Weekly meeting 6

Dr. Doina Bein
Friday, June 23, 10:30am-12pm

Surveys to be completed

To be done today, before starting research:

CIC-PCUBED Pre-event survey:

https://fullerton.qualtrics.com/jfe/form/SV_6YIVSkC6hLxbunA

Project 1: Data Science

What you need to do: topics & objectives

Objective 1: Learn Python using some textbook or some online courses such as (https://www.codecademy.com/learn/learn-python). Shared by Stephanie Pocci: Learn Python in a couple hours. This YouTuber does a very beginner-friendly crash course about the capabilities of Python and its uses. Here is the link: https://www.youtube.com/watch?v=rfscVS0vtbw

Objective 2: Learn how to use Jupyter Notebook. Start here http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

Objective 3: For data science, find a suitable dataset and start training some neural network using with Google tensorflow.

Logistics for all students

- Who is participating: list of current research students and their availability
- Research will be conducted virtually during the week with in-person meetings throughout the week
- Zoom meetings for me to teach new topics and for you to participate in open discussions
- Support:
  - If needed, you can meet me
    Zoom: Mon, Tu, Wed from 8:30-10:25 am
    IN PERSON: Mon, Tu, Wed from 8:30-9:30 am, Thursday 8:30-10am or by email
  - CIC-PCUBED peer mentor: (tentative) availability

Logistics for all students (contd.)

- Make a copy of this GDoc Work schedule, share the Gdoc copy with me, and maintain it weekly and daily; due at the end of Week 2
- Before the end of week 3, make a copy and maintain your Proposed work by individual or teams of up to three; due by the end of Week 3
- Complete your availability here; try to have it consistent over the 7 weeks such that it will be easy to partner in the project
- Group projects: to be decided; sample list here
- Oral or poster presentations: tentatively scheduled for Friday, July 28, from 8:30am-12:30 pm and if needed, from 1:30-4 pm

Please checkout:

- Other websites and ebooks
- Websites with free datasets
- More resources on selected topics
- If you find good, free resources, please share it by email or during weekly meetings
- Next meeting: I will lecture on ZOOM on Data Science:
  Thursday, June 29, from 10-11am (time changed)

Progress on Learning Python

- Free course: https://www.codecademy.com/learn/learn-python
- Free course: https://www.kaggle.com/learn/python
- Youtube video (about 4 hours): https://www.youtube.com/watch?v=rfscVS0vtbw

Data Science

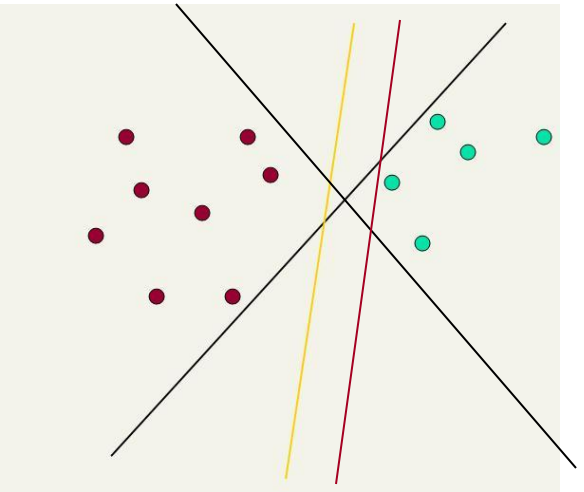
Support Vector Machines

(slides taken from https://web.stanford.edu/class/cs276/handouts/lecture14-SVMs.ppt)

Linear classifiers: Which Hyperplane?

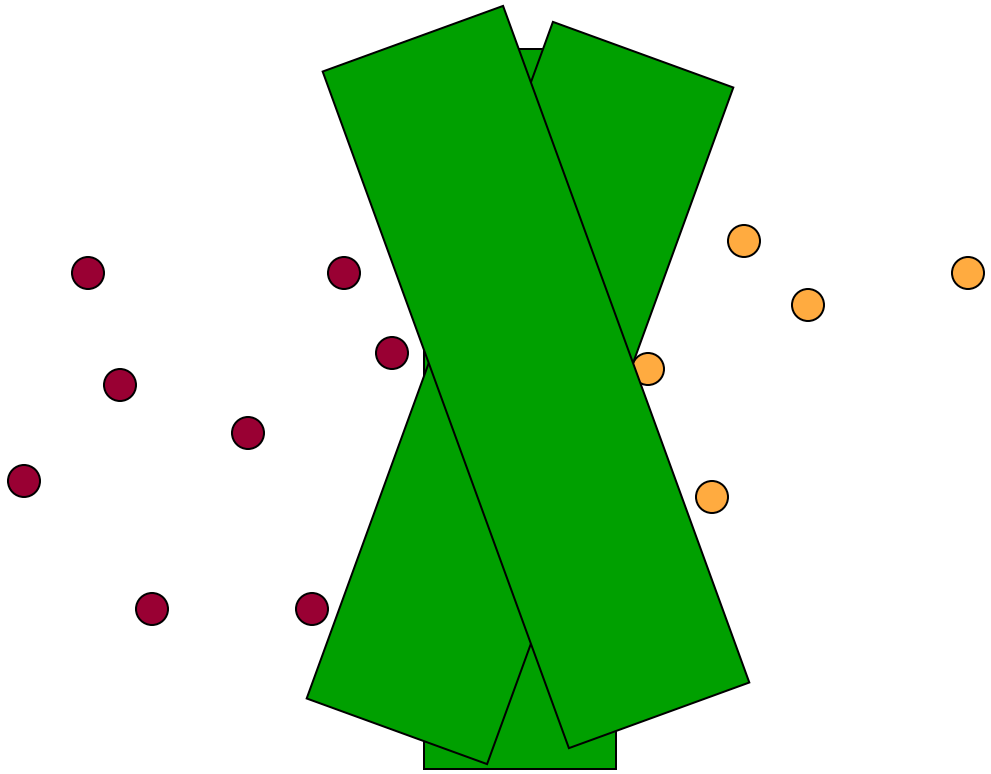
- Lots of possible solutions for *a, b, c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:
$ax + by - c = 0$



