INDIAN INSTITUTE OF TECHNOLOGY

BHU, VARANASI

Summer Project Report

Robustness: Hinge vs Huber Loss

July 25, 2018

Submitted by:
Vikas Kumar Ojha (159101160)
B.Tech, Third Year
Computer Science and Engineering
Manipal University Jaipur

Supervisor:
Dr. K.K. Shukla
Professor, Dept. of Computer Science and Engineering
IIT BHU, Varanasi

Table of Contents

1. Introduction

Machine learning is the science of getting computers to act without being explicitly programmed. The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it. Many researchers also think it is the best way to make progress towards human-level AI.

The learning process in ML techniques can be divided into two main categories, supervised and unsupervised learning. In *supervised learning*, a set of data instances are used to train the machine and are labelled to give the correct result. However, in *unsupervised learning*, there are no pre-determined data sets and no notion of the expected outcome, which means that the goal is harder to achieve.

Classification is among the most common methods that goes under supervised learning. It uses historical labelled data to develop a model that is then used for future predictions. Therefore, researchers make use of this knowledge to develop classification models that can make inference based on historical cases. It is useful to note that all the techniques used in this project fall under classification models.

2. Objective

The support vector machine (SVM) is a popular classifier in machine learning, but it is not robust to outliers. The main objective of this project is to show the comparison between Huber Loss (for classification) and Hinge Loss with respect to robustness to outliers. We have added 0%, 15%, and 30% Noise to Training and Validation set of the data and then checked the performance of our model. We have applied Linear SVM and SGD classifier model in this project.

3. Methodology and Framework

3.1 Machine Learning Algorithms

3.1.1 Support Vector Machine (SVM)

Support vector machines (SVMs) are *supervised learning models* with associated learning algorithms that analyse data used for *classification* and *regression* analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVM functions by selecting critical samples from all classes known as support vectors and separating the classes by generating a linear function that divides them as broadly as possible using these support vectors. Therefore, it can be said that a mapping between an input vector to a high dimensionality space is made using SVM that aims to find the most suitable hyperplane that divides the data set into classes. This linear classifier aims to maximize the distance between the decision hyperplane and the nearest data point, which is called the marginal distance, by finding the best suited *Hyperplane*. A good margin is one where this separation is larger for both the classes. It allows the points to be in their respective classes without crossing to other class.

In addition to performing linear classification, SVMs can efficiently perform a *non-linear classification* using what is called the *kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces.

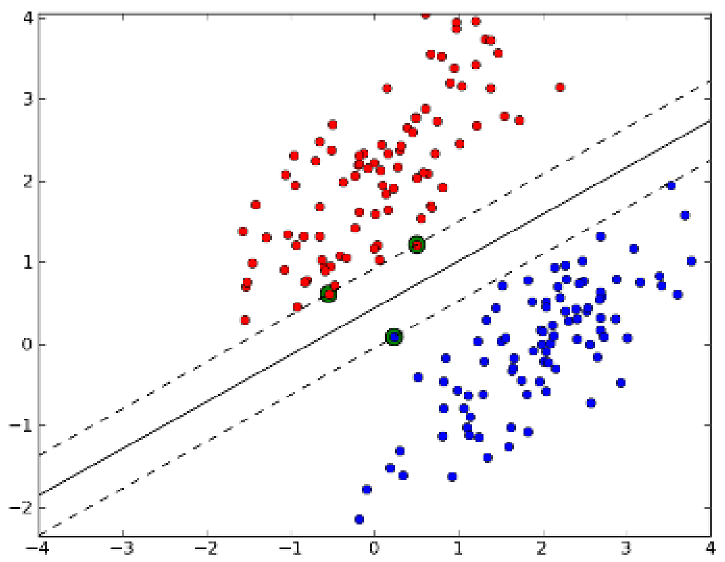


Figure 1: An example of SVM classification

Linear SVM

We are given a training dataset of n points of the form

$(x_1, y_1),\ldots, (x_n, y_n)$

where the y_i are either 1 or −1, each indicating the class to which the point x_i belongs. Each x_i is a *p*-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points x_i for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point x_i from either group is maximized.

Any *hyperplane* can be written as the set of points *x* satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

where *w* is the (not necessarily normalized) normal vector to the hyperplane. The parameter $b / ||w||$ determines the offset of the hyperplane from the origin along the normal vector w.

The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support vector machine weights have also been used to interpret SVM models in the past.

