COMP 642 — Machine Learning Jan 20, 2021

Lecture 4

Lecturer: A. Shrivastava Scribe By: Sinead Tracey, Kyle Rewick, Dan Eakin, Zach Lashaway

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Review of Jan-18 Lecture

Logistics

- This is the last online class. In-person instruction starts Jan-25
- TAs were announced and introdcued:
  - Keming Zhang (Office Hours every Wednesday 11:00am-12:00pm and Thursday 10:30-11:30am in McMurtry College Commons)
  - Minghao Yan (Office Hours from 3:30pm to 4:30pm every Tuesday at DCH3060)
  - Zhaozhuo (Office Hours from 1:30 am to 2:30 am every Monday at DCH3135)

Review of Jan-18 lecture:
Regression classifier and loss function:
Linear function follows the format $ax + b$:

$$F_w(x) = \omega^T x + b = \sum_{i=1}^{d} \omega_i x_i + b \tag{1}$$

Loss function (least square):

$$L(x) = (F_w(x) - y)^2 \tag{2}$$

General Notes on classification versus regression:

- *Regression* used to predict a number, *Classifiction* used to predict a class
- Can't use least square as the loss function for classification. This is illustrated by the following example:

A	O	P	G
0	1	2	3

Given sample x_i and correct classification $y_i = G$:

- Least square loss function would suggest that $y_i = P(L(x) = (4-3)^2 = 1)$ is *more* correct than predicting $y_i = A(L(x) = (4-0)^2 = 16)$
- This is not correct (one class is not *closer* to another class in this example)

Therefore, we need to use percetptron (aka linear binary classifier):

- "Perception is usually used to classify data into 2 parts:"[1]

- Binary classification:

$$y = \begin{cases} 0/-1 & \text{class 1} \\ 1 & \text{class 2} \end{cases} \tag{3}$$

- Classifier:

$$F_w(x) = sign(\omega^T x + b) \tag{4}$$

- Loss function (indicator function adds 1 for every incorrect classification):

$$L = \frac{1}{n} \sum i = 1^n I_{F_w(x_i) \neq y_i} \tag{5}$$

References

1. What the Hell is Perceptron?: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

2 Large Margin SVMs as regularized hinge loss

Recall, binary linear classifiers aim to differentiate observations into 2 categories. The classification is accomplished by way of a decision boundary dividing the data space into 2 regions, where $f(w) < 0$ and $f(w) > 0$. The classification boundary, also known as the discriminate, corresponds to $f(w) = 0$. In 2D, the boundary is a line whereas it is a plane or hyper-plane in higher dimensions.

Linear classifiers suffer from what is called the "Identifiability Problem" whereby multiple models that lead to 0 training errors exist. Furthermore, the identification of the best model among the 0 training error options is not possible.

Support Vector Machines

Support Vector Machines (SVMs) are supervised learning models used for binary classification problems. They have been successful in classifying observations as either positive or negative instances by exploiting the concept of regularized hinge loss. Hinge loss is a cost function that employs a margin from the classification boundary as a measure of confidence. Observations residing farther from the classification boundary are considered higher confidence classifications.

The equation for an SVM model is

$$f_w(w) = w^T x + w_0 \tag{6}$$

The outputs are

$$\hat{y} = sign(f_w(w)) \in \{-1, +1\} \tag{7}$$

Accordingly, the classification boundary whose margin, defined as the perpendicular distance in both the positive and negative directions from the classification boundary prior to encountering a point in the training data set, is maximal is selected. The margin is equidistant from $f(w) = 0$ in the positive and negative direction and therefore, parallel to $f(w) = 0$. This method enables the selection of an optimal model by seeking the boundary with the largest margin. The margin distance can be defined by constant k as follows

$$w^T x + w_0 \geq k \quad \text{when} \quad y_i = +1 \tag{8}$$

$$w^T x + w_0 \leq -k \quad \text{when} \quad y_i = -1 \tag{9}$$

The concept in 2D is depicted in Figure 1.