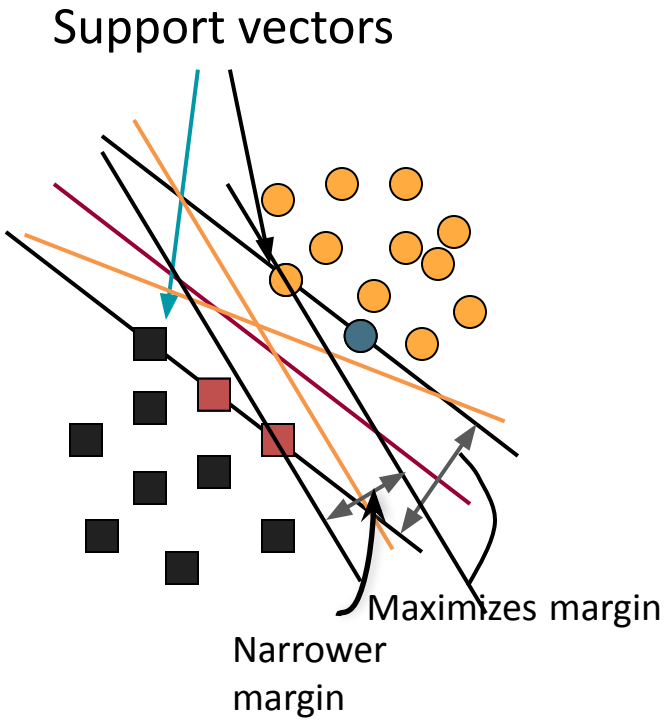
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



*but other discriminative methods often perform very similarly

Maximum Margin: Formalization

- **w**: decision hyperplane normal vector
- $\mathbf{x}_i$: data point *i*
- y_i: class of data point *i* (+1 or -1) NB: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $y_i(\mathbf{w}^T\mathbf{x}_i + b)$
  - But note that we can increase this margin simply by scaling **w**, **b**….
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

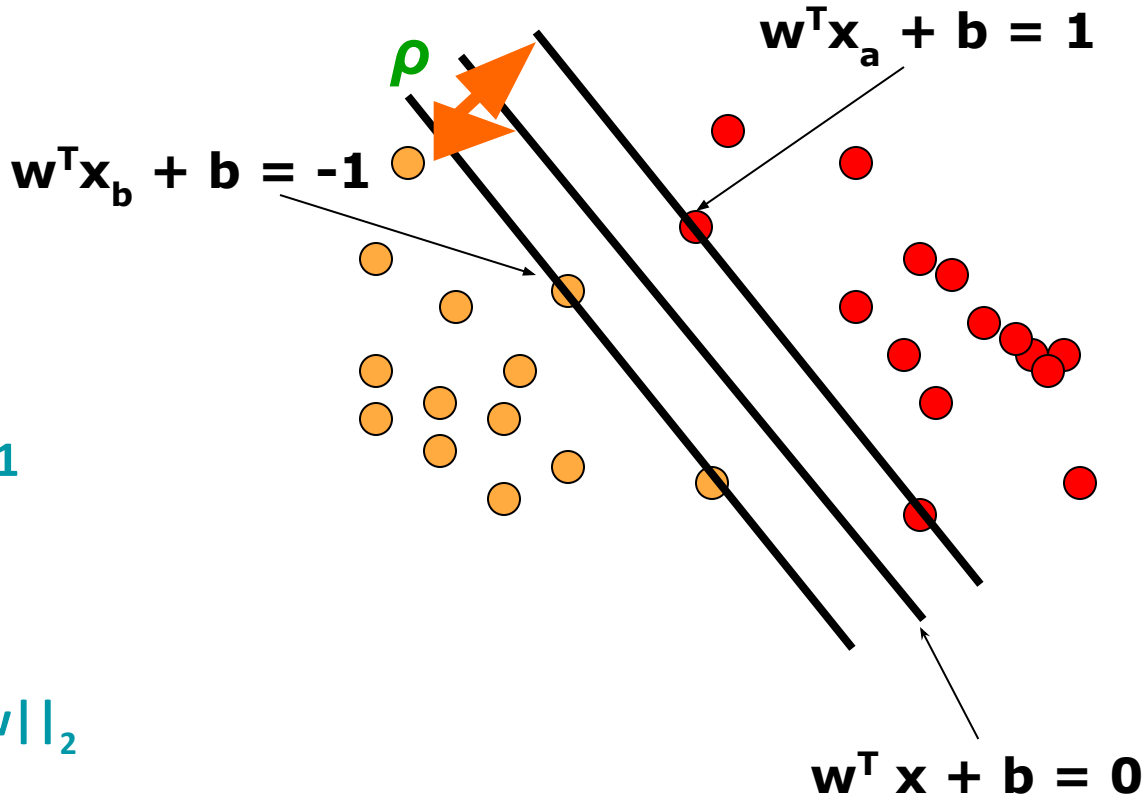
$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

- **Hyperplane**
  $\mathbf{w}^T \mathbf{x} + b = 0$
- **Extra scale constraint**:
  $\min_{i=1,\ldots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$
- This implies:
  $w^T(x_a - x_b) = 2$
  $\rho = ||x_a - x_b||_2 = 2/||w||_2$



Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w^T x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min $||\mathbf{w}||$ = max $1/||\mathbf{w}||$):

