COMP 642 Machine Learning Jan 24, 2023

Linear Classifiers, Deep Learning: Logistic Regression with Sequential Non-Linear Feature Extraction catered to minimize the loss

Lecturer: A. Shrivastava
Scribe By: Huafeng Liu (HL9), Ben Dowdell (BD52), Diyaz Yespayev (DY36), Ruslan Kharko (RK80)

Motivation

Linear Regression:

Given the function:

$$f(x_i) = \frac{w^T x_i}{\|w\| \|x_i\|}, f(x_i) \in [-1\ 1]$$

Its loss function is:

$$L_w = \frac{1}{n} \sum_{i=1}^{n} dist\left[\frac{w^T x_i}{\|w\| \|x_i\|}, y_i\right]$$

We've previously studied how the linear model $w^T x_i$ is used for data when the label y_i is in the set of real numbers. How can the linear model be used when the label is a *category*?

Data: The format of the label determines the type of problem we are trying to solve.

$\{x_i, y_i\}_{i=1}^{n}$ $x_i \in R^d$ (Vector)

$y_i \in R$	Regression
$y_i \in \begin{cases} -1 \\ 1 \end{cases}$	Binary Classification
$y_i = \{Cat, Dog, Cow, etc.\}$ or $\{0, 1, 2, 3, \ldots\}$	Multi-Class Classification

Linear Classifiers:

There are three types of linear classifiers: SVMs, Perceptron and Logistic Regression.

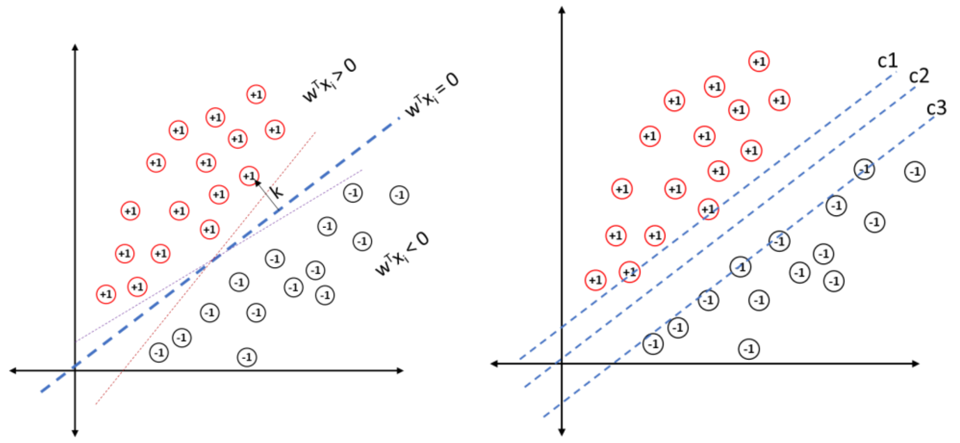
1. Support Vector Machines (SVMs):

   We can turn the linear model $w^T x_i$ into a linear classifier by taking its sign. This allows us to solve binary classification problems where $y_i\{-1, 1\}$.

$$f(x_i) = sign(w^T x_i), f(x_i) \begin{cases} 1, w^T x_i > 0 \\ 0, w^T x_i = 0 \\ -1, w^T x_i < 0 \end{cases}$$

$$L_w = \frac{1}{n} \sum_{i=1}^{n} dist[sign(w^T x_i), y_i]$$

   Note that there could be many solutions (infinite number of lines that separates points of different labels) for the decision boundary.



   The best classifier is picked by imposing an additional requirement that the classifier must be robust to perturbations in the data. In other words, if a data point is moved slightly, its predicted label should not change. In the image above, classifier c2 is the optimal classifier whereas c1 and c3 will both result in drastic changes in predicted labels when the data point is perturbed about the classifier's decision boundary.

