Breast Cancer Detection Through Multiple Classifiers

Josh Ascher*, **Seyed Amir Mohammad Bagheri**†

Abstract

Breast cancer diagnosis is an invasive and prohibitive process. We implement multiple binary classifiers that diagnose patients without the use of biopsies and extensive testing. We show that each of these classifiers perform well across multiple metrics, and that tuning hyperparameters and using the kernel trick with various kernels gives the best results.

1 Introduction

Breast cancer is the most prevalent cancer among women worldwide. Early and accurate detection is crucial for improving survival rates and enhancing the quality of life for patients. Traditional diagnostic methods, such as biopsies, while highly accurate, are invasive, expensive, and time-consuming. This make prohibitive for many, particularly in impoverished communities. Machine learning techniques have emerged as powerful tools for developing cost-effective, non-invasive, and reliable diagnostic systems for breast cancer classification.

Logistic regression, a binary classifier, is frequently used for cancer diagnoses due to its interpretability and simplicity. It has been employed in studies to model the relationship between various clinical features (like the features we describe in Section 1.2) and breast cancer diagnoses.

Support vector machines (SVMs) have demonstrated strong performance in breast cancer classification due to their ability to handle high-dimensional data and non-linear relationships through the kernel trick. Studies have shown that SVMs achieve high accuracy in distinguishing malignant and benign tumors. Their robustness to overfitting, especially in small to medium-sized datasets, makes them a popular choice in breast cancer research.

Cross-validation has become an essential component in developing reliable diagnostic models for breast cancer. By partitioning the dataset into training and validation subsets, cross-validation ensures that the model generalizes well.

Ensemble methods, particularly AdaBoost, have gained significant attention for their ability to improve classification performance by combining multiple classifiers into a single classifier. AdaBoost iteratively focuses on misclassified samples, making it highly effective for breast cancer classification. Studies have shown that ensemble methods like AdaBoost not only improve accuracy but also reduce the likelihood of false negatives, which is critical in breast cancer diagnosis to avoid missing malignant tumors.

The integration of these machine learning techniques into breast cancer classification addresses critical public health concerns. Accurate classification models not reduce the number of false negatives, but also minimize false positives. While reducing the number of false negatives is the most important issue, decreasing the number of false negative reduces unnecessary procedures, patient anxiety, and healthcare costs. Moreover, the development of these models supports the global effort to provide cost-effective diagnostic tools to underserved areas.

1.1 Prior Work

As mentioned above, machine learning techniques have become popular tools for diagnosing cancer and other debilitating medical conditions over the last few decades. For example, logistic regression has been used for cancer classification for at least three decades Choong and de Silva [1996]. We describe the data in more detail in Section 1.2 but note here that several papers have been published using logistic regression to analyze this data set (Gupta and Huang [2008], Pan et al. [2016]). Similarly, the other classifier we study, sup-

*Department of Computer Science, Drexel University, Philadelphia, PA, USA. Email: ja3443@drexel.edu

†Civil, Architectural and Environmental Engineering, Drexel University, Philadelphia, PA, USA. Email: sb4628@drexel.edu

port vector machines, has also been used extensively for decades. Again, just on the dataset we are studying, this technique was employed several times (Li et al. [2005], Utkin et al. [2013]).

1.2 Data

Our dataset came from the University of California Irvine Machine Learning Repository (UCIMLR) Wolberg et al. [1993]. It was donated by researchers at the University of Wisconsin. The first paper to use this dataset studied support vector machines Street et al. [1993]. The dataset includes 569 samples and 30 features. The features come from attributes observed when imaging breast masses. These include metrics like radius and area, as well as texture and concavity. Each sample is classified as 'B' benign, or 'M' malignant. We convert these values to 0 and 1 respectively (note that in Section 2.2 we use -1 and 1 respectively).

The first column of the data is an ID number. We drop this as it does not affect our classification. After this preprocessing, we normalize our data (z-scoring) to ensure that feature scale does not affect our classifier.