The optimization of an SVM is given by

$$min \quad C \sum_{i=1} max(0, 1 - y_i[w^T x_i + w_0]) + \lambda \parallel w \parallel_2^2 \tag{10}$$

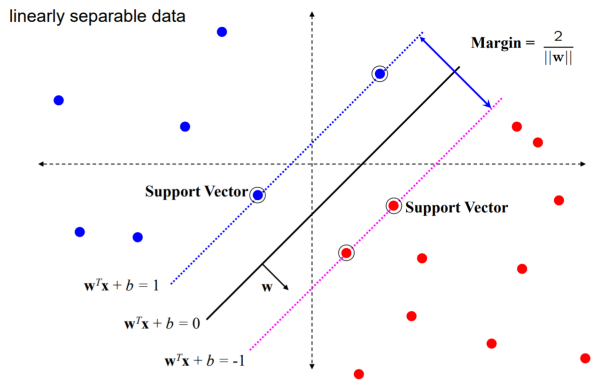

Figure 1: SVM Margin

The first term is the hinge loss and represents the classification errors. The second term is the regularization term. The hinge loss is zero if the data point is classified correctly and increases when data points reside close to the classification boundary inside the margin. A data point residing on the margin will have a hinge loss of 0 (Figure 2).

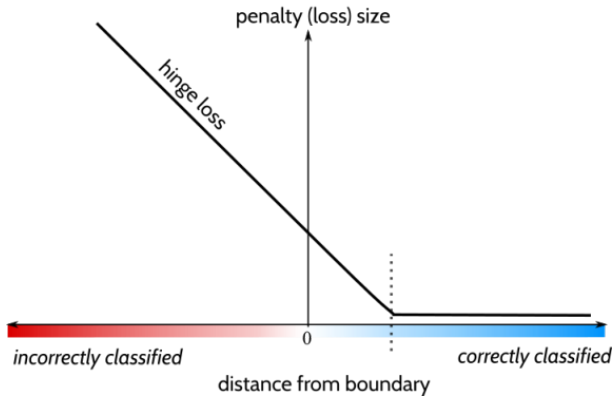

Figure 2: Hinge Loss

The parameter C is the regularization parameter. The regularization parameter becomes relevant when data sets are overlapping, i.e. not definitively separable by a classification boundary. This is known as a soft margin SVM. Large regularization parameter values correspond to smaller margins leading to an increased number of correct classifications. Conversely, large margins are associated with smaller parameter values and more instances of misclassification.

The distance, $d_w(x_i)$, of a data point x_i from the line $f(w) = w^T x_i + w_0$ is

$$d_w(x_i) = \frac{w^T x_i + w_0}{\parallel w \parallel_2} \tag{11}$$

Zero loss occurs beyond the margin, specifically when,

$$max(0, 1 - y_i[w^T x_i + w_0]) = 0 \tag{12}$$

which then implies

$$y_i[w^T x_i + w_0] = 1 \tag{13}$$

when a data point is on the margin.

Then, by equation 11,

$$d_w(x_i) = \frac{1}{y_i \parallel w \parallel_2} = \frac{1}{\parallel w \parallel_2} \tag{14}$$

is the margin distance in either the positive or negative direction. The total margin distance is

$$d_w(x) = \frac{2}{\parallel w \parallel_2} \tag{15}$$

As a result, minimizing the loss function is equivalent to maximizing the margin because the training errors are fewest when the margin distance is maximal. Mathematically, the loss function is minimized when the hinge loss term in equation 5 is 0 and the L2 Norm is minimized

$$min \parallel w \parallel_2^2 \tag{16}$$

Minimizing the L2 Norm (equation 11) is equivalent to maximizing the margin (equation 10).