3.1.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under *convex* loss functions such as (linear) *Support Vector Machines* and *Logistic Regression*. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

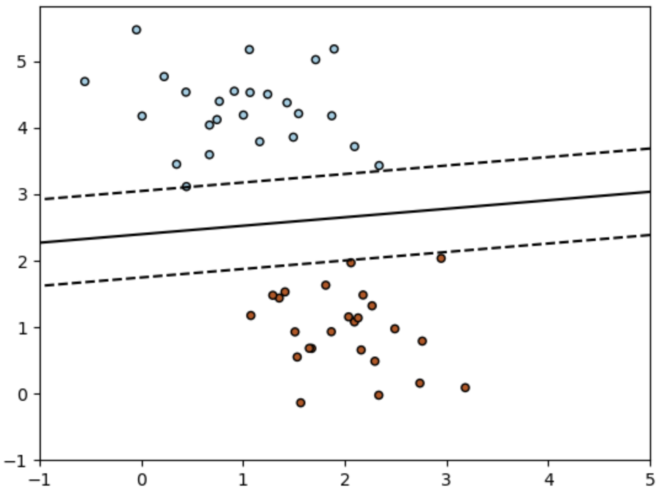


Figure 2: Maximum margin separating hyperplane using a linear Support Vector Machines classifier trained using SGD

Given a set of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbf{R}^m$ and $y_i \in \{-1, 1\}$ our goal is to learn a linear scoring function $f(x) = w^T x + b$ with model parameters $w \in \mathbf{R}^m$ and intercept $b \in \mathbf{R}$. In order to make predictions, we simply look at the sign of $f(x)$. A common choice to find the model parameters is by minimizing the regularized training error given by

$$E(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

where L is a loss function that measures model (mis)fit and R is a regularization term (aka penalty) that penalizes model complexity; $\alpha > 0$ is a non-negative hyperparameter

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing.

3.2 Loss function

In mathematical optimization, statistics, econometrics, decision theory, machine learning and computational neuroscience, a loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. An optimization problem seeks to minimize a loss function.

In machine learning and mathematical optimization, **loss functions for classification** are computationally feasible loss functions representing the price paid for inaccuracy of predictions in classification problems (problems of identifying which category a particular observation belongs to). Given *X* as the vector space of all possible inputs, and *Y = {–1,1}* as the vector space of all possible outputs, we wish to find a function *f : X ----> {R}* which best maps *x* to *y*. However, because of incomplete information, noise in the measurement, or probabilistic components in the underlying process, it is possible for the same x to generate different y.

$$I[f] = \int_{X\times Y} V(f(\vec{x}), y)p(\vec{x}, y)\, d\vec{x}\, dy$$

where *V(f(x),y)* is the loss function, and *p(x,y)* is the probability density function of the process that generated the data.

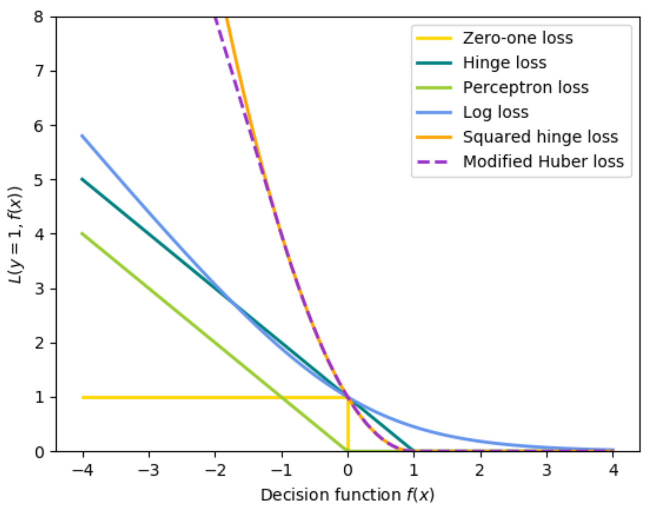


Figure 3: Comparison of various Loss function

3.2.1 Hinge Loss

The hinge loss is a loss function used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for support vector machines (SVMs). For an intended output t = ±1 and a classifier score y, the hinge loss of the prediction y is defined as

$$\ell(y) = \max(0, 1 - t \cdot y)$$

Note that y should be the "raw" output of the classifier's decision function, not the predicted class label. For instance, in linear SVMs, **y = w.x + b**, where *(w,b)* are the parameters of the hyperplane and *x* is the point to classify.