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w^T x_i} + b) \geq 1$

Functional Margin

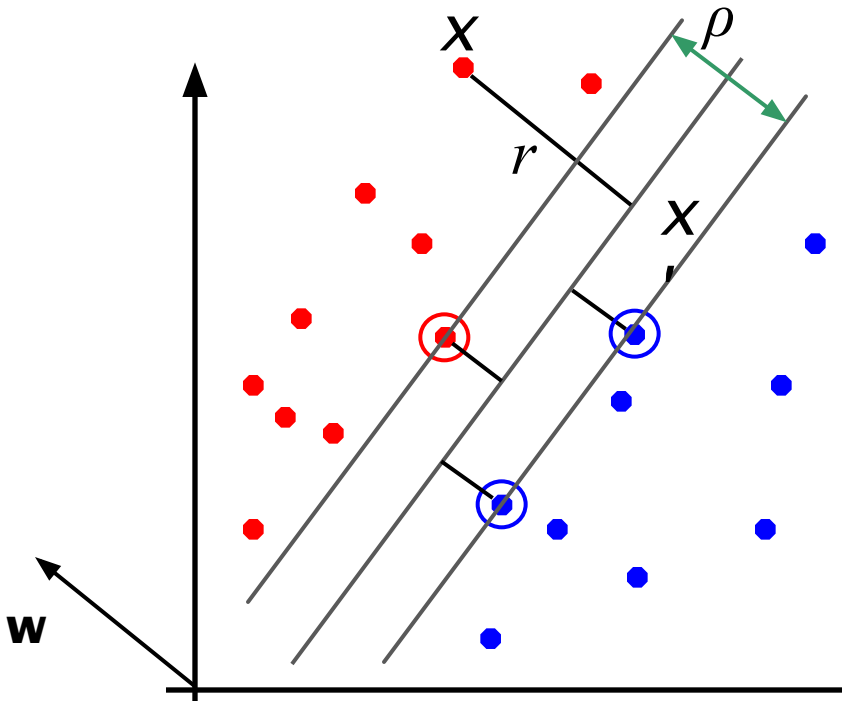
(https://stackoverflow.com/questions/20058036/svm-what-is-a-functional-margin)

The functional margin represents the correctness and confidence of the prediction if the magnitude of the vector(w^T) orthogonal to the hyperplane has a constant value all the time.

- By correctness, the functional margin should always be positive, since if wx + b is negative, then y is -1 and if wx + b is positive, y is 1. If the functional margin is negative then the sample should be divided into the wrong group.
- By confidence, the functional margin can change due to two reasons: 1) the sample(y_i and x_i) changes or 2) the vector(w^T) orthogonal to the hyperplane is scaled (by scaling w and b). If the vector(w^T) orthogonal to the hyperplane remains the same all the time, no matter how large its magnitude is, we can determine how confident the point is grouped into the right side. The larger that functional margin, the more confident we can say the point is classified correctly.

Geometric Margin

- Distance from example to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.



Derivation of finding *r:*
Dotted line **x'**–**x** is perpendicular to decision boundary so parallel to **w**.
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|.
x' = **x** – yr**w**/|**w**|.
x' satisfies $\mathbf{w}^T\mathbf{x}'$+b = 0.
So $\mathbf{w}^T$(**x** –yr**w**/|**w**|) + b = 0
Recall that |**w**| = sqrt($\mathbf{w}^T\mathbf{w}$).
So $\mathbf{w}^T$**x** –yr|**w**| + b = 0
So, solving for r gives:
r = y($\mathbf{w}^T$**x** + b)/|**w**|

Geometric Margin

(https://stackoverflow.com/questions/20058036/svm-what-is-a-functional-margin)

A *geometric margin* is simply the signed euclidean distance between a certain x (data point) to the hyperlane. Keep in mind that euclidean distance is always positive while the geometric margin can be positive or negative.

The geometric margin is just a scaled version of the functional margin. The functional margin is an unscaled version of the geometric margin.

The magnitude ||w||, also called the norm, is the length of the segment between the origin and the tip of the vector.

A vector is an object that has both a magnitude and a direction.

The direction of the vector is defined by the angle θ with respect to the horizontal axis, and with the angle α with respect to the vertical axis.

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q(\alpha)} = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

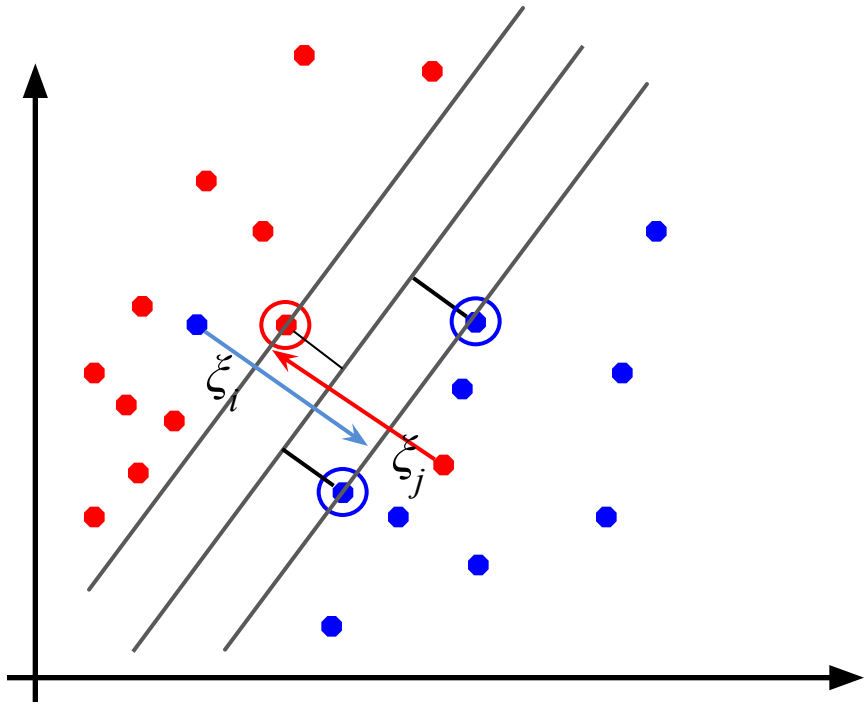
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting
  - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$
$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x}_k$ where $k = \underset{k'}{\text{argmax}}\ \alpha_{k'}$

w is not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

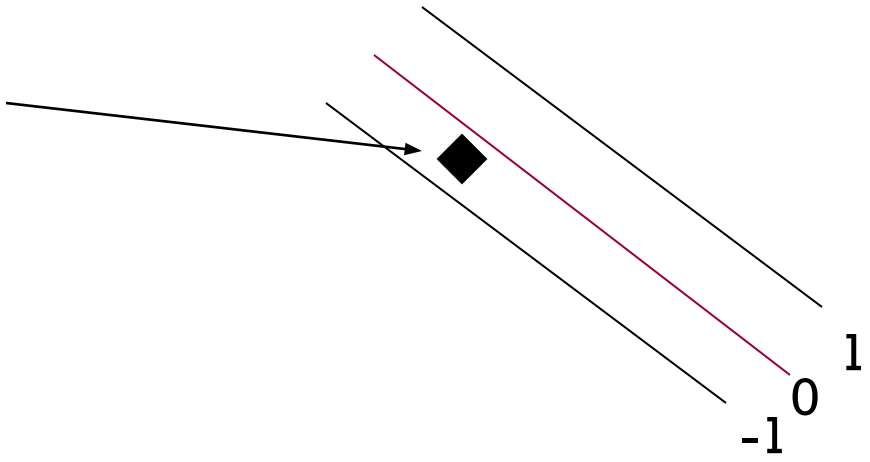
Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$
    - Decide class based on whether < or > 0
  - Can set confidence threshold *t*.