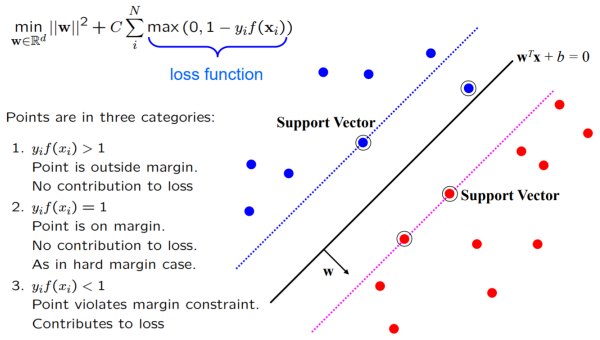

Figure 3: Hinge Loss

References

1. Support Vector Machines:
https://en.wikipedia.org/wiki/Support-vector-machine

2. Support Vector Machine (SVM) Theory:
https://www.commonlounge.com/discussion/a49bcd907bdc4824ae53483c060f0259

3. Understanding Hinge Loss and the SVM Cost Function:
https://programmathically.com/understanding-hinge-loss-and-the-svm-cost-function/

4. A definitive explanation to the Hinge Loss for Support Vector Machines:
https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1

3 Set up to Regularization

Summary of Margins

- **Margin**:

  $\forall x_i \; \exists \; D \;\; f(x_i) \geq \; k \; \forall x_i \; where \; y_i = 1 \;\; f(x_i) \leq \; -k \; \forall x_i \; where \; y_i = 0$

- Best practice is to choose a classifier with the maximum margin K
- The classifier that is most robust to perturbation (on the training data set) has the maximum margin which is a unique solution
- We want to maximize K (2k) $->$ margin, such that:

$$n \; constraints => \begin{cases} w^T x_i \; + \; b \; \geq \; k \; when \; y_i = 1, \; and \\ w^T x_i \; + \; b \; \leq \; -k \; when \; y_i = -1 \end{cases}$$

  - *If you know your margin is K, also want to maximize the number of constraints that are also being satisfied

  Translation:

  - For all positive data points, k is the distance one can go without touching them
  - For all negative data points, -k is the distance one can go without touching them

  Assumptions:

  - Constraint that we have at least k margin (k can be as small as 0 or as large as infinity)
  - Assuming separable data (0 training error on the training data)

  Issues:

  - If you artificially scale w, i.e. multiply $w^T x_i$*100, K will increase

- max K (2k) equation then becomes:

$$min||w||_2(norm) \;\; \sqrt{w_1^2 + w_2^2 + ...w_d^2}, \text{ such that: } \begin{cases} w^T x_i + b \; \geq \; 1, \forall (x_i, y_i) \; y = 1 \\ w^T x_i + b \; \leq \; -1, \forall (x_i, y_i) \; y = 0 \end{cases}$$

- You want a classifier that separates the data and minimizes the norm. More specifically, in machine learning you want to not only minimize the loss function, but also the norm of your classifier
- Minimizing the norm makes your classifier robust to perturbations, and leads to a unique classifier

  This is REGULARIZATION.

4 Issues with Linear Classifiers

Linear classification at first might appear simple given the simplicity of the algorithm.

$$f(x) = w^T x + b \tag{17}$$

However, the difficulty is in training the linear classifier, that is, in determining the parameters w and b based on the training set.

Perceptron networks have several limitations. First, the output values of a perceptron can take on only one of two values (0 or 1) because of the hard-limit transfer function. Second, perceptrons can only classify linearly separable sets of vectors.

SVM (support vector machine) has several limitations such as: SVM algorithm is not acceptable for large data sets, does not execute very well when the data set has more sound i.e. target classes are overlapping, in cases where the number of properties for each data point outstrips the number of training data specimens, SVM will under-perform.
As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.