It can be seen that when t and y have the same sign (meaning y predicts the right class) and |y| > 1, the hinge loss *l(y) = 0*, but when they have opposite sign, l(y) increases linearly with y (one-sided error).

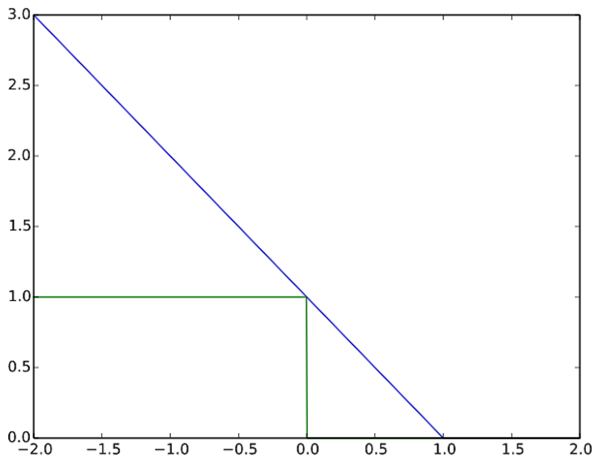


Figure 4: Plot of hinge loss (blue, measured vertically) vs. zero-one loss (measured vertically; misclassification, green: y < 0) for t = 1 and variable y (measured horizontally). Note that the hinge loss penalizes predictions y < 1, corresponding to the notion of a margin in a support vector machine.

Squared Hinge Loss

The hinge loss is a convex function, so many of the usual convex optimizers used in machine learning can work with it. It is not differentiable, but has a subgradient with respect to model parameters w of a linear SVM with score function *y = w.x* that is given by

$$\frac{\partial \ell}{\partial w_i} = \begin{cases} -t \cdot x_i & \text{if } t \cdot y < 1 \\ 0 & \text{otherwise} \end{cases}$$

However, since the derivative of the hinge loss at *ty = 1* is undefined, *smoothed* versions may be preferred for optimization, such as the quadratically smoothed

$$\ell_\gamma(y) = \begin{cases} \frac{1}{2\gamma} \max(0, 1 - ty)^2 & \text{if } \ ty \geq 1 - \gamma \\ 1 - \frac{\gamma}{2} - ty & \text{otherwise} \end{cases}$$

suggested by Zhang.[8] The modified Huber loss L is a special case of this loss function with gamma = 2, $L(t,y) = 4l_2(y)$.

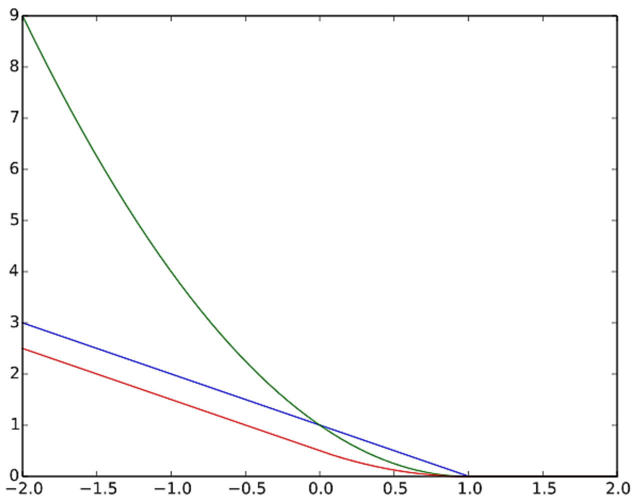


Figure 5: Plot of three variants of the hinge loss as a function of $z = ty$: the "ordinary" variant (blue), its square (green), and the piece-wise smooth version by Rennie and Srebro (red).

3.2.2 Huber Loss

In statistics, the Huber loss is a loss function used in **robust regression**, that is **less sensitive to outliers** in data than the squared error loss. A variant for classification is also sometimes used.