Score > *t*: yes

Score < -*t*: no

Else: don't know



Linear SVMs: Summary

- The classifier is a *separating hyperplane.*
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

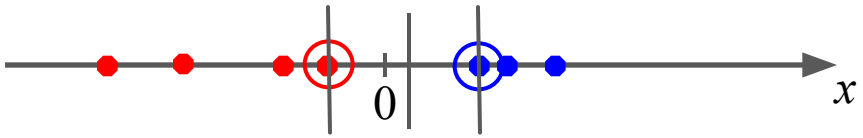
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

Read More

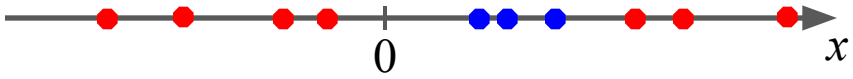
- SVM Tutorial, https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-2/
- Notes for Reviewing SVM: https://regularization.medium.com/notes-for-reviewing-svm-5e60797b34b
- An Idiot's guide to Support vector machines (SVMs), https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Non-linear SVMs

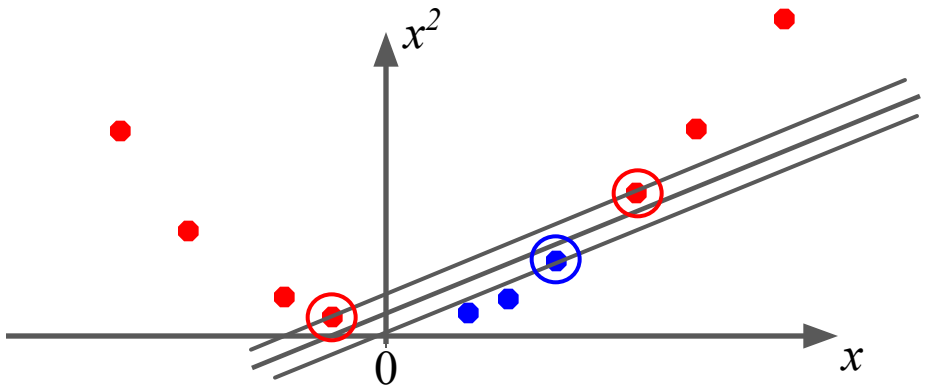
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

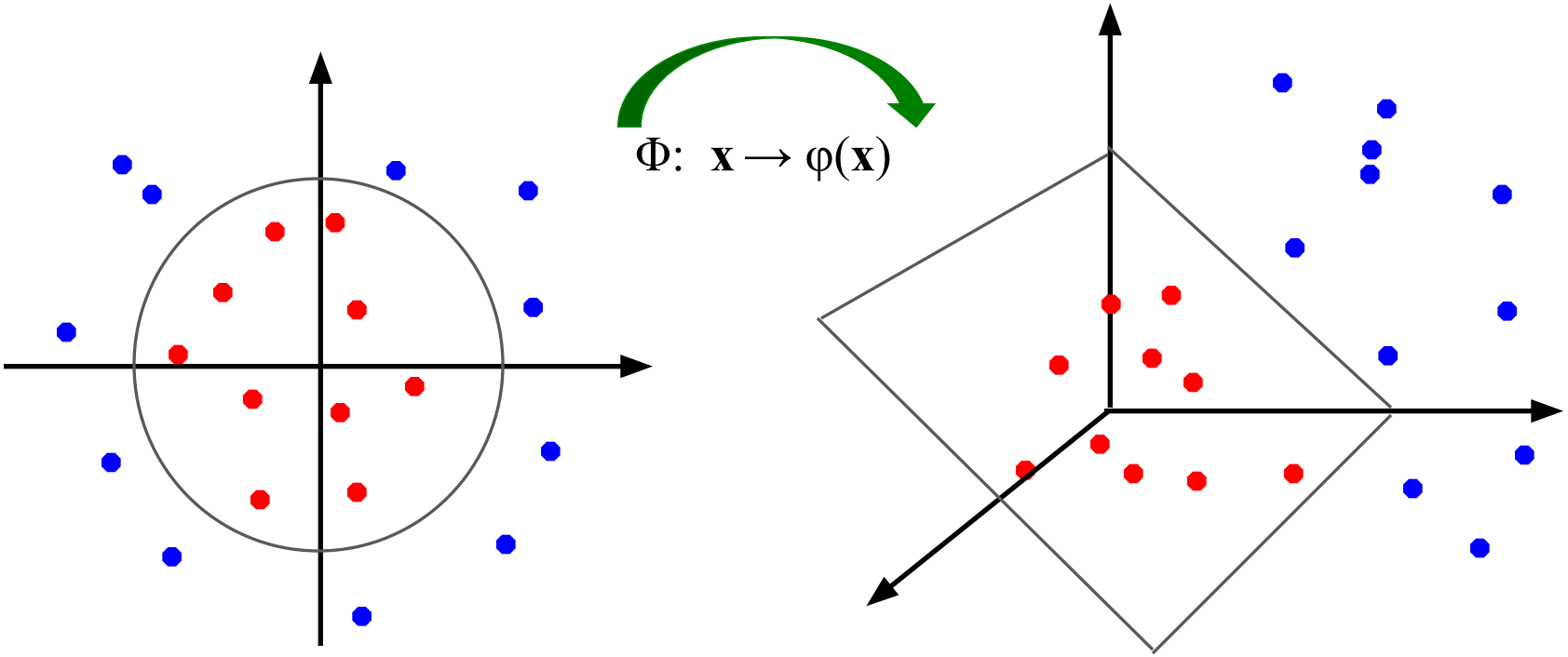


- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

  $= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ where $\varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial $K(x,z) = (1+x^Tz)^d$
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

- Haven't been very useful in text classification

Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Nearest Neighbor Classification: NN-Rule

- The *nearest neighbor classification rule* assigns a label to an element x as the class $l(x) \in \{-1,+1\}$ where $l(x)$ is the label of the closest labeled point $\mathrm{NN}(x)$ in the training set:

$l(x) = l(\mathrm{NN}_Z(x))$.