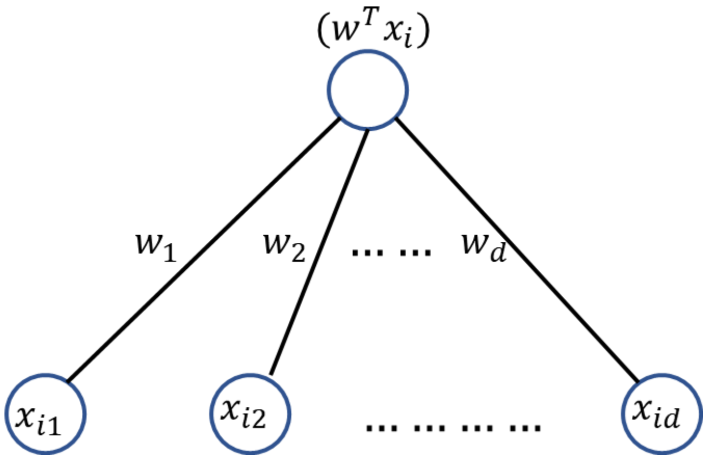
   We quantify the distance of each data point from the decision boundary as k and write $y_i(w^T x_i) \geq k$. This illustrates that the optimal linear classifier is the one that **maximizes** k. As such, SVM can be thought of as *hinge loss* with an L2 normalizer:

$$min \ \frac{1}{2}||w||_2 \ + \ \sum_{i=1}^{n} max[0, \ 1 \ - \ y_i(w^T x_i)]$$

Additionally, imposing regularization (e.g. maximize error margin) could narrow down the number of solutions to 1.

2. Perceptron:

What is a perceptron? A perceptron is a visual representation of a linear classifier (although they can be used in regression tasks, as well). In a perceptron, the outputs are linear combinations of inputs (x) with corresponding weights (w) plus a non-linear activation function on top of it.



Perceptrons introduce non-linearity through the activation function to the linear classifier $w^T x_i$. Examples of activation functions include:

Activation functions

$$\text{Sign}(w^T x_i) = \begin{cases} 1, & if \ w^T x_i < 0 \\ -1, & if \ w^T x_i > 0 \end{cases}$$

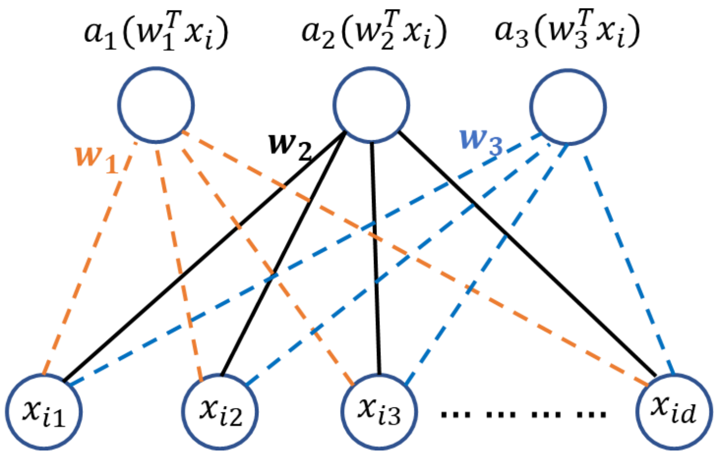
$$\text{Sigmoid}(w^T x_i) = \frac{1}{1+e^{w^T x_i}}$$

$$\text{ReLU}(w^T x_i) = \begin{cases} 0, & if \ w^T x_i < 0 \\ w^T x_i, & if \ w^T x_i > 0 \end{cases}$$

3. Logistic Regression:

Say that we have $y_i = \{1, 2, 3\}$. SVM works nicely for binary classification, but when we have more than one class, an issue arises. Logistic Regression can be formulated as given input data x_i and output label y_i (in categories, e.g.1,2,3,…), find the function that predicts y_i with high accuracy. Despite the name, it is a classification problem rather than a regression problem.

If we let each class in y_i be represented by a perceptron and impose that each perceptron's value is the probability that an observation x_i belongs to its class, we can get around SVM's binary classification limitation.