The Huber loss function describes the penalty incurred by an *estimation procedure f*. Huber (1964) defines the loss function piecewise by

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

This function is quadratic for small values of a, and linear for large values, with equal values and slopes of the different sections at the two points where |a|=delta . The variable a often refers to the residuals, that is to the difference between the observed and predicted values *a = y-f(x)*, so the former can be expanded to

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

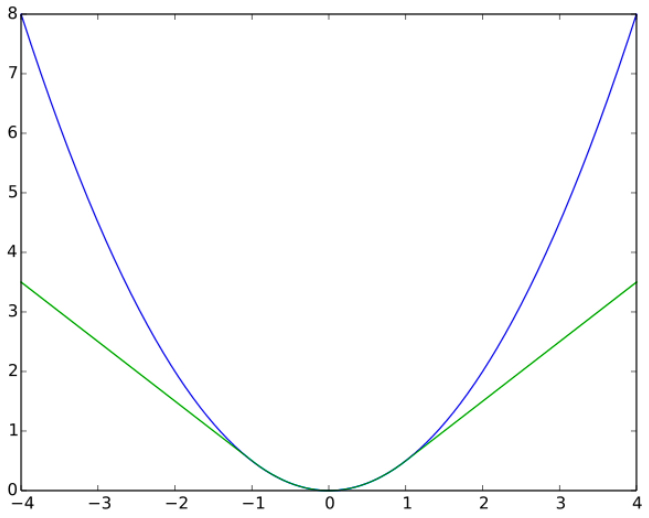


Figure 6: Plot of Huber loss (green, delta =1} and squared error loss (blue) as a function of *y-f(x).*

As defined above, the Huber loss function is **convex** in a uniform neighborhood of its minimum *a=0*, at the boundary of this uniform neighborhood, the Huber loss function has a differentiable extension to an affine function at points *a= -delta* and *a= delta*. These properties allow it to combine much of the sensitivity of the mean-unbiased, minimum-variance estimator of the mean (using the quadratic loss function) and the robustness of the median-unbiased estimator (using the absolute value function).

Modified Huber

For classification purposes, a variant of the Huber loss called modified Huber is sometimes used. Given a prediction f(x) (a real-valued classifier score) and a true binary class label y in {+1,-1}, the modified Huber loss is defined as

$$L(y, f(x)) = \begin{cases} \max(0, 1 - y\, f(x))^2 & \text{for } y\, f(x) \geq -1, \\ -4y\, f(x) & \text{otherwise.} \end{cases}$$

The term *max(0,1-y\,f(x)*) is the hinge loss used by support vector machines; the quadratically smoothed hinge loss is a generalization of *L*.

3.3 Machine Intelligence Library

Scientific computing libraries like: matplotlib, numpy, pandas and scikit-learn were used. All experiments on the classifiers described in this project were conducted using libraries from Python 3 in Jupyter notebook learning environment.

3.3.1 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. You can control the defaults of almost every property in matplotlib: figure size and dpi, line width, colour and style, axes, axis and grid properties, text and font properties and so on.

3.3.2 Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

3.3.3 Sci-kit learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Among other things, it offers:

- Simple and efficient tools for data mining and data analysis

- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

3.3.4 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.
Pandas is a NumFOCUS sponsored project. It has Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.

3.4 The Dataset

Four Datasets have been used in this project for comparison of Hinge and Huber Loss with respect to robustness.

3.4.1 Pima Indians Diabetes Dataset

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

The datasets consists of several medical predictor variables and one *target variable, Outcome*. There are 8 predictor variables like the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. This data set consist of 768 patient records.

3.4.2 ILPD (Indian Liver Patient Dataset) Dataset

This data set contains 416 liver patient records and 167 non-liver patient records. The data set was collected from north east of Andhra Pradesh, India. Selector is a class label used to divide into groups (liver patient or not). This data set contains 441 male patient records and

142 female patient records. This data set contains 10 variables that are age, gender, total Bilirubin, direct Bilirubin, total proteins, albumin, A/G ratio, SGPT, SGOT and Alkphos.

3.4.3 Breast Cancer Wisconsin (Diagnostic) Dataset

The objective of the dataset is to predict the patients' stage of breast cancer that can be Benign or Malignant. Ten real-valued features are computed for each cell nucleus: like radius, texture, perimeter, smoothness etc. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The Dataset contains total 569 instances.

3.4.4 Abalone Dataset

The objective of the dataset is to predict the sex of the abalone out of Male, Female or Infant. There are 8 attributes like length, height, diameter, rings, shell weight and so on. The Dataset contains total 4177 instances. Data comes from an original (non-machine-learning) study: "The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait", Sea Fisheries Division.

