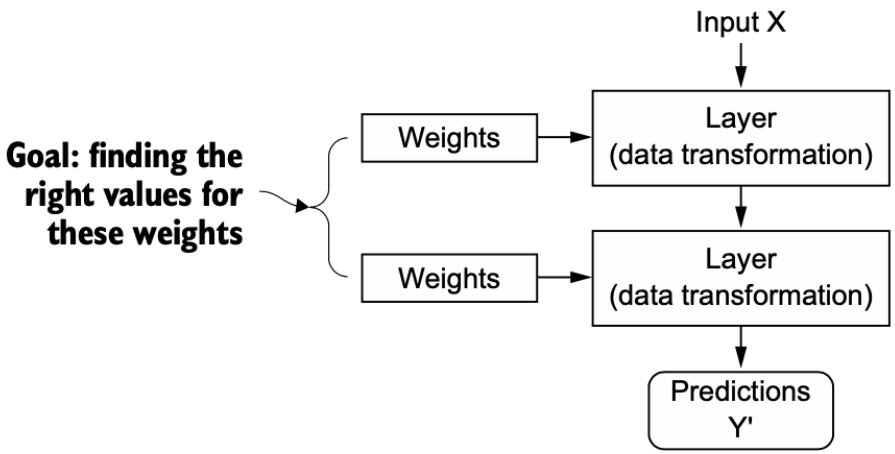


Figure: Chollet 2021, A neural network is parameterized by its weights.

Understanding How Deep Learning Works

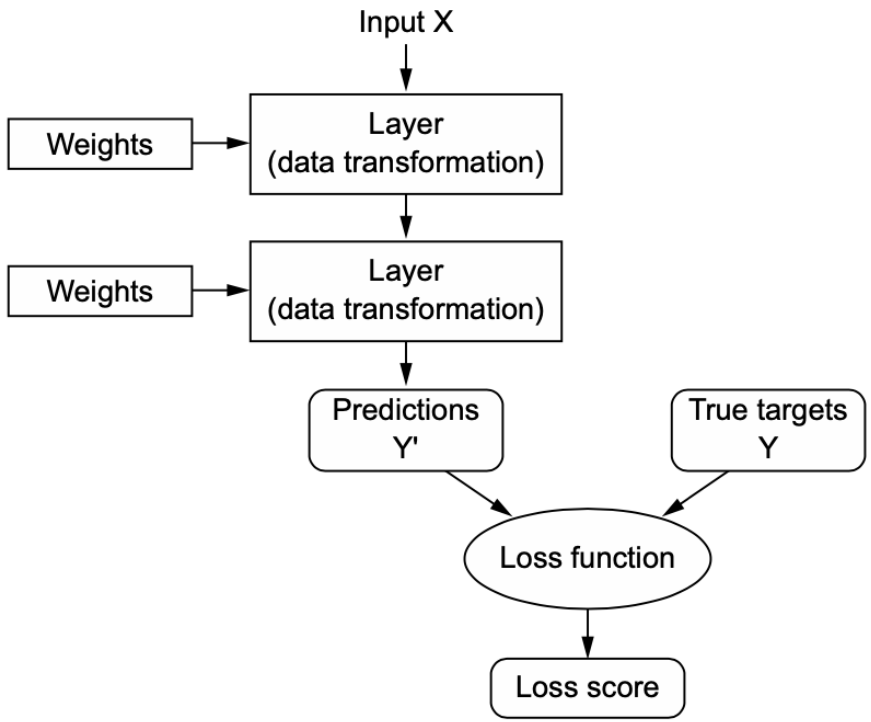


Figure: Chollet 2021, A loss function measures the quality of the network's output.

Understanding How Deep Learning Works

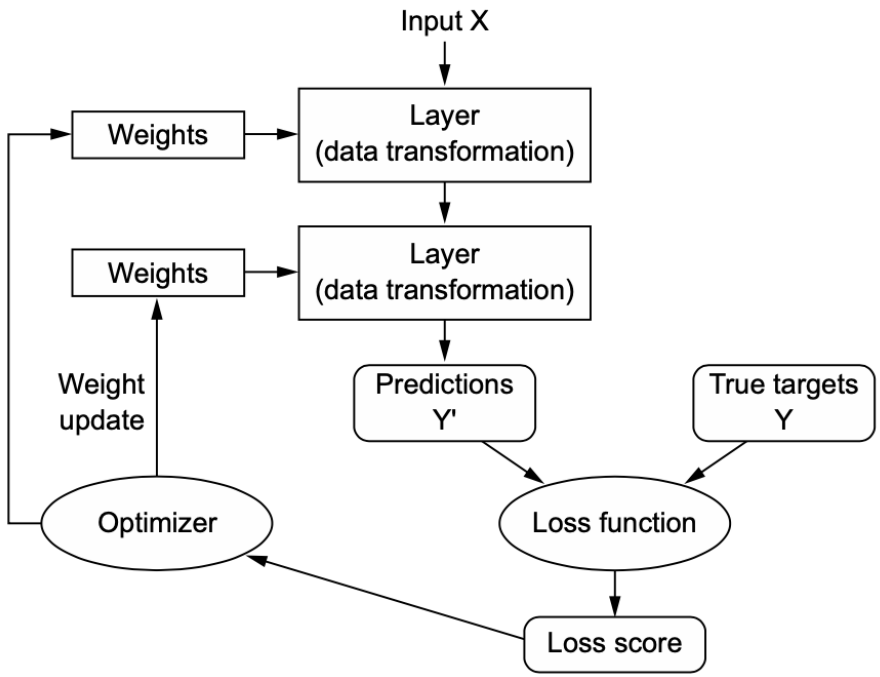


Figure: Chollet 2021, The loss score is used as a feedback signal to adjust the weights.

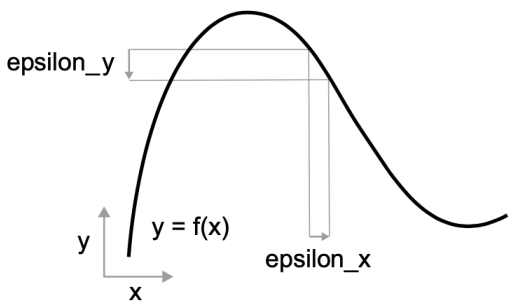
How do we achieve this?

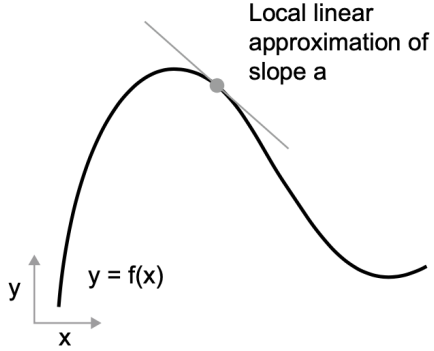
Here is our model $output = relu(dot(input, W) + b)$; W and b called weights and biases. We initialize our model with small random values (random initialization).

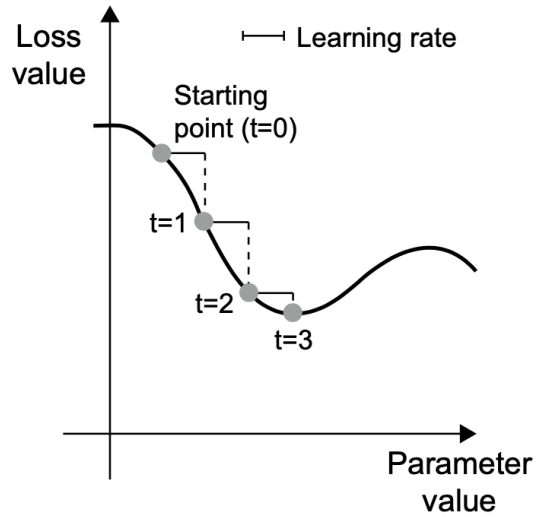
- Draw a batch of training samples, x, and corresponding targets, y_true.
- Run the model on x (a step called the forward pass) to obtain predictions, y_pred.
- Compute the loss of the model on the batch, a measure of the mismatch between y_pred and y_true.
- Update all weights of the model in a way that slightly reduces the loss on this batch.