- That is, we have $l(x) = y_e$ for $e = arg\min_{i=1}^{N} D(x, x_i)$ where $D(\cdot, \cdot)$ is an appropriate distance function (usually taken as the Euclidean distance).
- If there exist several points yielding the same minimal distance, we choose arbitrarily one of those points.
- We classify t = |Q| new unlabeled observations of the testing set by answering t nearest neighbor (NN) queries in X.
- It is crucial to answer these queries as fast as possible, at least in sub-linear time so that we can beat the naive algorithm that scans all the points of X.

- Notice that a distance or a monotonically increasing function of this distance, like the square function, does not change the relative ordering of points according to a query point q.
- There exists many data-structures to answer such NN queries but as the dimension d increases, it becomes (provably) difficult to beat significantly the naive algorithm.
- This is the phenomenon that bears the name of the curse of dimensionality!
- Historically, this curse of dimensionality was introduced by Bellman, the founder of the dynamic programming paradigm.

k-Nearest Neighbor Classification (k-NN)

- Unlike all the previous learning methods, k-NN does not build model from the training data.
- To classify a test instance *d*, define *k*-neighborhood *P* as *k* nearest neighbors of *d*
- Count number *n* of training instances in *P* that belong to class c_j
- Estimate $\Pr(c_j|d)$ as n/k
- No training is needed. Classification time is linear in training set size for each test case.

k-NN Algorithm

Algorithm kNN(D, d, k)

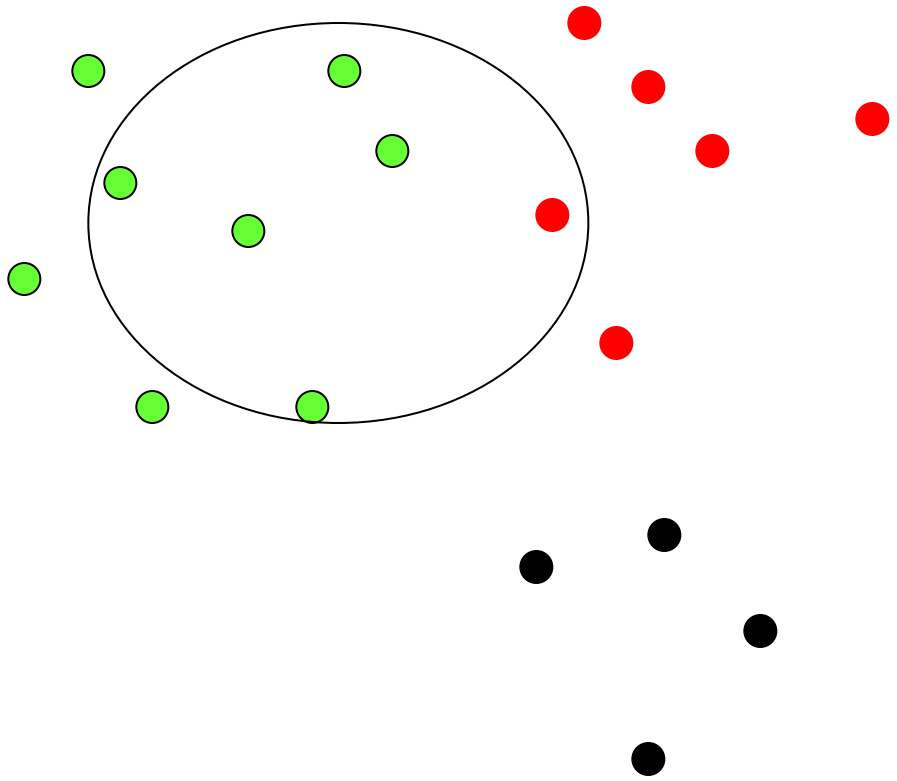
1 Compute the distance between d and every example in D;

2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);

3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

Example: k=6 (6NN)



Government

Science

Arts

A new point
Pr(science|)?

Discussions

- k-NN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of k-NN can be quite strong and in many cases as accurate as those elaborated methods.
- k-NN is slow at the classification time
- k-NN does not produce an understandable model

Enhancing the NN-Rule with the k-NN Rule by Voting

- In order for the classifier to be resilient to noisy data-sets (say, imprecise input and outliers), we may class a new observation *q* by choosing among the first *k* nearest neighbors of *X*, the dominant class.
- For binary classification, it is useful to choose an odd value for *k* in order to avoid vote ties. In practice, increasing *k* allows one to be tolerant to outliers in the data-set, but the drawback is that the decision boundary becomes more fuzzy as *k* increases.
- Many techniques or rules of thumbs to choose the most appropriate value of *k* for this *k*-NN rule.
- For example, the *cross- validation* method that uses part of the training set to train, and the remaining part to test.
- Note that the *k*-NN voting rule generalizes the NN-rule (by choosing *k* = 1, NN = 1-NN). When dealing with multi-classes, the rule consists in choosing the dominant class inside the *k*-NNs.

Neural Networks

- Neural networks are composed of multiple layers, each layer is a single most data structure of the network.
- It takes input of one or more tensors and outputs one or more tensor (tensor being a value or a vector).
- These layers have a weight based loss function which is feedback signal used for learning.
- Each neural net has an optimizer that determines how learning proceeds.