4. Work done

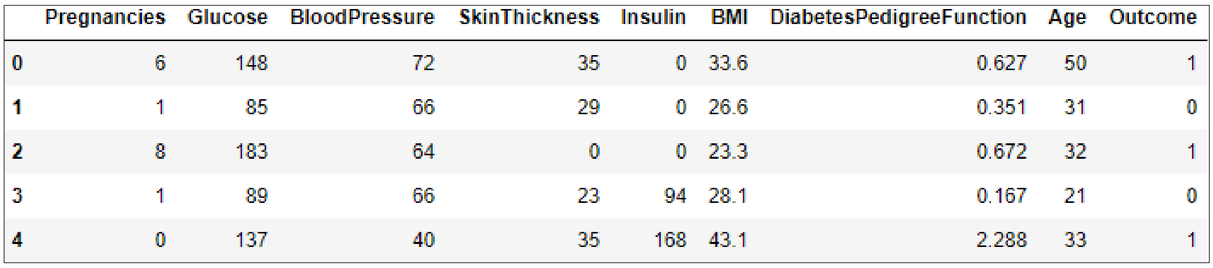
4.1 Experimental Setup

All experiments in this study were conducted on a laptop computer with Intel Core(TM) i5 5500HQ CPU @ 2.30GHz x 4, 4GB of DDR3 RAM.
The datasets were partitioned into three sets: 70% (training), 20% (validation) and 10% (testing).

4.2 Implementation

Reading Dataset

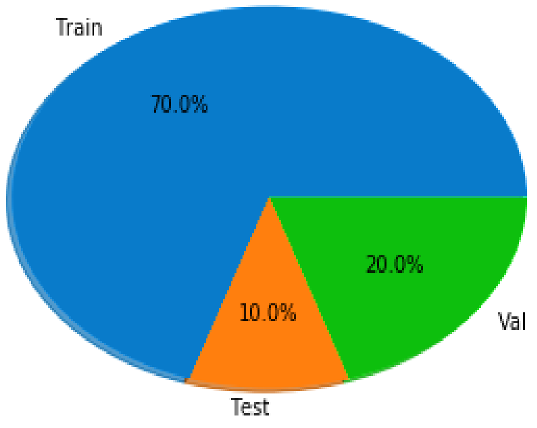
```
df = pd.read_csv('diabetes.csv')
df.head()
```

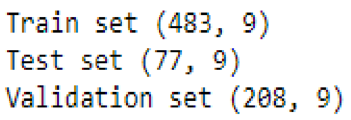


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Splitting Dataset

```
traindf, valdf = train_test_split(df, test_size = 0.3)
labels = 'Train', 'Test', 'Val'
plt.pie([70, 10, 20], labels=labels, autopct='%1.1f%%', shadow=True)
plt.show()
print("Train set", traindf.shape)
print("Test set", testdf.shape)
print("Validation set", valdf.shape)
```





```
Train set (483, 9)
Test set (77, 9)
Validation set (208, 9)
```

Function to add Noise to data

```
def add_labelnoise(noise_level, seed=None):
    np.random.seed(seed)
    l = int(noise_level*len(df.index))
    for i in range(l):
        if df['Outcome'][i]==1:
            df['Outcome'][i]=0
        elif df['Outcome'][i]==0:
            df['Outcome'][i]=1
```

Linear SVC

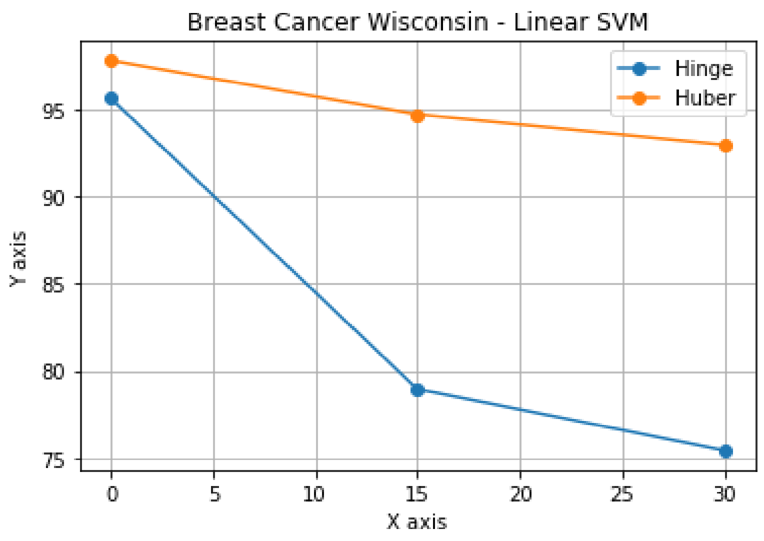
```
clf = LinearSVC(penalty='l1', loss='hinge', dual=False, tol=1e-3, C=1.0,multi_class='ovr', fit_intercept=True, intercept_scaling=1,
class_weight=None, verbose=0, random_state=None, max_iter=1000)
clf.fit(X,y)
print("Accuracy :", clf.score(test_X, test_Y))
scores = cross_val_score(clf, val_X, val_Y, cv=5, scoring='accuracy')
print("Cross_val score :", scores.mean())
```