This requirement that each perceptron is the probability of a given data point belonging to its class is fulfilled via the *softmax* activation function:

Softmax activation: $$a_j = \frac{e^{w_j^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i} + e^{w_3^T x_i}}, j = 1,2,3$$

Where $a_1(w_1^T x_i)$, $a_2(w_2^T x_i)$, $a_3(w_3^T x_i)$ are probabilities of x_i being in each class and $a_1(w_1^T x_i) + a_2(w_2^T x_i) + a_3(w_3^T x_i) = 1$

Regularization, Generalization, Overfitting

Generalization refers to a model's ability to predict new, unseen data accurately. The goal of machine learning is to train a model that can generalize well to new examples rather than memorizing the training data. When a model generalizes well, it means that it has learned the underlying patterns in the data rather than just the specific examples it was trained on. A model that generalizes well will perform well on new data, whereas a model that overfits the training data will perform poorly on new data. Techniques like regularization, which help to prevent overfitting, also improve a model's ability to generalize.

<u>Overfitting</u> is a phenomenon that occurs in machine learning when a model is too complex and performs well on the training data but poorly on new, unseen data.

Overfitting can be caused by an overly complex (many, many parameters) model that predicts noise. If you have lots of data and increase the number of model parameters beyond the number of observations in the data, you see the phenomena of Double Descent.

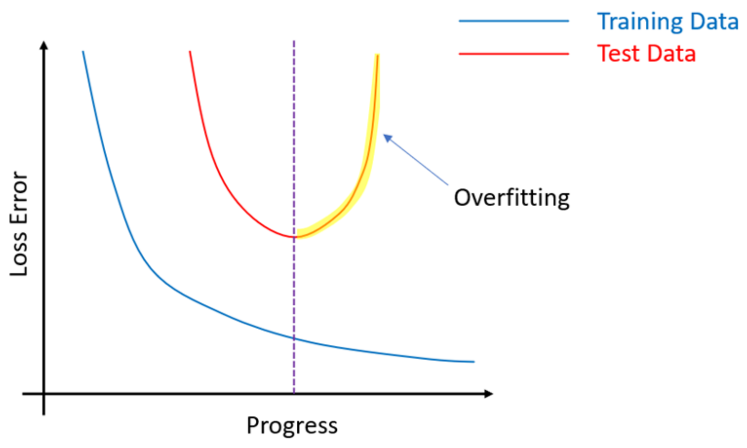
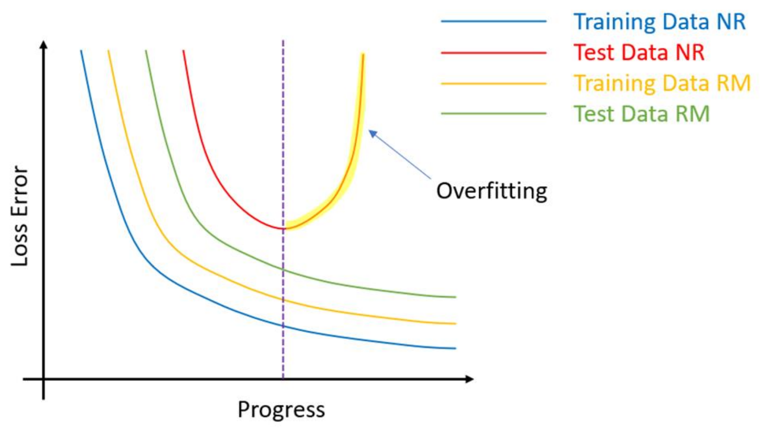


Image: Example of overfitting (black bear vs white bear). If black bear is always in pictures with a green background and white bear's images have white background, the model could be overfit predicting based on the background rather than the animal, hence performing poorly on images on an animal in the zoo.