How do we update weights? Gradient Descent with backpropagation

The goal is to minimize our cost function $\mathcal{J}(\boldsymbol{\theta})$, such as MSE and cross-entrophy.

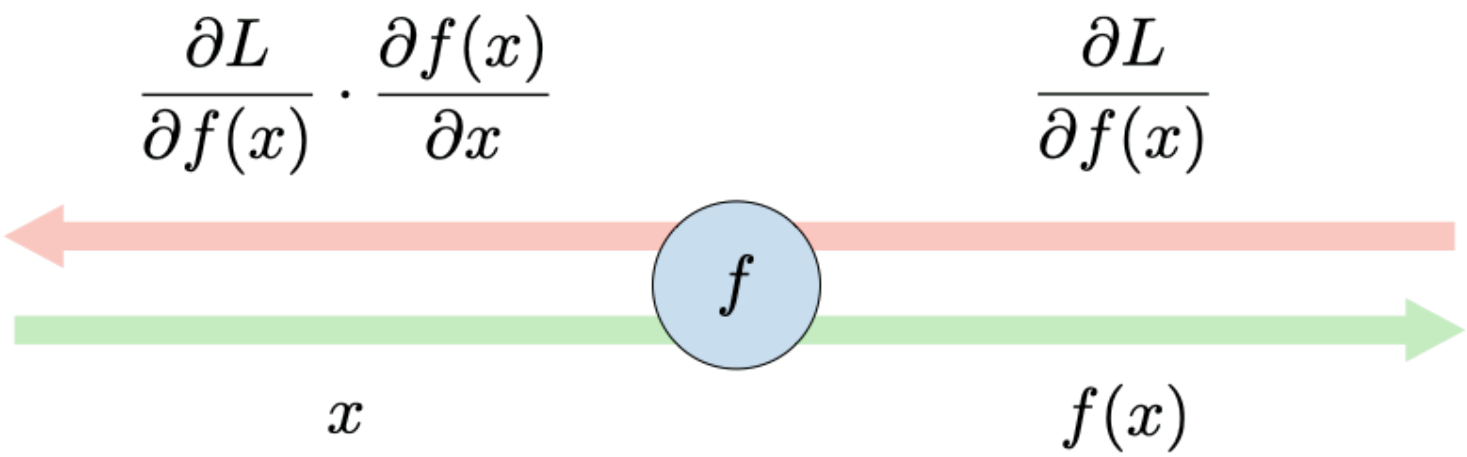






Backpropogation

Backpropagation is a method to update the weights in the neural network by taking into account the actual output and the desired output. The derivative with respect to each weight w is computed using the chain rule.



A computational graph

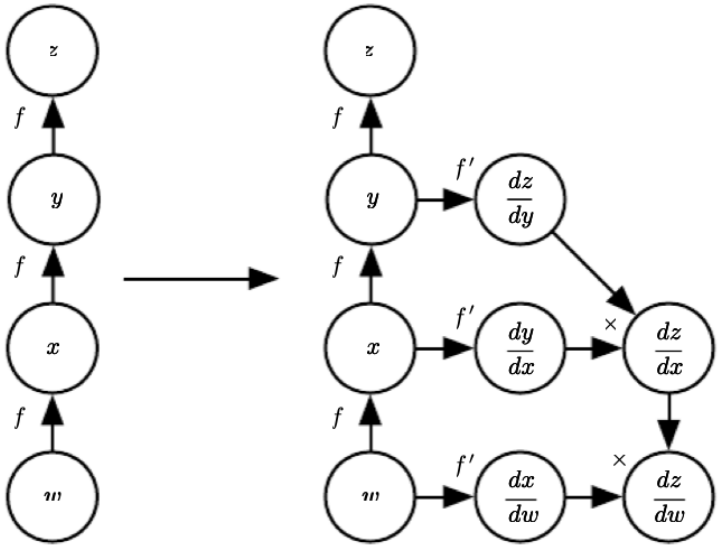


Figure: Goodfellow et al. 2016. An example of the symbol-to-symbol approach to computing derivatives. In this approach, the back-propagation algorithm does not need to ever access any actual specific numeric values. In this example, we begin with a graph representing $z = f(f(f(w)))$. We run the back-propagation algorithm, instructing it to construct the graph for the expression corresponding to $\frac{dz}{dw}$. In this example, we do not explain how the back-propagation algorithm works. The purpose is only to illustrate what the desired result is: a computational graph with a symbolic description of the derivative.

Mini-batch stochastic gradient descent (mini-batch SGD)

- Draw a batch of training samples, x, and corresponding targets, y_true.
- Run the model on x (a step called the forward pass) to obtain predictions, y_pred.
- Compute the loss of the model on the batch, a measure of the mismatch between y_pred and y_true.
- Compute the gradient of the loss with regard to the model's parameters (backward pass).
- Move the parameters a little in the opposite direction from the gradient— for example, $W -= learning\_rate * gradient$—thus reducing the loss on the batch a bit. The learning rate would be a scalar factor modulating the "speed" of the gradient descent process.

Updating Weights

1. Take a batch of training data and perform forward propagation to compute the loss.
2. Step 2: Backpropagate the loss to get the gradient of the loss with respect to each weight.
3. Use the gradients to update the weights of the network.

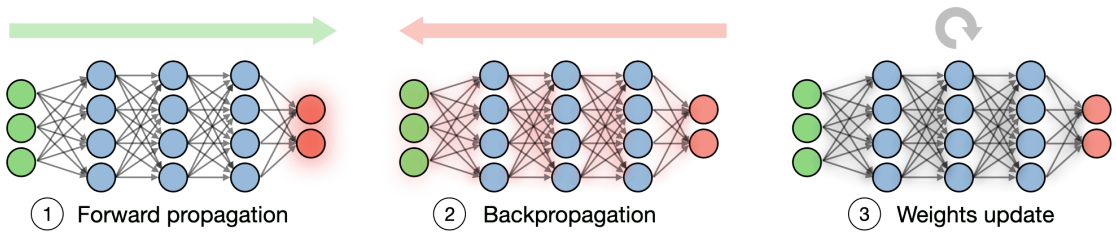


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks`

Optimizing convergence

Method	Explanation	Update of w	Update of b
Momentum	• Dampens oscillations • Improvement to SGD • 2 parameters to tune	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	• Root Mean Square propagation • Speeds up learning algorithm by controlling oscillations	$w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \longleftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	• Adaptive Moment estimation • Most popular method • 4 parameters to tune	$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \longleftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

Optimizing convergence

STRENGTHS AND WEAKNESSES OF SOME POPULAR OPTIMIZERS

Optimizer	State Memory (bytes)	# Parameters	Strengths	Weaknesses
SGD	0	1	1) Excellent generalization (after sufficient training)	1) Prone to saddle points or local optima 2) Slow convergence rate 3) Sensitive to hyper-parameter initialization and learning rate selection
Momentum	4n	2	1) Accelerate in directions of steady descent 2) Not prone to falling into local optima	1) Sensitive to hyper-parameter initialization and the values of learning rate and momentum
Adagrad	∼4n	1	1) Adjust learning rate adaptively 2) Convergence rate is faster on data with sparse features	1) Worse generalization and likely to converge to sharp minimum 2) Gradient vanishing due to aggressive scaling
RMSprop	∼4n	3	1) Adjust learning rate adaptively but less aggressive than Adagrad 2) Convergence rate is faster on data with sparse features 3) Built in Momentum	1) Worse generalization and likely to converge to sharp minimum
Adam	∼8n	3	1) Automatically decay the learning rate 2) Convergence rate is faster on data with sparse features 3) Combines the advantages of Momentum, Adagrad, and RMSprop	1) Worse generalization and likely to converge to sharp minimum