ARTIFICIAL NEURAL NETWORKS

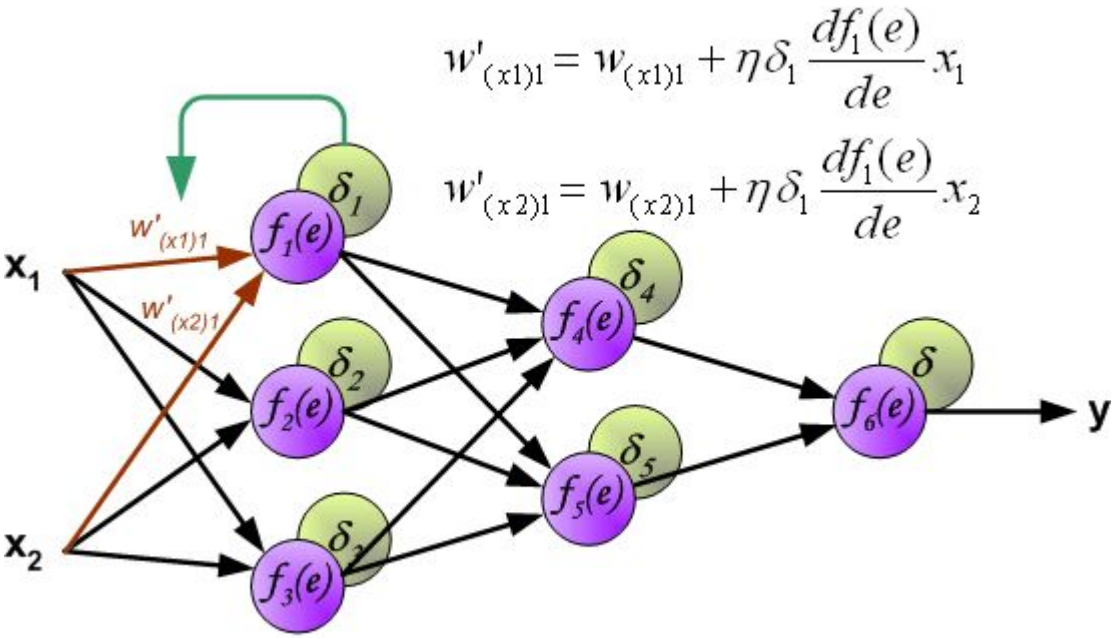
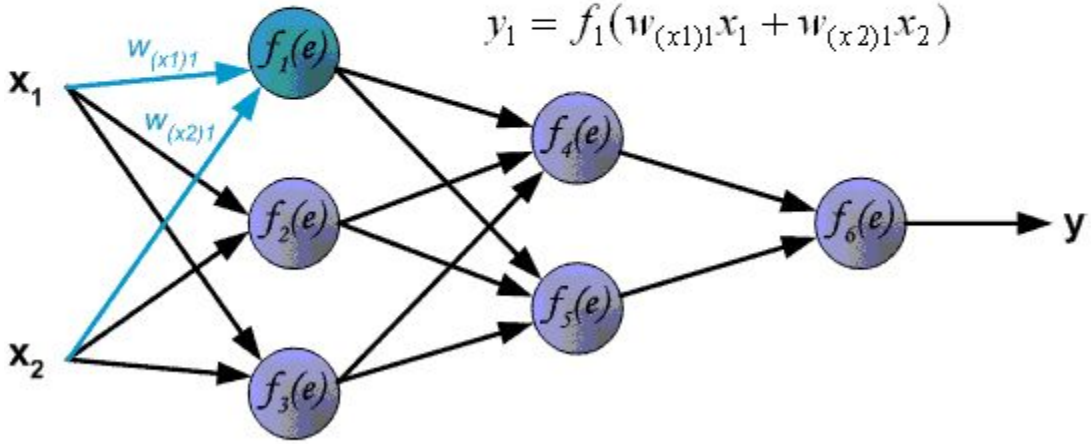


Figure 3. Forward & Backpropagation. Adapted from "Danifree's blog," by daniel-D. Retrieved December 11, 2018, from http://www.cnblogs.com/daniel-D/archive/2013/06/03/3116278.html/. Copyright by daniel-D.