We also note that there is a moderate class imbalance in this dataset, as is common in biological datasets. The minority class is the positive class (malignant), representing around 37% of the samples. We discuss strategies for dealing with this imbalance in Section 2.3.

2 Methods

In order to best analyze this data, we implement multiple binary classifiers, as well as techniques for combining classifiers and eliminating class imbalances. Let $\mathcal{X}$ be the set of observations, and $\mathcal{Y}$ be the corresponding labels. Note that $\mathcal{X}$ has size $N \times D$ and $\mathcal{Y}$ has size $N \times 1$, where N and D are detailed above. In order to train our model, we split the data into training and validating sets using cross validation (see Section 2.4). Let $\mathcal{X}_t, \mathcal{X}_v$ be the training and validation sets respectively, with a corresponding split of $\mathcal{Y}$.

2.1 Logistic Regression

As mentioned earlier, each of our samples belongs to one of two classes: negative (benign mass) or positive (malignant mass). Throughout this subsection, let $\mathcal{Y} = \{0, 1\}$. Logistic regression is a technique for computing the probability that a given sample is in the positive class Cox [1958]. Specifically, for a sample $\mathbf{x} \in \mathcal{X}_t$, we compute

$$\mathbb{P}(y = 1 \mid \mathbf{x}) = g(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{x}\mathbf{w}+b)}}$$

where $\mathbf{w}$ is a $D \times 1$ vector of weights for each feature, and b is a bias scalar. For simplicity in computation, we modify $\mathbf{x}$ and $\mathbf{w}$ to include b:

$$\mathbf{x} = [1 \quad x_1 \quad x_2 \quad \cdots \quad x_D], \quad \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

This yields

$$g(\mathbf{x}) = g(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}\mathbf{w}}} \tag{2.1}$$

Note that the function $g(z) = \frac{1}{1+e^{-z}}$ takes values in the range $[0, 1]$, is monotonically increasing, and is continuous, so it is a cumulative distribution function Wikipedia contributors [2024]. So, we can interpret the output as a probability. Then, we want to find weights $\mathbf{w}$ that maximize the probability of correctly classifying the sample $\mathbf{x}$.

Fix a training sample $\mathbf{x}$, and let y be the target value. Let $\hat{y}$ be the prediction that $\mathbf{x}$ belongs to the positive class, i.e. $y = 1$. Then,

$$\hat{y} = \frac{1}{1 + e^{-\mathbf{x}\mathbf{w}}}$$

From this, we can compute the probability (likelihood) that this prediction is correct:

$$J = (\hat{y})^y (1 - \hat{y})^{(1-y)}$$

To maximize the probability of our prediction being correct, and hence minimize our error, we want to maximize J. Note that for a fixed sample, J is a function of $\mathbf{w}$. So, we can maximize J by taking the derivative with respect to $\mathbf{w}$. However, because this function involves multiple exponential functions, it is easier to maximize the logarithm of J:

$$\begin{aligned} J' = \log J &= \log\left((\hat{y})^y (1 - \hat{y})^{(1-y)}\right) \\ &= y \log \hat{y} + (1 - y) \log (1 - \hat{y}) \end{aligned}$$

Since the logarithm is an increasing function, J' is maximized when J is maximized. Finally, note that maximizing J' is equivalent to minimizing $J'' = -J'$. We abuse notation by calling this J:

$$J = -\left(y \log \hat{y} + (1 - y) \log (1 - \hat{y})\right) \tag{2.2}$$

In general, there are two common approaches to maximizing J:

1. Directly solve for the vector $\mathbf{w}$ that maximizes J by setting $\frac{dJ}{d\mathbf{w}}$ to zero and solving for $\mathbf{w}$
2. Iteratively approach an optimal solution by computing $\frac{dJ}{d\mathbf{w}}$ at each timestep and subtracting this derivative from the current weights