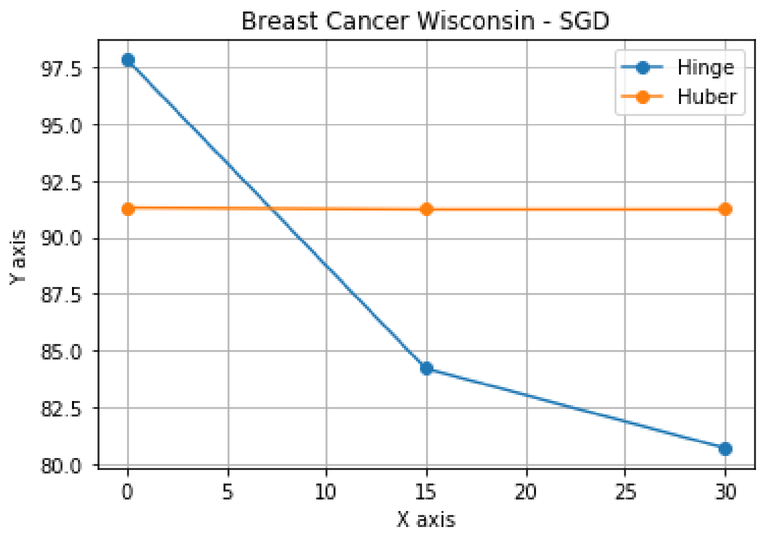
SGD Classifier

```
clf = SGDClassifier(loss='modified_huber', penalty='l1', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=1e-3, shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, class_weight=None, warm_start=False, average=False, n_iter=None)
clf.fit(X,y)
print("Accuracy :", clf.score(test_X, test_Y))
scores = cross_val_score(clf, val_X, val_Y, cv=5, scoring='accuracy')
print("Cross_val score :", scores.mean())
```