ARTIFICIAL NEURAL NETWORKS

FP



RECURRENT NEURAL NETWORKS

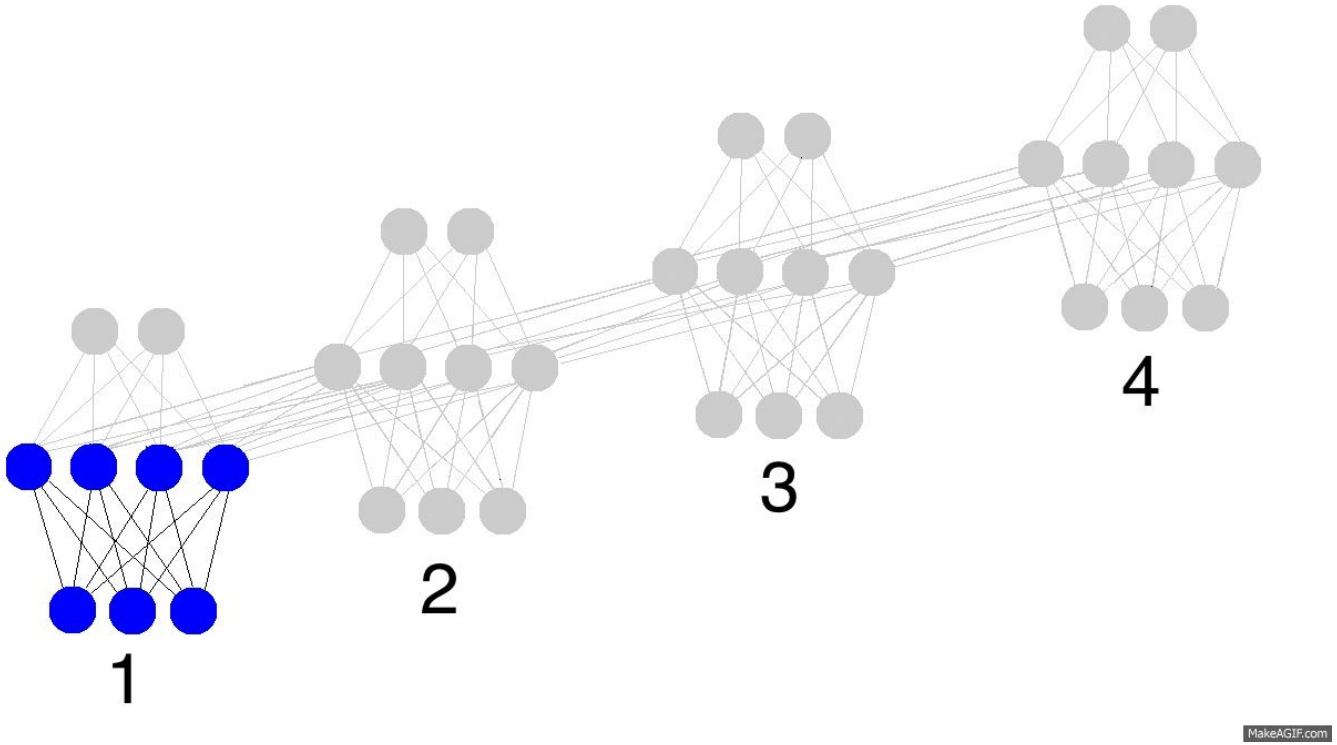


Figure 4. Recurrent Neural Networks. Adapted from "Anyone can learn to code an LSTM-RNN in Python (Part-1: RNN)," by Andrew Trask. Retrieved December 11, 2018, from https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/. Copyright 2018 by i am trask.

LSTM - LONG SHORT TERM MEMORY

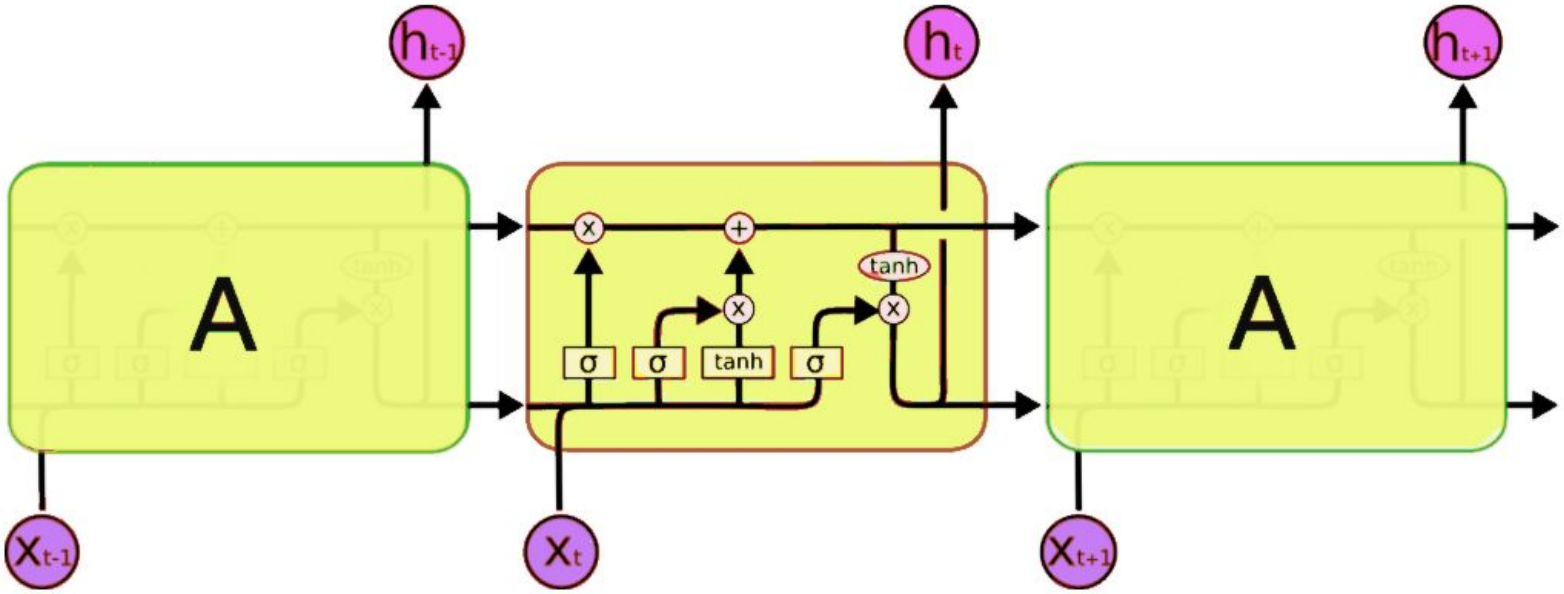


Figure 5. Long short term memory. Adapted from "Essentials of Deep Learning: Introduction to Long Short Term Memory," by Pranjal Srivastava. Retrieved December 11, 2018, from https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/. Copyright 2013-2018 by Analytics Vidhya.