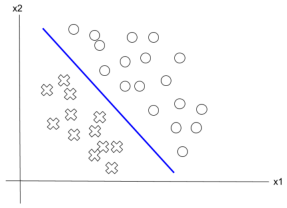

Figure 1: An example of linearly separable data

There is no closed form direct solution, so our implementation iteratively solves for $\mathbf{w}$. Because we are computing the derivative (gradient) of J, and this is a minimization problem, the technique we use is gradient descent. Specifically, we fix some number M epochs and a learning rate η. In each epoch, we compute

$$\frac{dJ}{d\mathbf{w}} = \frac{d}{d\mathbf{w}}\left(-\left(y\log\hat{y} + (1-y)\log\left(1-\hat{y}\right)\right)\right)$$

subbing in for $\hat{y}$ and solving gives

$$= \mathbf{x}^T\left(y - \frac{1}{1+e^{-\mathbf{x}\mathbf{w}}}\right)$$

Then, we update the weights

$$\mathbf{w} = \mathbf{w} - \eta \cdot \frac{dJ}{d\mathbf{w}}$$

After the final epoch, we multiply the validation set by the weights to get predictions, which we compare to the target set:

$$\mathcal{X}_v \cdot \mathbf{w} \stackrel{?}{=} \mathcal{Y}_v$$

2.2 Support Vector Machines

When using support vector machines, we are attempting to find a hyperplane that separates the data into distinct halfspaces. In particular, we are trying to find the separating hyperplane that maximizes the distance to the nearest samples (the margin) Vapnik [1995]. The samples nearest to the hyperplane are called the support vectors. In the case that there is a hyperplane that completely splits the data, we say that the data is linearly separable. An example of such a dataset can be seen in Figure 1.

However, our data is not linearly separable. So, we project our data into higher dimensionality using a kernel. We tried multiple kernels: linear, polynomial with varying degree, and the Radial Basis Function (RBF). Here we show the computation using RBF. Throughout this section, let $\mathcal{Y} = \{-1, 1\}$.

The formula for margin is $J = \frac{2}{|\mathbf{w}|}$. Maximizing this formula is the same as minimizing the reciprocal:

$$J = \frac{|\mathbf{w}|}{2} = \frac{1}{2}\sqrt{\mathbf{w}^T\mathbf{w}}$$

We remove the square root as it does not affect the minimization problem. Additionally, we add a constraint for each sample that it is on the correct side of the hyperplane. With these constraints and the objective function J, we can solve the problem with Lagrange multipliers. Incorporating this into J, we have

$$J = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \alpha_i(y_i x_i - 1)$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \left(\boldsymbol{\alpha}^T\text{diag}(\mathcal{Y}_t)\mathcal{X}_t\mathbf{w} - \text{trace}(\text{diag}(\boldsymbol{\alpha}))\right)$$

Unlike in logistic regression, there is a direct solution for minimizing J. We first take the derivative of J with respect to $\mathbf{w}$, which allows us write $\mathbf{w}$ as a function of $\boldsymbol{\alpha}$:

$$\mathbf{w} = \mathcal{X}_t^T\text{diag}(\mathcal{Y}_t)\boldsymbol{\alpha} \tag{2.3}$$

We plug this into J to get the objective function solely in terms of $\boldsymbol{\alpha}$

$$J = -\frac{1}{2}\boldsymbol{\alpha}^T\text{diag}(\mathcal{Y})\mathcal{X}_t^T\mathcal{X}_t\text{diag}(\mathcal{Y}_t)\boldsymbol{\alpha} + \text{trace}(\text{diag}(\boldsymbol{\alpha}))$$

Then we compute the derivative with respect to $\boldsymbol{\alpha}$:

$$\frac{dJ}{d\boldsymbol{\alpha}} = -\text{diag}(\mathcal{Y}_t)\mathcal{X}^T\mathcal{X}_t\boldsymbol{\alpha} + \text{ones}(\text{size}(\boldsymbol{\alpha}))$$

Setting this equal to zero and solving for $\boldsymbol{\alpha}$ gives

$$\boldsymbol{\alpha} = \left(\text{diag}(\mathcal{Y}_t)\mathcal{X}_t^T\mathcal{X}_t\right)^{-1}\text{ones}(\text{size}(\boldsymbol{\alpha}))$$