Parameters Tuning

COMMON HYPERPARAMETER TUNING

Hyper-parameter	Description	Strategy
learning rate	Learning rate refers to the step size of updating network weights. It can be constant, or variable. As mentioned in optimizer part, optimization algorithms determine different learning rates. In order to make the gradient descent perform better, the value of learning rate should be set in an appropriate range. If the learning rate is too small, the convergence speed will be slow; if it is too large, the parameters will oscillate back and forth on both sides of the optimal solution.	Usually, the initial value of parameters is far away from the optimal value. With iterations, the value will be closer to the optimal value. The basic idea of the adjustment strategy is to make learning rate gradually decrease with the training, i.e., using a relatively large learning rate at the beginning to speed up training, and then using a small learning rate to improve training stability, to avoid skipping the optimal value.
epoch	The epoch refers to the number of times that the whole training set is input to the neural network for training. When the gap of accuracy between training set and validation set is small, current epoch is considered appropriate. Otherwise, if the gap decreases, it means that the epoch is too small, resulting in underfitting; if the gap increases, the epoch is too large, resulting in overfitting.	If computing resources permit, set epoch as large as possible at first, and then stop training when training loss is stable or training accuracy is similar to validation accuracy. If computing resources are limited, train as many epochs as possible. Note that too small epoch will cause model underfitting, and too large epoch will cause model overfitting.
mini-batch size	Mini batch size is the number of samples sent to the model in each training. In the process of network optimization, too small batch size means that the number of samples input into the network is too small, i.e., not representative, and the noise increases correspondingly, which makes the network difficult to converge. Too large batch size makes the gradient direction nearly stable and easy to fall into the local optimum or saddle point.	In practice, mini-batch size is usually set to dozens to hundreds, depending on GPU's memory and computing core, and is usually set to the power of 2, because it can take full advantage of GPU performance. Moreover, when using the second-order optimization algorithm, such as Conjugate Gradient (CG) and L-BFGS, it is often necessary to use a large mini-batch size because if the first derivative is not well estimated, there will be a huge error in estimating the second derivative.
number of conv layers and conv kernels	Each conv layer usually contains different-level features. Shallow layers can detect edge features, local features, and other low-level features of the image, while deep layers can detect global features. Usually, networks with more layers and kernels have ability to represent more complex features, but meanwhile, are harder to train.	Unfortunately, to my best knowledge, there is no theoretical support or universal rule to determine the number of layers and kernels beforehand. Many proposed fancy structures depend on trial and error with the help of personal experience.
size of conv kernels	On the premise of the same receptive field, the smaller the convolution kernel is, the less the parameters and computational complexity are. Specifically, when the convolution kernel size is larger than 1, it can increase the receptive field. If the convolution kernel with even size is used, even if padding is symmetrically added, the size of input feature map and output feature map cannot be kept unchanged.	How to determine convolution kernel size of each layer has no solid theoretical basis. Researchers usually set it based on experience. Fortunately, many literatures have proved that several small convolution kernels can reach the receptive field of a large convolution kernel with fewer parameters. Hence, based on the left description, 3×3 kernel is frequently used. Besides,many novel kernels, such as 1×1, 1×n, and n×1 kernels, are worth trying.

Why Should Social Scientists Use ML?

- Discovery
- Measurement
- Causal inference
- Any others?

What kinds of deep learning tools can we use in social science research?

- Convolutional Neural Networks (e.g., image analysis)
- Recurrent Neural Networks (e.g., sequential data analysis)
- Any others?

We start with a simple forward-feed neural network.

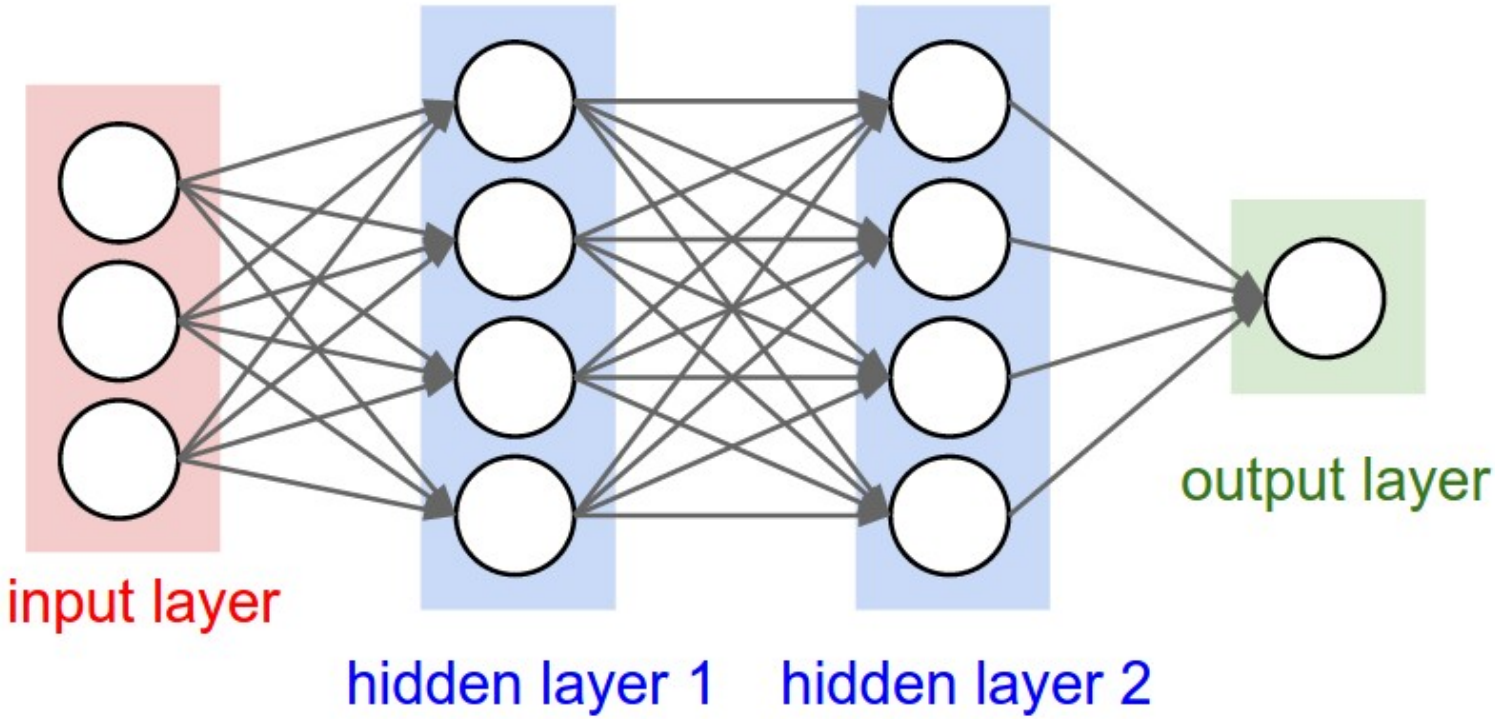


Figure: A regular 3-layer Neural Network.

The traditional CNN architecture stacks ConvNets, pooling layers, and fully connected layers together.