CONVOLUTIONAL NEURAL NETWORKS

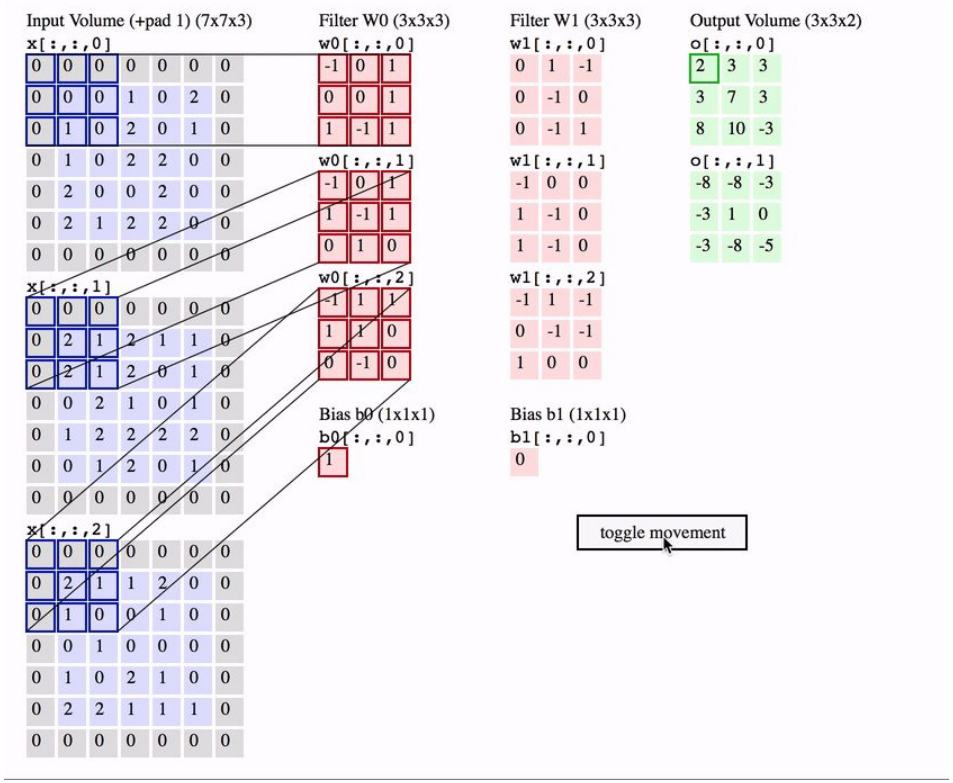
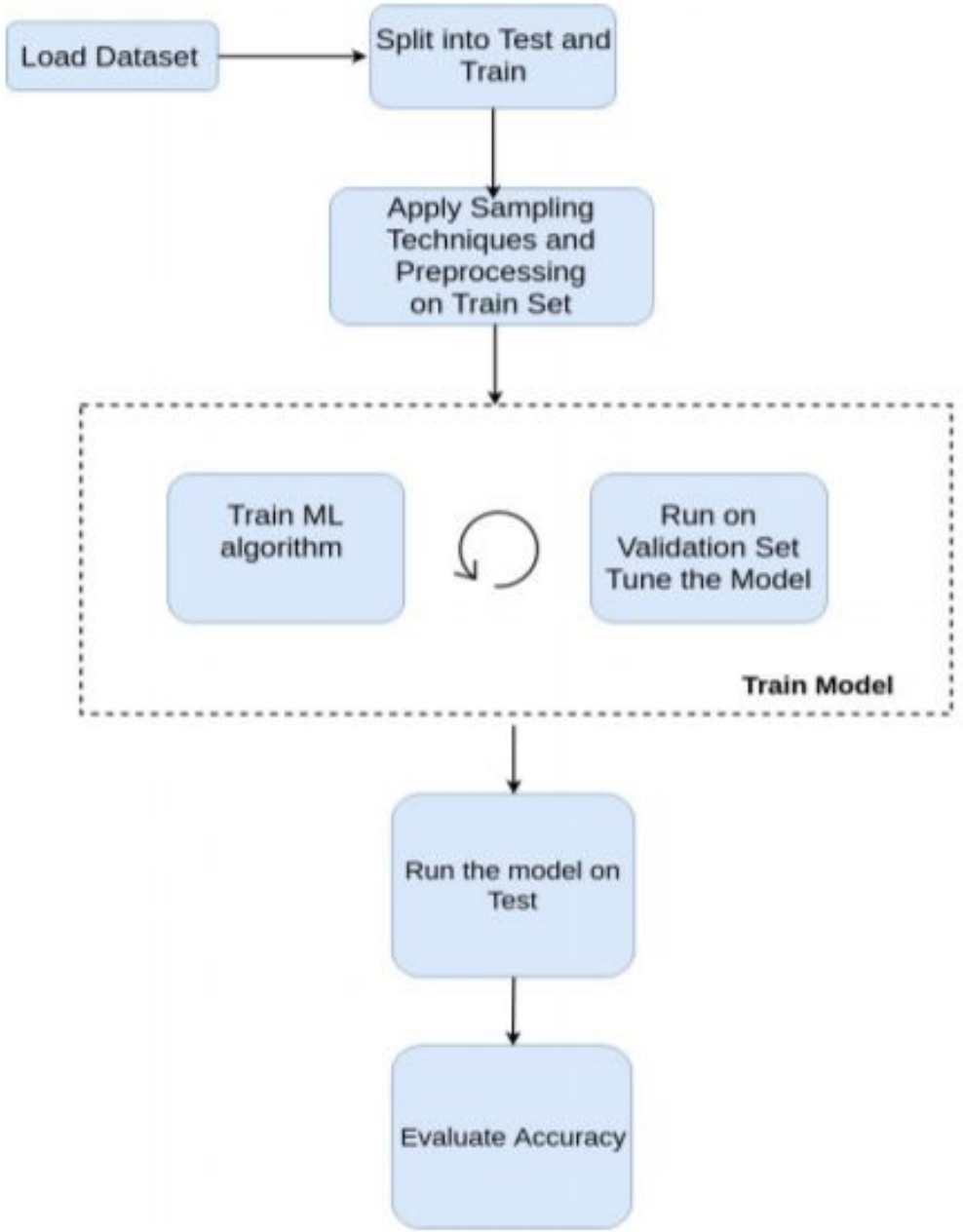


Figure 6. Convolutional Neural Networks from "An intuitive guide to Convolutional Neural Networks," by Daphne Cornelisse. Retrieved December 11, 2018, from https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.

Data Pre-processing

- First obtain the dataset and load into a csv or json file format of preference.
- Then decide what columns to use and what columns to ignore
- Dataset must be split into Train and Test with a 70-30 or 80-20 distribution.
- The objective of the model is to learn and apply generalized solutions for a given problem.
- Another important rule for a good machine learning is to split the Train data into Train and Validate.
    - Train on the training data and evaluate model on the validation data.
    - Once model is ready, test it one final time on the test data.


Load Dataset
Split into Test and Train
Apply Sampling Techniques and Preprocessing on Train Set
Train ML algorithm
Run on Validation Set Tune the Model
Train Model
Run the model on Test
Evaluate Accuracy

Obtaining your free dataset

- Check this GDoc created by me in Summer 2021: https://docs.google.com/document/d/1bSFxrX0_PdugEuv6s7GWpS1M_7yxmUuVy87VhSyR1po/edit?usp=sharing
- Demo on how to search on Kaggle and Data.gov

Evaluating the ML Model

- The test data will be used to evaluate if model has learnt correctly.
- Model’s performance is measured and its accuracy evaluated.
- The performance measures adapted in this model are Area under ROC, Precision, Recall, and Average Precision.