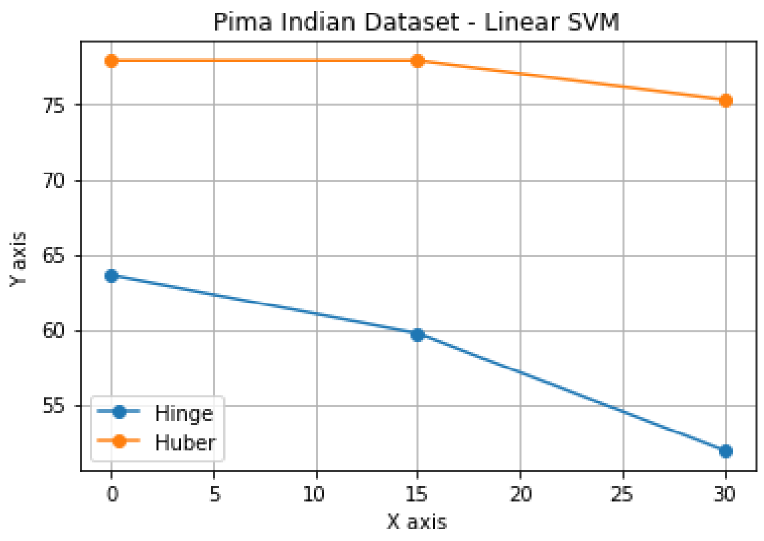
4.3 Result

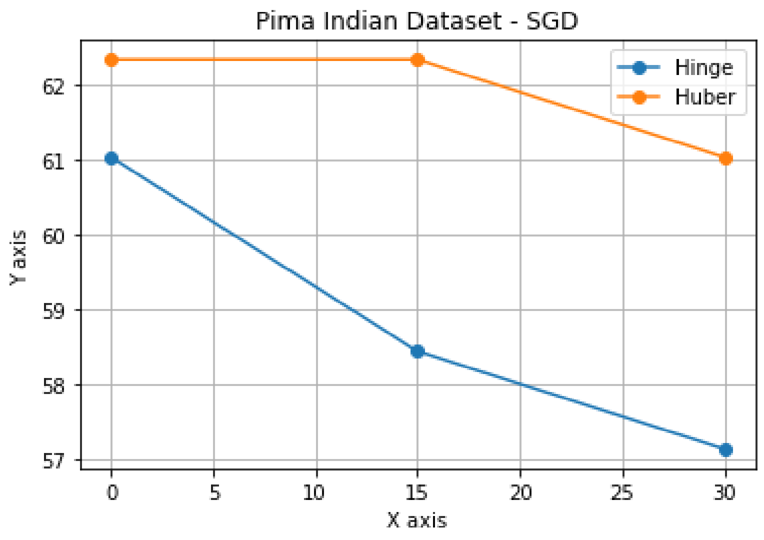
4.3.1 Breast Cancer Wisconsin (Diagnostic) Dataset