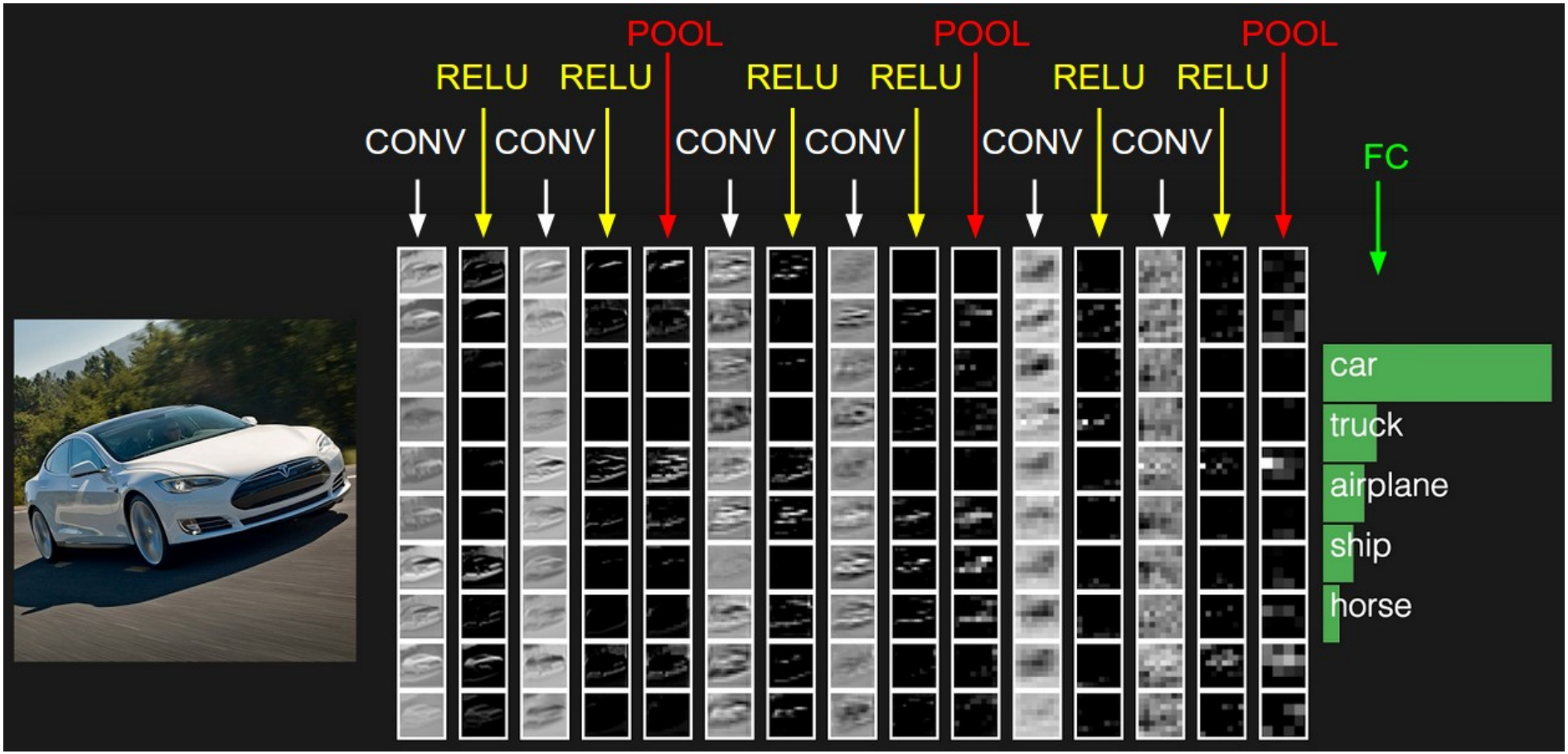


Figure: `https://cs231n.github.io/convolutional-networks`

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters.

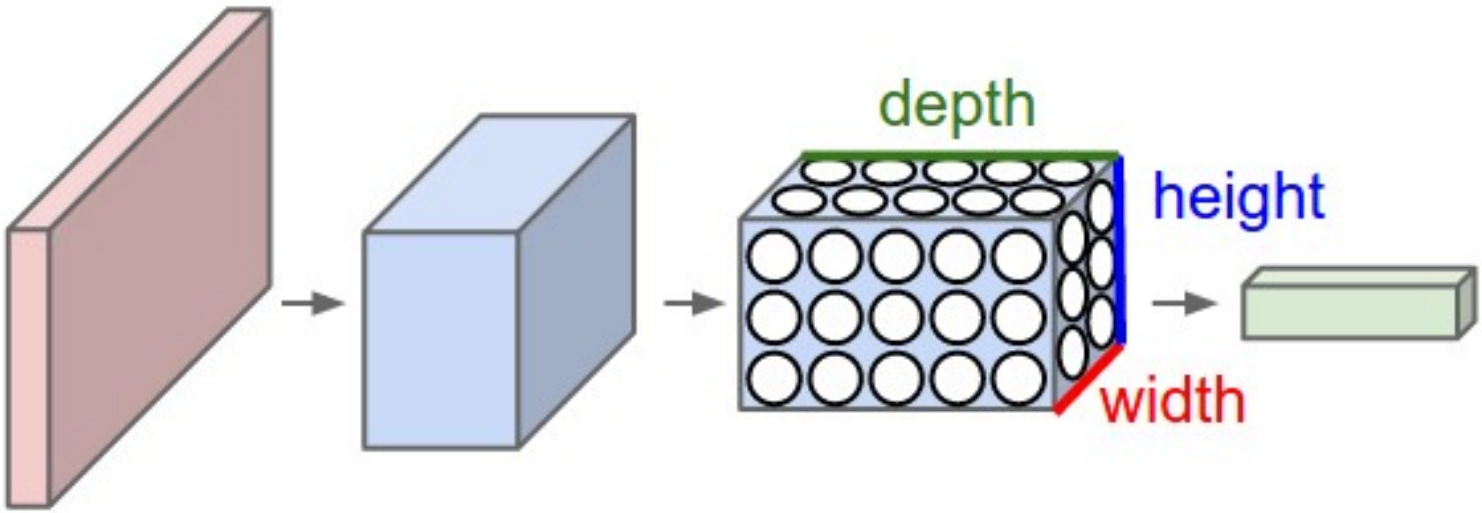


Figure: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.

Figure: `https://cs231n.github.io/convolutional-networks`

The convolution layer (ConvNet) uses filters that perform convolution operations as it is scanning the input I with respect to its dimensions. Its hyperparameters include the filter size F and stride S. The resulting output O is called feature map or activation map.

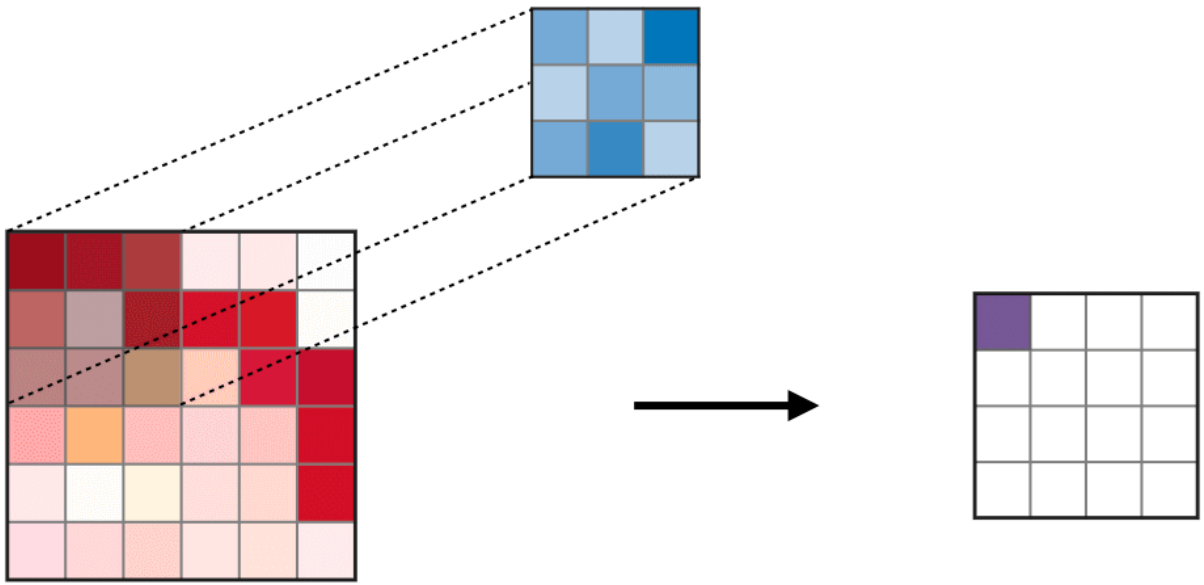


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks`

The pooling layer is a down-sampling operation. You can do max pooling or average pooling.