5 XOR Problem

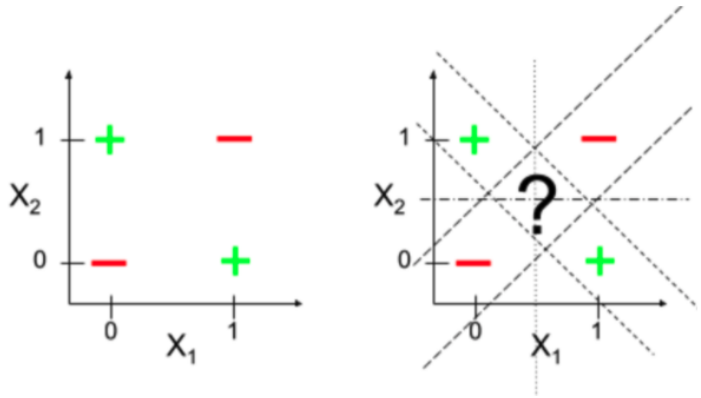

Figure 4: XOR Problem

Can never have a linear classifier that results in zero training error. However, in a higher dimension space can always have a solution.
For example consider an extended feature space such as:

$$\mathbf{w}^\top \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}}_{\psi(\mathbf{x})} > -w_0 \tag{18}$$

Instead of an input vector x, make use of a feature vector ψ, with three elements, x_1, x_2, and $x_1 x_2$

These non-linear features allow linear classifiers to solve non-linear classification problems. This is analogous to polynomial curve fitting as well as Support Vector Machines wherein we use kernels to lift the feature vector space.
3D Example:

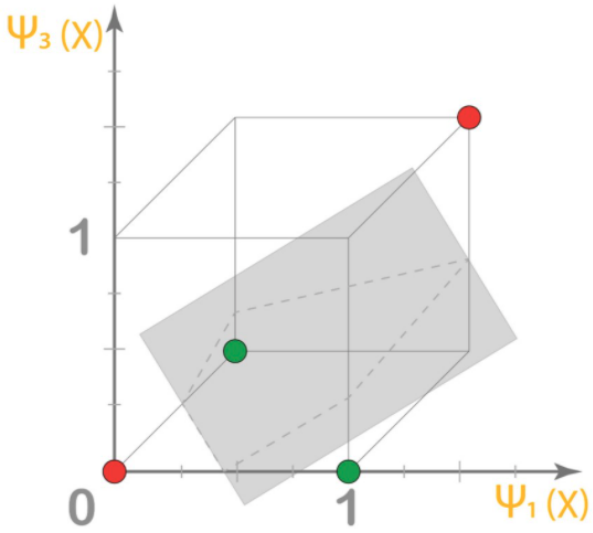

Figure 5: XOR Problem

Observe how the green points are below the plane and the red points are above the plane. This plane is nothing but the XOR operator's decision boundary.

So, by shifting our focus from a 2-dimensional visualization to a 3-dimensional one, we are able to classify the points generated by the XOR operator far more easily.

This exercise brings to light the importance of representing a problem correctly. If we represent the problem at hand in a more suitable way, many difficult scenarios become easy to solve as we saw in the case of the XOR problem. Let's understand this better.

References

1. https://nlp.stanford.edu/IR-book/html/htmledition/linear-versus-nonlinear-classifiers-1.html
2. https://www.mathworks.com/help/deeplearning/ug/perceptron-neural-networks.html;jsessionid=f207eed6d4
3. https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/ 4. https://www.tech-quantum.com/solving-xor-problem-using-neural-network-c/ 5. https://datahacker.rs/006-solving-the-xor-problem-using-neural-networks-with-pytorch/

6 Overfitting

What is overfitting?

- Overfitting occurs when a model is constraint to a training set and not able to perform well on unseen data (FigX, FigXc). Conversely, if our model is too simple, or has not been trained long enough.