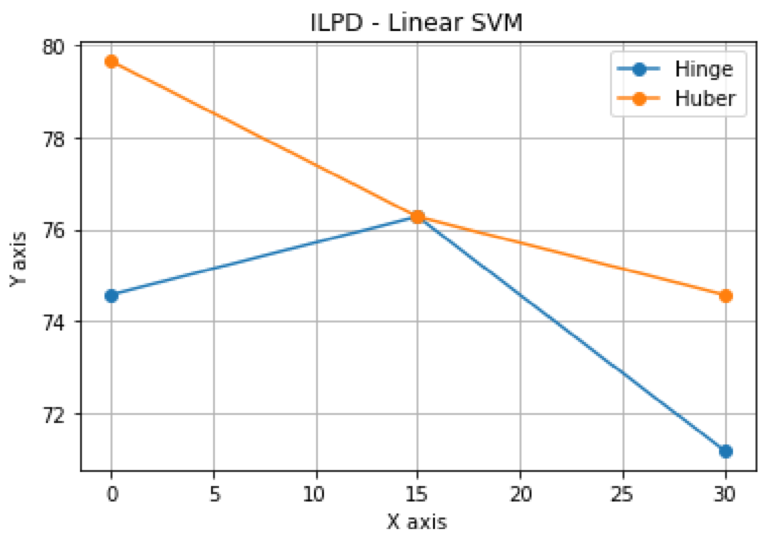


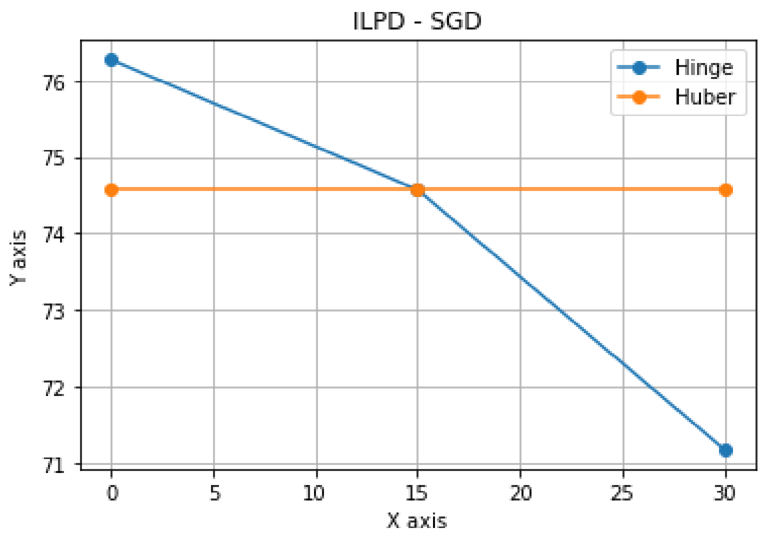
4.3.2 Pima Indians Diabetes Dataset



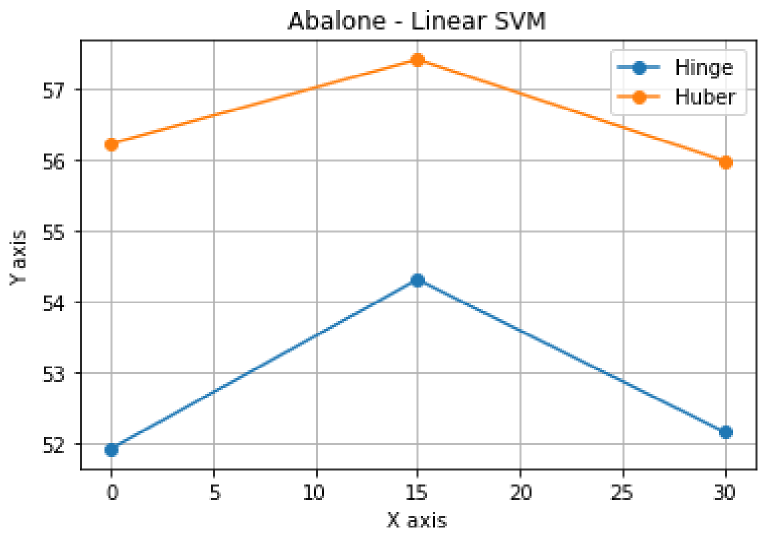


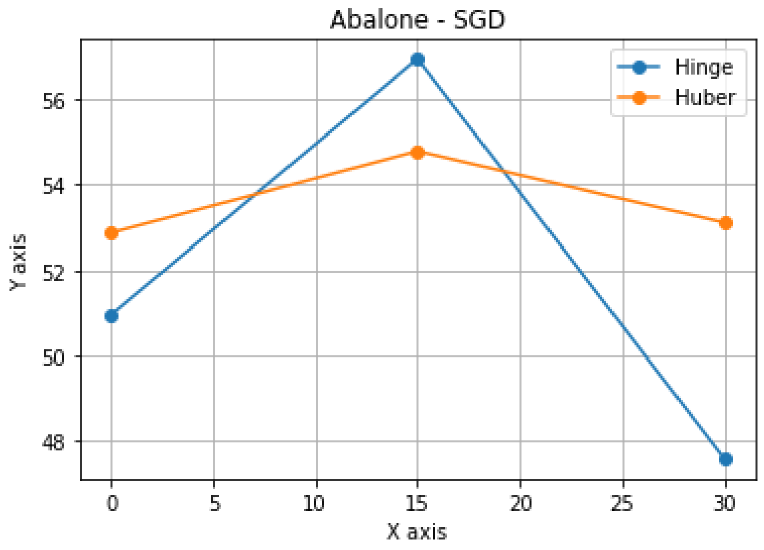
4.3.3 ILPD (Indian Liver Patient Dataset) Dataset





4.3.4 Abalone Dataset





5. Conclusion

Table: Comparison of Hinge and Huber Loss

Dataset	Model	Penalty	Hinge Loss			Huber Loss		
			0%	15%	30%	0%	15%	30%
Pima Indian	Linear SVM	L2 | L1	63.63	59.74	51.94	77.92	77.92	75.32
	SGD Classifier	L1	61.03	58.44	57.14	62.33	62.33	61.03
Abalone	Linear SVM	L2 | L1	51.91	54.30	52.15	56.22	57.41	55.98
	SGD Classifier	L1	50.95	56.93	47.60	52.87	54.78	53.11
Breast Cancer Wisconsin	Linear SVM	L2 | L1	95.65	78.94	75.43	97.82	94.73	92.98
	SGD Classifier	L1	97.82	84.21	80.71	91.30	91.22	91.22
ILPD	Linear SVM	L2 | L1	74.57	76.27	71.18	79.66	76.27	74.57
	SGD Classifier	L1	76.27	74.57	71.18	74.57	74.57	74.57

The plots for all the four datsets show that the Hinge loss is more prone to outliers (noise) than Huber loss. Further we can also see from the table above that change in accuracy (increase or decrease) due to noise is less in case of Huber loss as compared to the Hinge loss. This is true for both Linear SVM and SGD classifier. Thus, we conclude that the Huber loss for classification is more robust to the outliers.

References

[1] Robust support vector machines based on the rescaled hinge loss function Guibiao Xu, Zheng Cao, Bao-Gang Hu, Jose C. Principe

[2] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (September (3)) (1995) 273–297.

[3] B. Frenay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn. Syst. 25 (May (5)) (2014) 845–869. http://dx.doi.org/ 10.1109/TNNLS.2013.2292894.

[4] Y. Wu, Y. Liu, Robust truncated hinge loss support vector machines, J. Am. Stat. Assoc. 102 (September (479)) (2007) 974–983