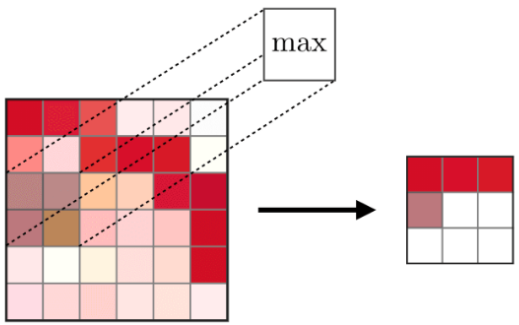
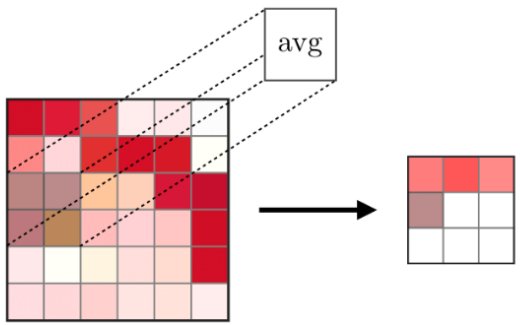
Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks`

The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons. If present, FC layers are usually found towards the end of CNN architectures and can be used to optimize objectives such as class scores.

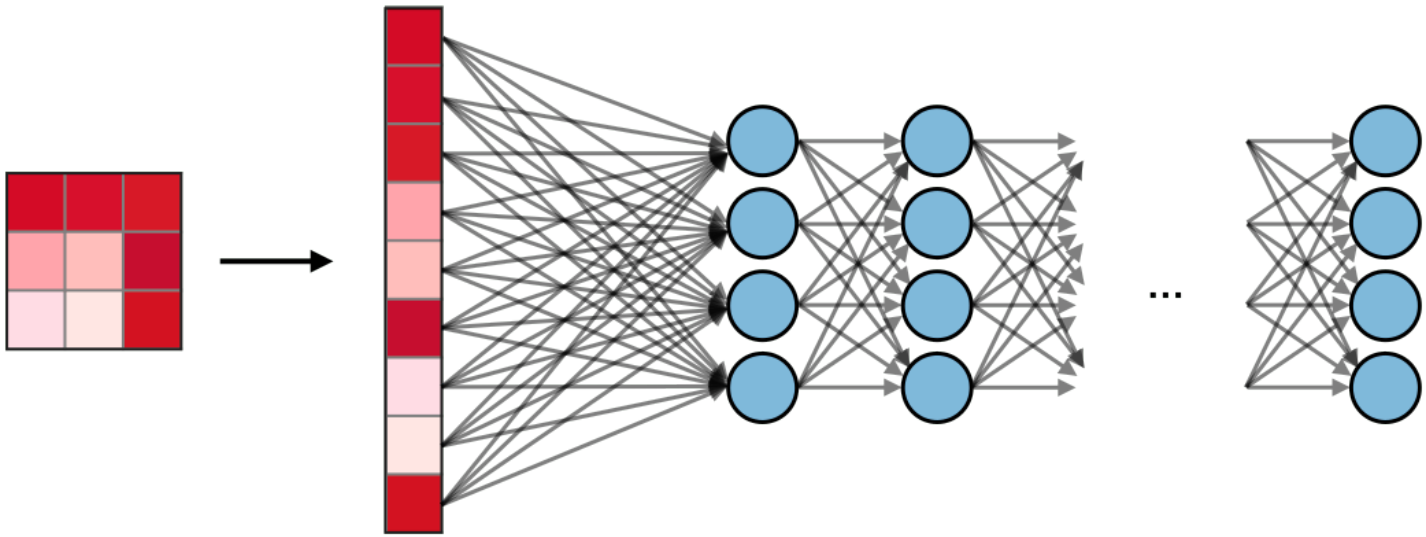


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks`

Sometimes also dropout layer. Dropout is a technique used in neural networks to prevent overfitting the training data by dropping out neurons. It forces the model to avoid relying too much on particular sets of features.

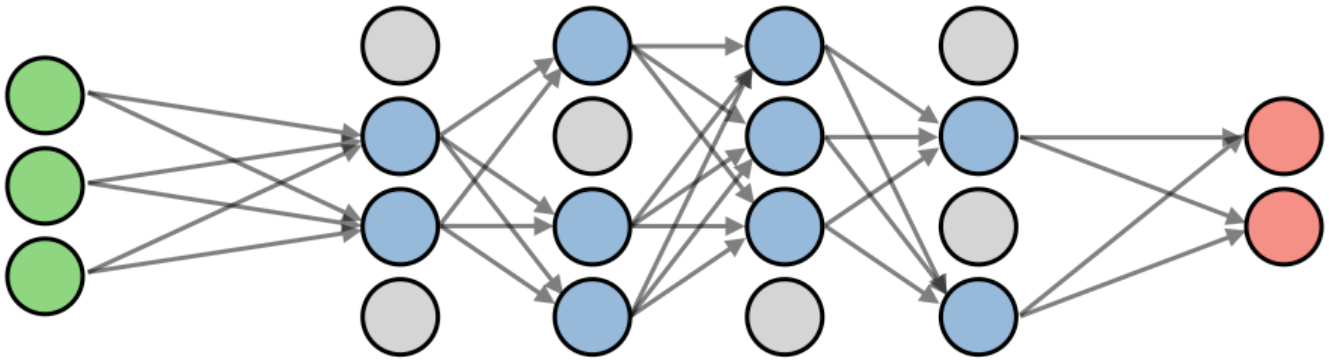


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks`

Recurrent Neural Networks (RNN) are often used to process sequential data, such as machine translation, speech analysis, sentence generation, etc. Because RNNs allow previous outputs to be used as inputs while having hidden states.