<u>Regularization</u> techniques are used to calibrate the linear regression models to minimize the adjusted loss function and prevent overfitting or underfitting. Using Regularization, we can fit our machine learning model appropriately without observing the characteristic U-shaped overfitting loss error on a given test set. Regularization techniques are commonly

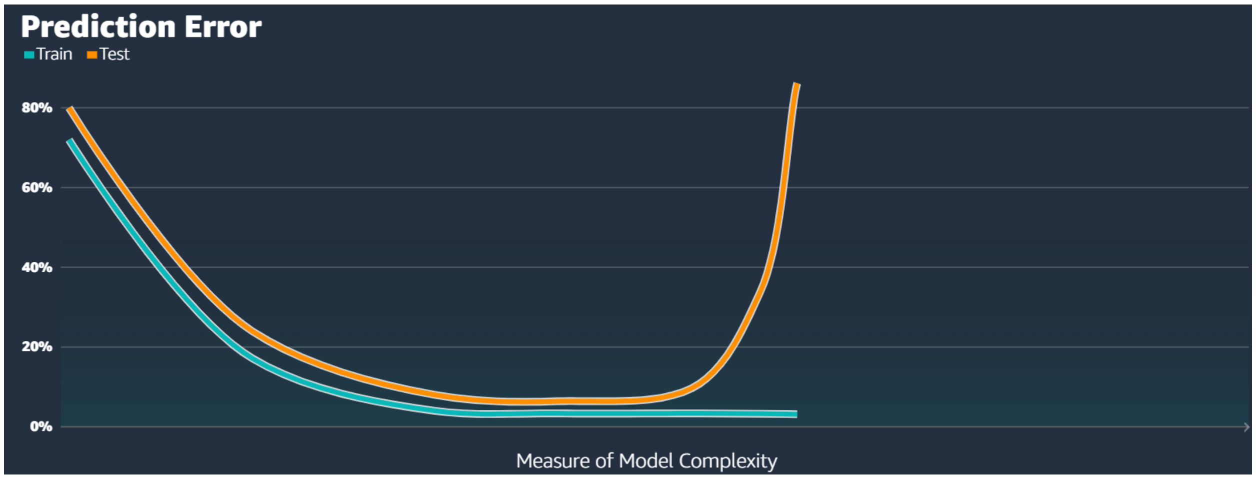
used to force data to fit an assumption that data are *independent, identically distributed,* or *i.i.d.*



Comparison of loss errors of non-regularized models (NR) and regularized models (RM)

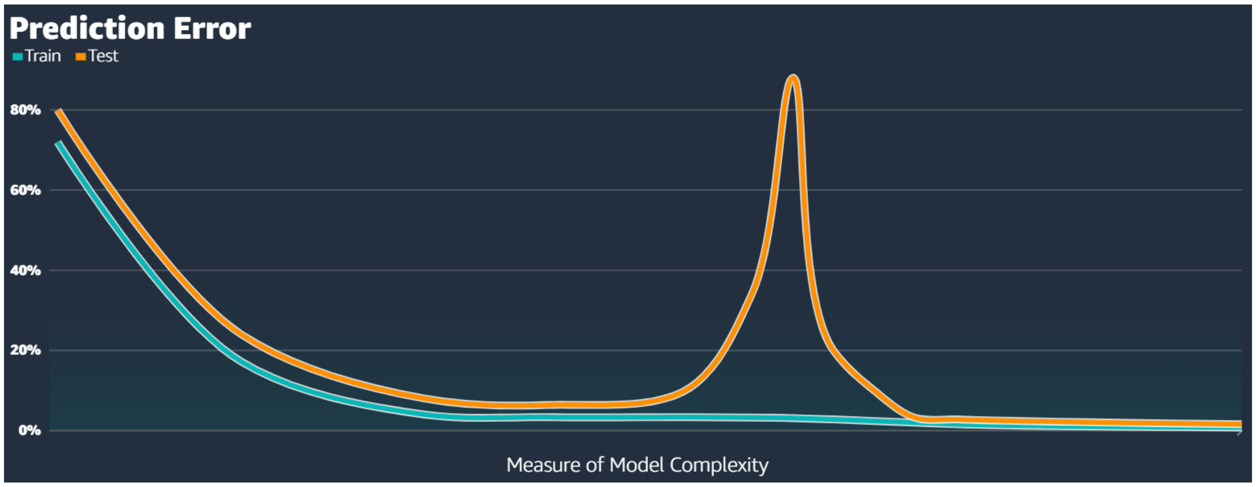
Double Descent:

We already saw a graph of Loss Error vs Progress (or model complexity). Under the classical bias-variance tradeoff, as we move further right along the x axis (i.e. increasing the complexity of our model), we overfit and expect the test error to skyrocket further and further upwards even as the train error continues decreasing.



References: https://mlu-explain.github.io/double-descent/

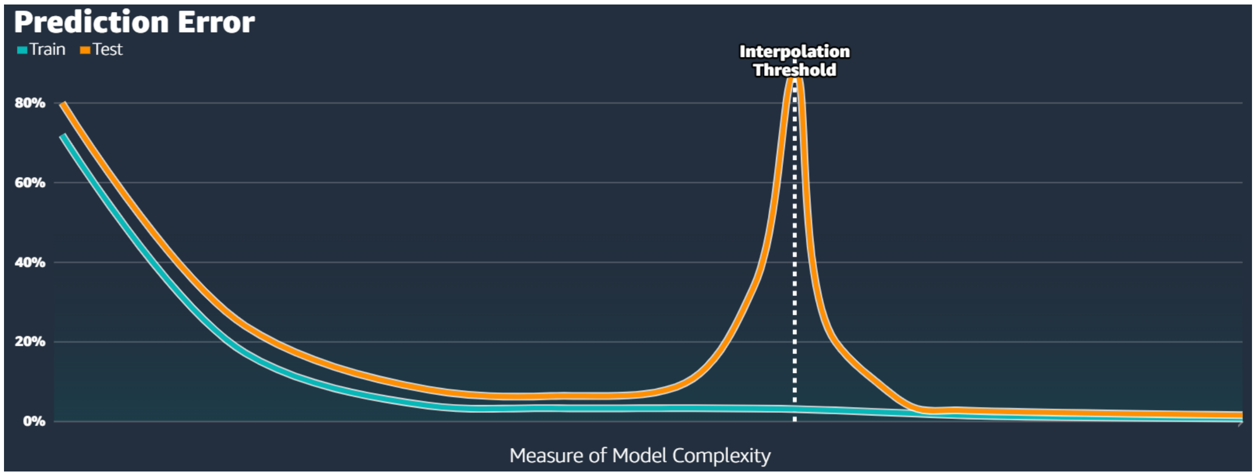
However, what we observe is quite different. Indeed the error does shoot up, but it does so before descending back down to a new minimum. In other words, even though our model is extremely overfitted, it has achieved its best performance, and it has done so during this second descent (hence the name, double descent)!



References: https://mlu-explain.github.io/double-descent/

We call the under-parameterized region to the left of the second descent the classical regime, and the point of peak error the interpolation threshold. In the classical regime, the bias-variance tradeoff behaves as expected, with the test error drawing out the familiar U-shape.

To the right of the interpolation threshold, the behavior changes. We call this over-parameterized region the interpolation regime. In this regime, the model perfectly memorizes, or interpolates, the training data. That is, every model passes exactly through the given training data, thus the only thing that changes is how the model connects the dots between these data points.