- Overfitting can be caused by:

  - An overly complex model that that is predicting noise

  - The search domain for the model function is large and there is not enough data to constrain the search

  - If you have lots of data, you see then phenomena of Double Descent (see Section 1)

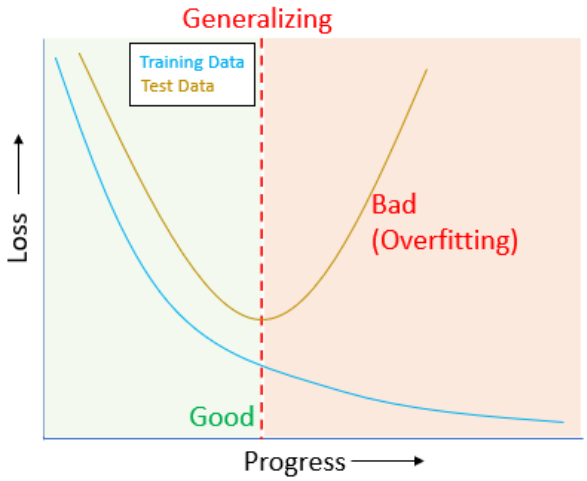

Figure 6: Schematic graph of model progress vs. loss. The model generalizes very well in the training data but cannot replicate results in Test Data

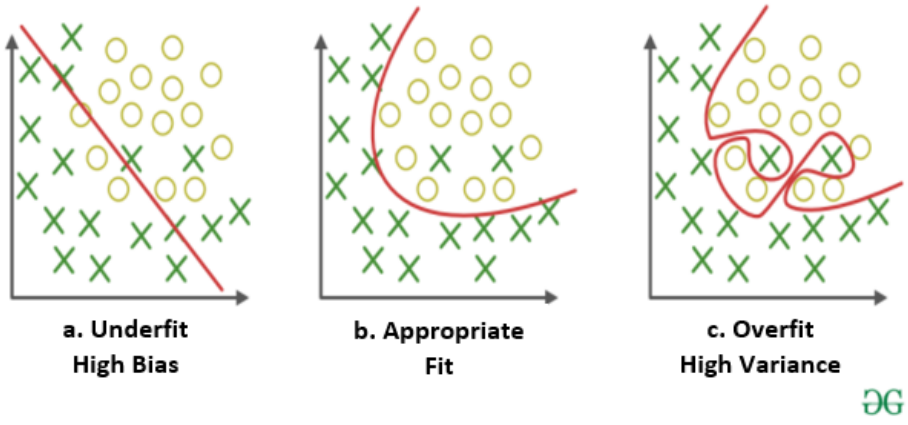

Figure 7: Schematic showing examples of underfitting, overfitting, and approriately fit date. Modified from GeeksforGeeks: Regularization in Machine Learning

7 Regularization and Model Constraint

What is Regularization?

- Regularization is introduced when you do not have a unique solution or you suspect that your model is overfitting. It does this by penalizing unnecessary complexity in our model.

- The four most common types of regularization are:
  - L1 (Lasso Regularization)
  - L2 (Ridge Regularization)
  - **More Data!
  - Dropout *not covered (see bibliography for more information)

How does Regularization work?

In general, for a linear classifier, we add a penalty term to the loss function in form the norm scaled by some value λ

- **L1 - Lasso Regularization**

$$L(f_W(x) = \frac{1}{n}\sum_{i=1}^{n}(F_w(x_i) - y_i) + \lambda * |w|_1$$

  - Gradients of parameters for L1 are independent of parameters, so some parameters can be set to 0, or effectively ignored. May use this type of regularization when there are useless features in your model.

- **L2- Ridge Regression**

$$L(f_W(x) = \frac{1}{n}\sum_{i=1}^{n}(F_w(x_i) - y_i)^2 + \lambda * |w|_2$$

  - Gradient of the loss function is linearly dependent, so parameters can never be 0. Every parameter has a minimal effect on the model.

Adding More Data: How do we add more constraint to out data?

Remember, overfitting is caused by lack of constraint in the search space of an optimal function. How can we fix this?

- Add more data - New features

- Augment current data

  - Create more data from the data you already have by applying transformations. This will make multiple versions of the dame feature that are different resulting in "more" data!