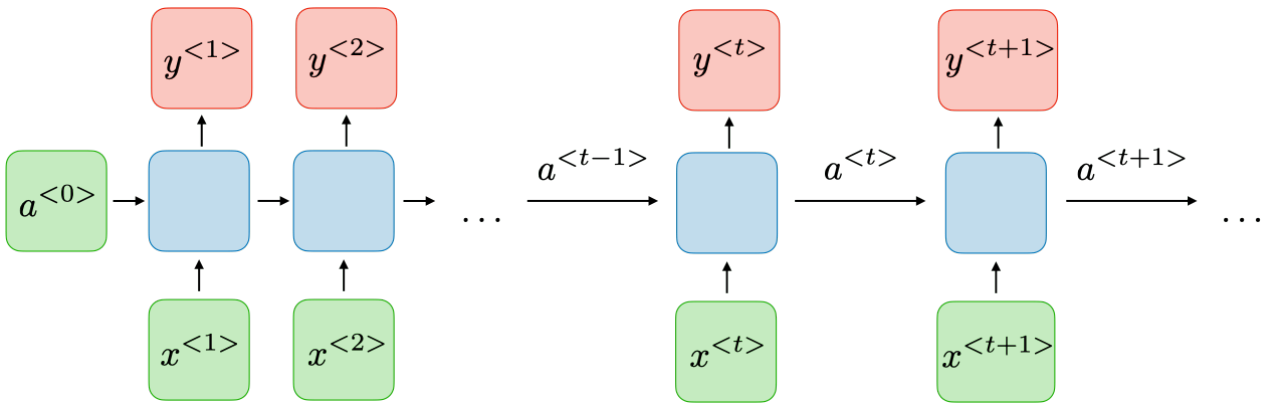


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

RNN cell

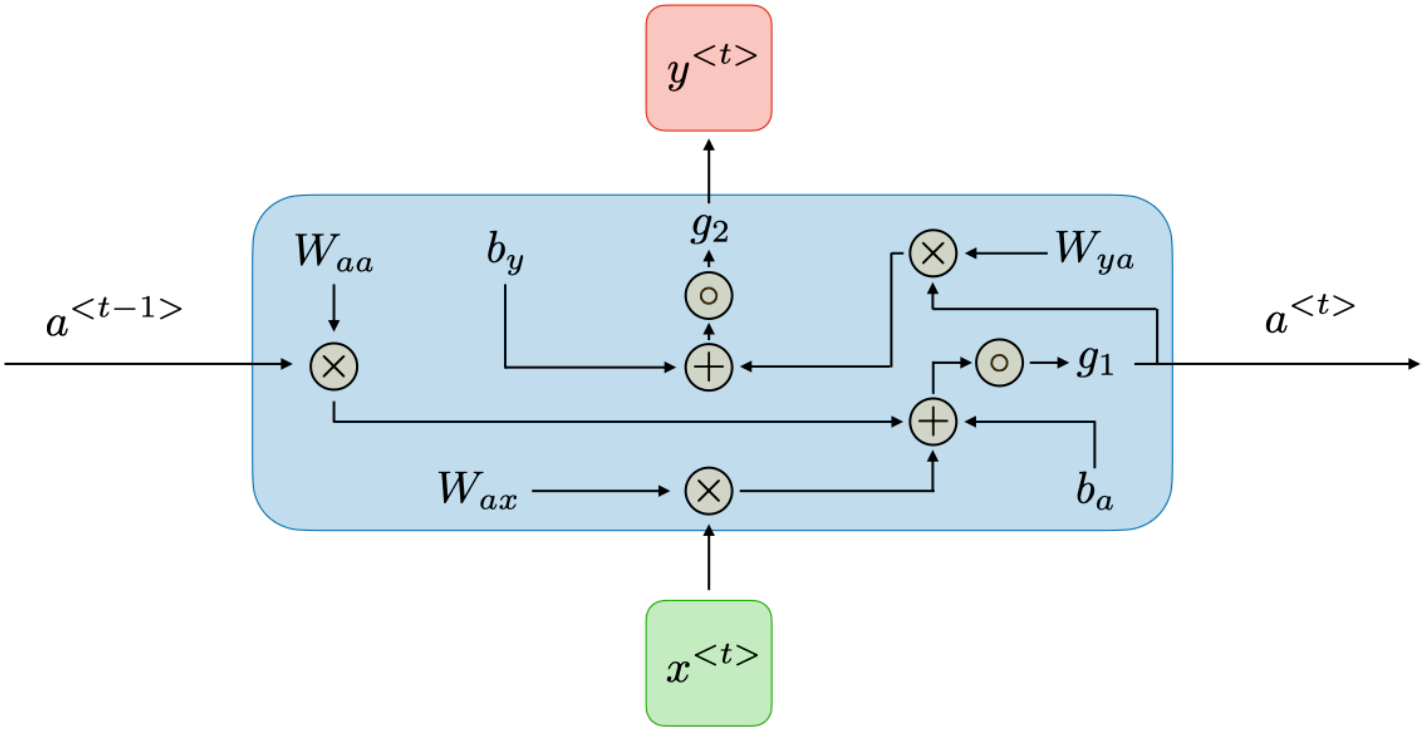


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

RNN Applications: music generation

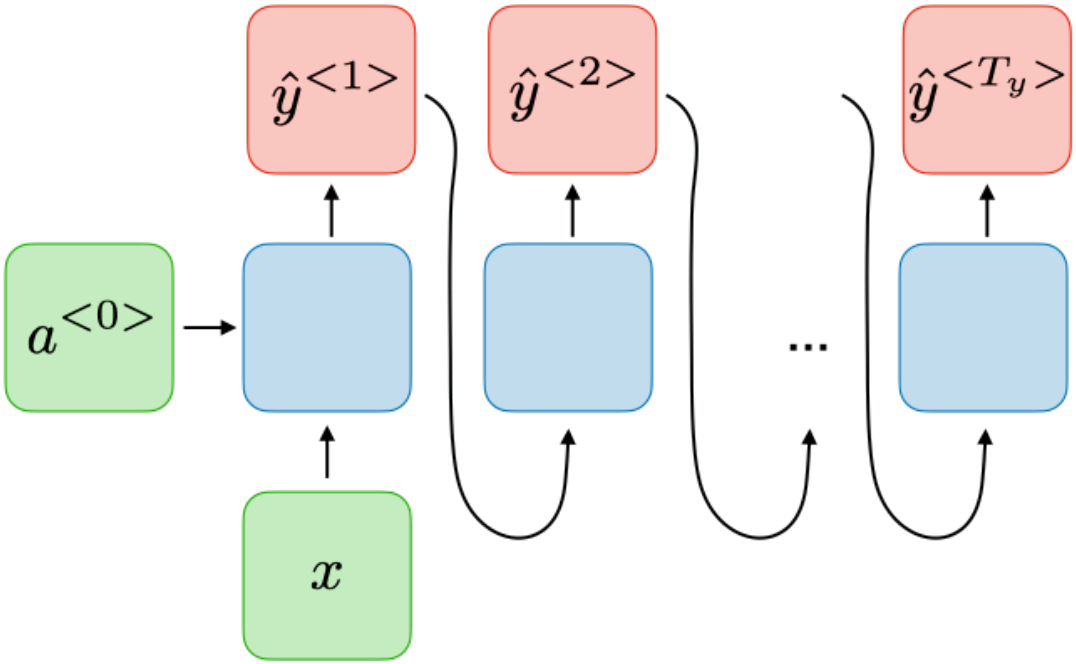


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

RNN Applications: sentiment classification

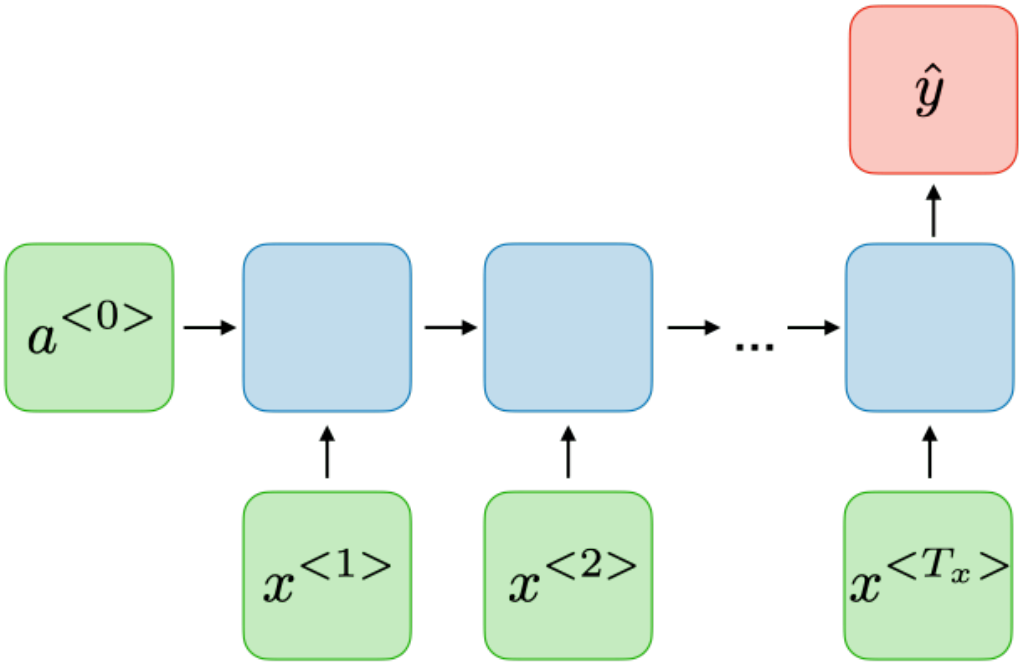


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

RNN Applications: name entity recognition