References: https://mlu-explain.github.io/double-descent/

XOR Problem

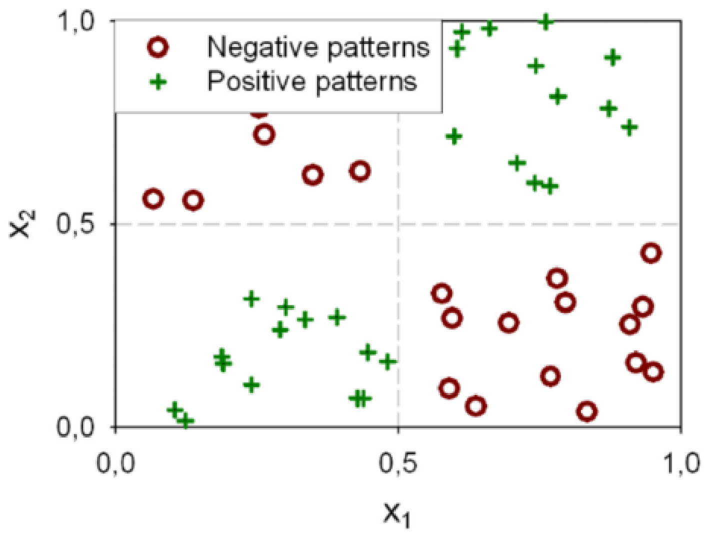
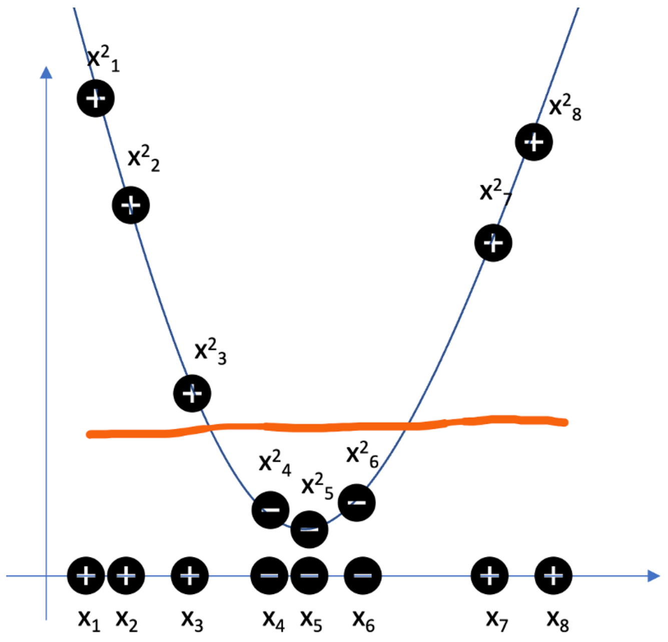


Image1: Graphical representation of the XOR problem
(source: https://www.researchgate.net/figure/Graphical-representation-of-the-XOR-problem_fig1_221303423)

The XOR problem is a classic example of a problem that is not linearly separable in two dimensions. The XOR problem is defined as a binary classification problem where the goal is to separate a dataset of points into two classes, such that the points in one class have the value (0,0) or (1,1) for their two input features, and the points in the other class have the value (0,1) or (1,0). Since the points cannot be separated by a straight line in two dimensions, it is impossible to solve this problem using a traditional linear classifier.

However, the XOR problem can be solved by transforming the input data into a higher-dimensional space using a kernel function (feature transformation), and then applying a linear classifier in that space. The kernel function maps the input data into a new space where the data becomes linearly separable.



In the example above, binary class data lie along the line, $y_i = 0$, and are seemingly inseparable. However, by transforming the data such that $y_i = x_i^2$, we see that the data are transformed into a parabola and picking a linear classifier to separate the data becomes trivial.

Further Reading

Support Vector Machines
https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3

Perceptrons
https://towardsdatascience.com/what-is-a-perceptron-basics-of-neural-networks-c4cfea20c590

Activation Functions
https://en.wikipedia.org/wiki/Activation_function

Logistic Regression
Logistic regression - Wikipedia

Softmax Activation Function
https://en.wikipedia.org/wiki/Softmax_function

Generalization, Regularization, and Overfitting
https://mlu-explain.github.io/double-descent/

XOR Problems
https://towardsdatascience.com/how-neural-networks-solve-the-xor-problem-59763136bdd7