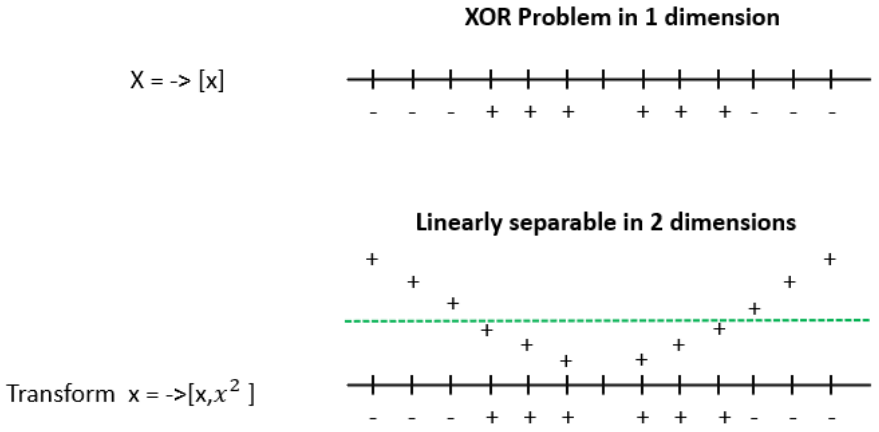

Figure 8: Schematic showing 1D XOR. If we apply a linear transformation to x, we can achieve separability in 2D

  - List of popular transformations: https://docs.fast.ai/vision.augment

- VC Dimension of a classifier

  - Measure of how complex a model is

  - Intuition is that if you give more parameters to a model, you can fit any training data. Also see Section 1 "Double Descent".

References

1. Regularization Dropout In Deep Learning: https://towardsdatascience.com/regularization-dropout-in-deep-learning-5198c2bf6107
2. List of popular transformations: https://docs.fast.ai/vision.augment
3. Regularization in Machine Learning: https://www.geeksforgeeks.org/regularization-in-machine-learning/
4. Regularization. What, Why, When, and How? https://towardsdatascience.com/regularization-what-why-when-and-how-d4a329b6b27f

8 Overparameterization: The phenomena of Double Descent [2]

- Classical machine learning (ML) models assumed that "larger models are worse"
- This is illustrated by the following image (from *Double Descent* - URL in sources). The x-axis describes model complexity:

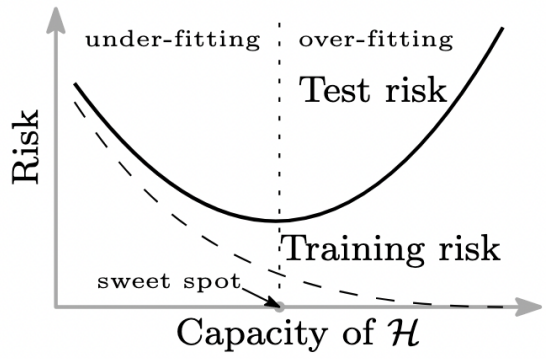

Figure 9: Classical ML

- As we increase capacity / complexity: we *overfit* the training data but does not fit the *test* risk well past a specific complexity
- Classical ML taught that models should be developed at the "sweet spot" (see image) to minimize test risk and training risk
- In contrast, more modern ML algorithms tend to display the following trends:

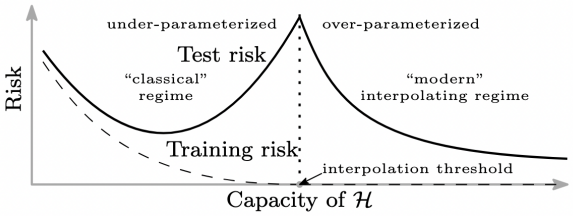

Figure 10: Double Descent

- Note that this figure matches the classical image to a point called the *interpolation threshold.* However, beyond that point the risk decreases for the test dataset.
  - This implies that *simpler may NOT be better for modern ML algorithms (ie nueural networks)*
  - More complex models with a larger number of parameter may in fact be better for modern machine learning models

References

1. Double Descent: https://medium.com/mlearning-ai/double-descent-8f92dfdc442f