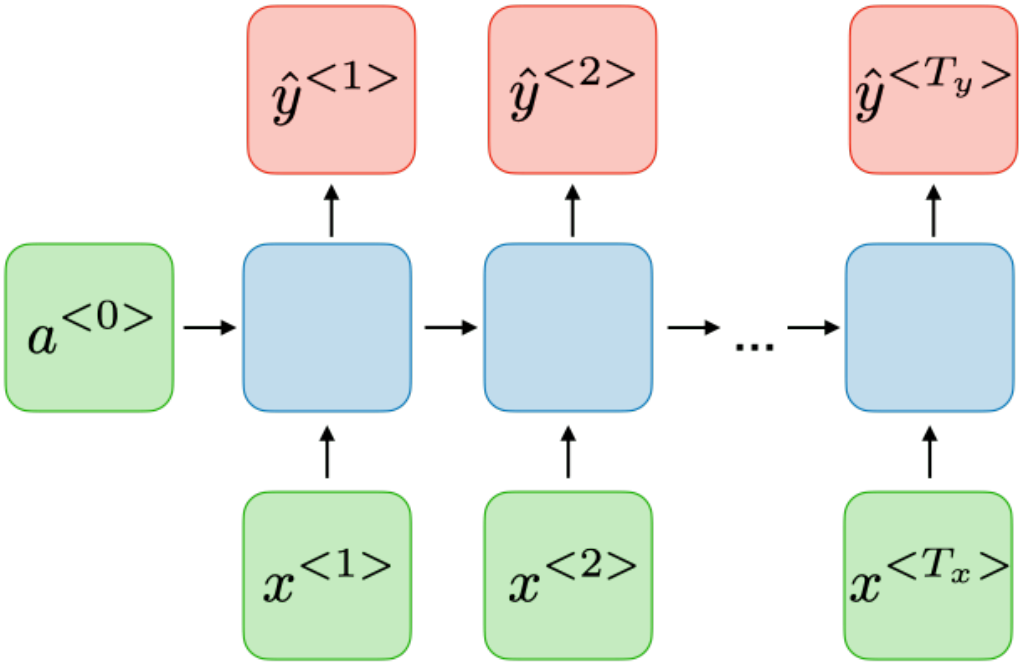


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

RNN Applications: machine translation

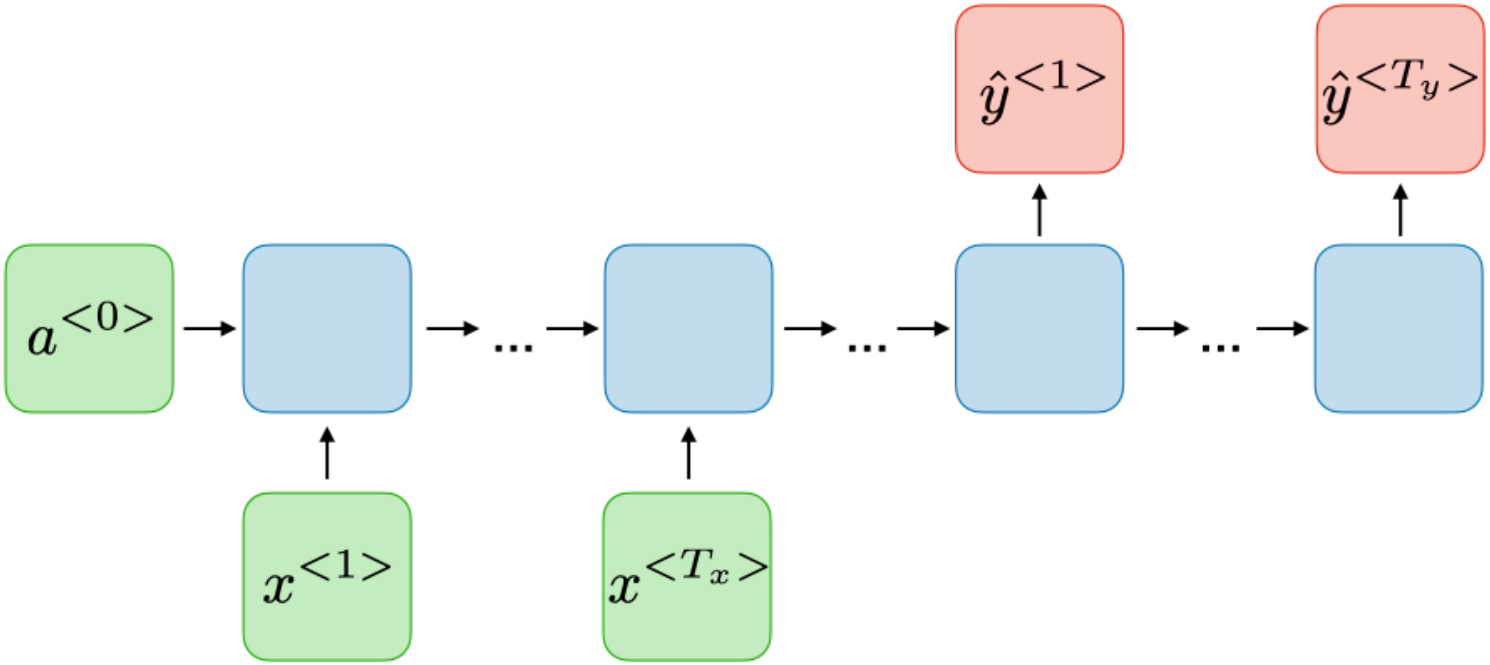


Figure: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`

The traditional RNN has some issues. e.g., gradient vanishing/exploding

It is difficult for RNN to capture long term dependencies because of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers.

Long-short term memory units (LSTM) and gated recurrent units (GRU) were created as the solution to short-term memory.

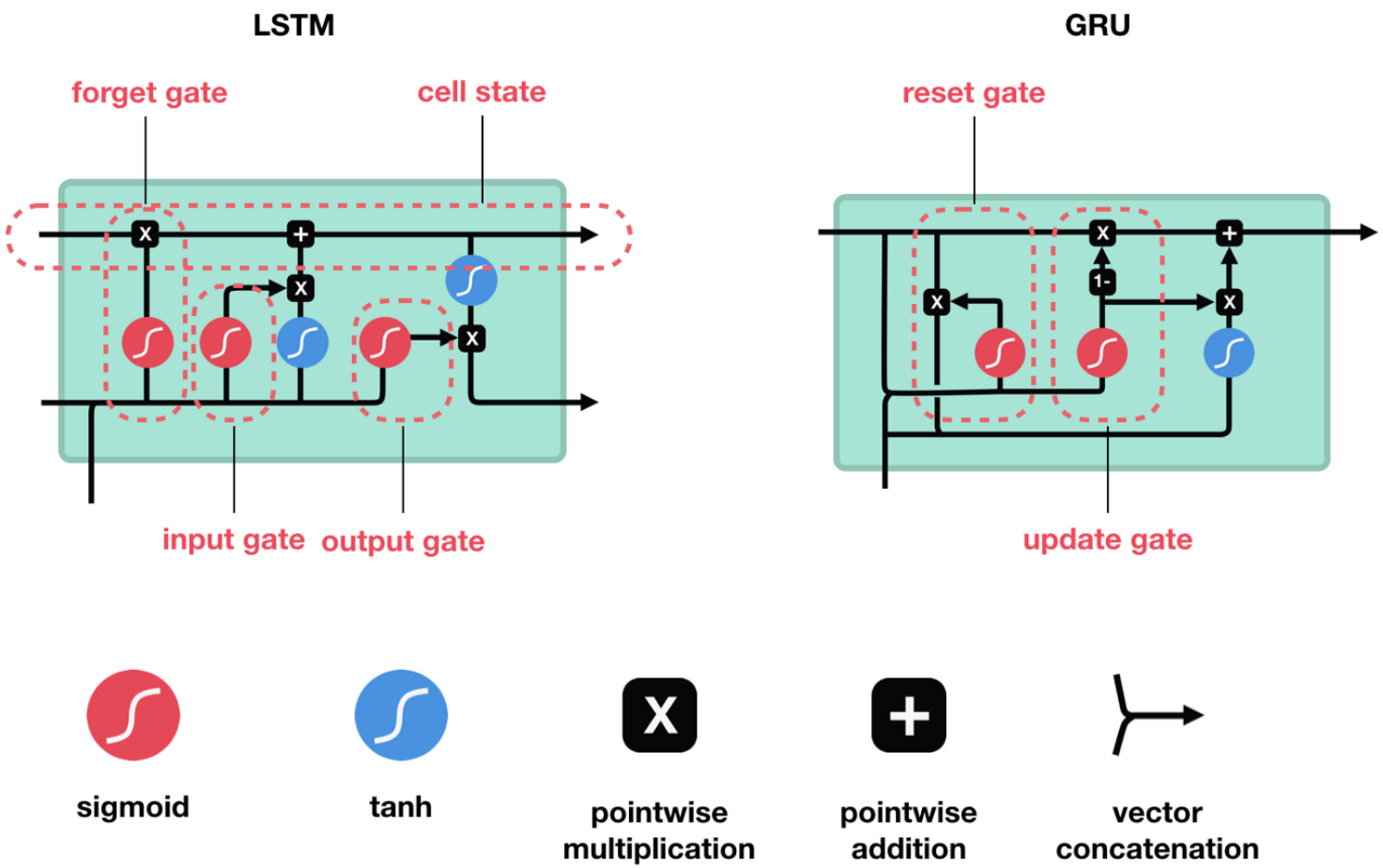


Figure: watch https://youtu.be/8HyCNIVRbSU; see https://tinyurl.com/yhfb3a36

Two activation functions

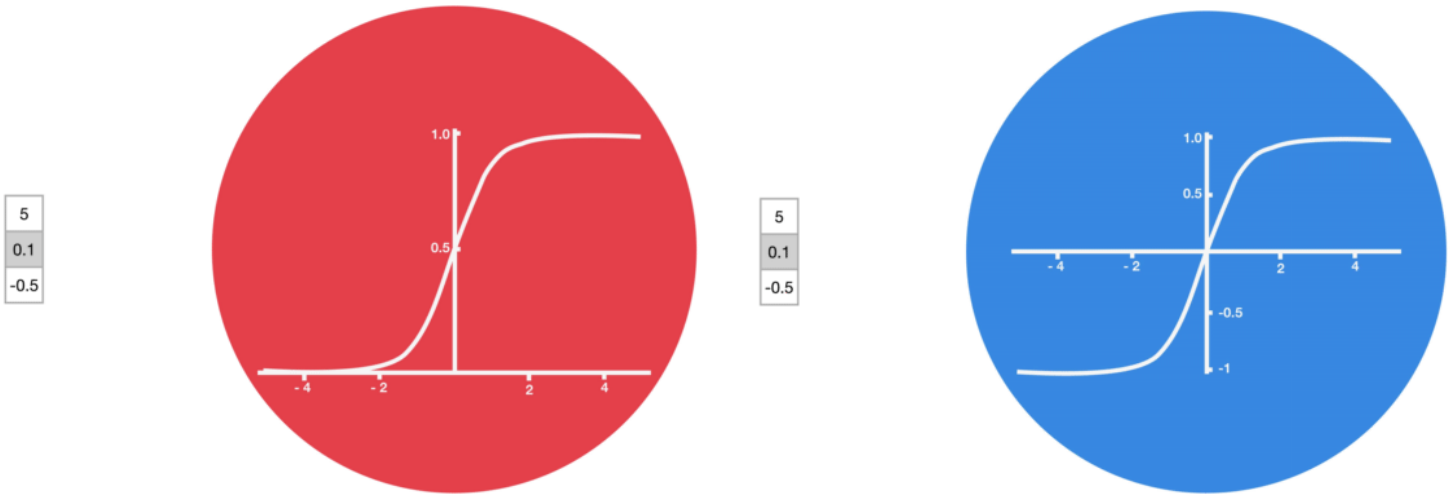


Figure: see `https://tinyurl.com/yhfb3a36`

Let us focus on LSTM. The core of LSTM is the cell state and its various gates.

- The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the "memory" of the network.
- The gates (forget gate, input gate, and output gate) can learn what information is relevant to keep or forget during training.

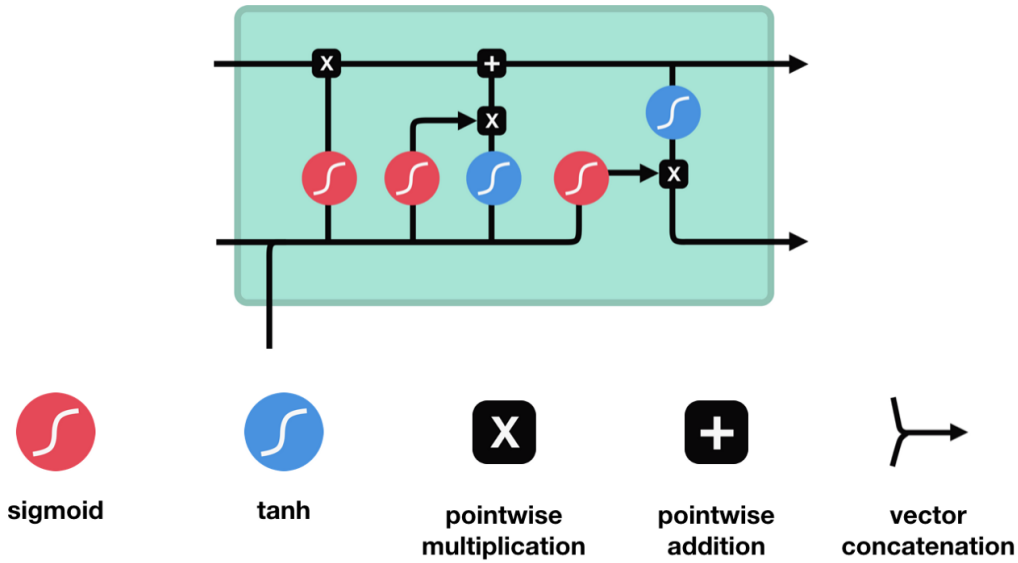


Figure: https://tinyurl.com/yhfb3a36

YongjunZhang.com | @DrJoshZhang
Thank you!
Following us on Twitter: @SBU_Sociology;@IACSComputes