As mentioned above, we will be using the kernel trick. Let

$$K(\mathbf{x}, \mathcal{X}_t) = e^{-\frac{(\mathbf{x}-\mathcal{X}_t)(\mathbf{x}-\mathcal{X}_t)^T}{2\sigma^2}}$$

be the RBF kernel, for some hyperparameter σ (we implicitly are broadcasting $\mathbf{x}$). We now apply the RBF kernel to our Lagrange Multiplier $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} = \left(\text{diag}(\mathcal{Y})K(\mathcal{X}_t, \mathcal{X}_t)\right)^{-1}\text{ones}(\text{size}(\boldsymbol{\alpha}))$$

Finally, we can plug back into Equation (2.3)

$$g(\mathcal{X}_v) = K(\mathcal{X}_v, \mathcal{X}_t)\text{diag}(\mathcal{Y}_t)\boldsymbol{\alpha}$$

As in logistic regression, we then get our predictions:

$$\mathcal{X}_v \cdot \mathbf{w} \stackrel{?}{=} \mathcal{Y}_v$$

2.3 Adaptive Boosting

While the above classifiers are highly effective on their own, we also implemented an *ensemble* to combine them. Specifically, we used adaptive boosting (AdaBoost) Freund and Schapire [1997]. Boosting is the process of selecting certain samples with lower probability if they have already been correctly predicted using an earlier classifier.

In our implementation, we also test how boosting is impacted by the class imbalance. To offset the class imbalance we use oversampling to get a roughly equal split between the majority class (negative) and the minority class (positive) in our training set. We then run logistic regression using this training set (with the rest of the data as the validation set). Once we have our predictions from logistic regression, we calculate the training accuracy.

We then create our training set for support vector machines by using our calculated logistic regression training accuracy to penalize samples that we correctly predicted. Next, we run SVM (our results are with RBF kernel) on this training set. Finally, we classify all validation samples using the predictions from both classifiers. We give more weight to the classifier with higher training accuracy.

2.4 Cross Validation

Cross-validation is a resampling method used to assess the performance of machine learning models. It ensures that the model generalizes well to unseen data by splitting the dataset into training and validation subsets. Because we do not have a particularly large dataset, we implement S-folds cross validation Stone [1974]. This allows us to train our classifiers on more data by feeding them S distinct training sets. Fixing $S = 5$ gave us training sets with $\frac{\text{len}(\mathcal{X})}{S} \approx 100$ samples each.

To begin, we shuffled the data so that the data was permuted uniformly at random. Then, we took the first ≈ 100 samples as our training set. We ran the classifiers using the rest of the data as the validation set. We then repeated this process with the each batch of samples and computed multiple metrics in each round. Finally, we took the mean of the metrics to see how our classifier performed "on average."

Cross validation significantly increased our classifiers' accuracy. Because the dataset is small, when not using cross validation, there is relatively high chance of getting a "bad" training split. Running multiple iterations of the classifier with distinct data sets offsets "bad" splits by averaging their results with more representative splits.

3 Results

After computing weights and making predictions using logistic regression, support vector machines (equipped with various kernels), and an ensemble of the two, we compute various statistics to measure their performance. The statistics returned from one run of these methods can be see in Table 1.

In general, logistic regression and SVM perfomed very well. Logistic regression was tested with several different numbers of epochs and learning rates. Ultimately, we found that 10^5 epochs with a learning rate of 4×10^{-6} gave particularly good results. It is possible that we could have gotten better results with different hyperparameters, however, increasing the number of epochs drastically increases the runtime.

Method	Training Acc.	Validation Acc.	Precision	Recall	F-measure
Logistic Regression	0.9429	0.9351	0.8964	0.9397	0.9167
SVM (Linear)	0.9649	0.9559	0.9907	0.8943	0.9389
SVM (Polynomial, $p = 2$)	1.0000	0.6996	0.5879	0.6144	0.6005
SVM (Polynomial, $p = 5$)	1.0000	0.7945	0.7375	0.6857	0.7092
SVM (Polynomial, $p = 15$)	0.9086	0.8961	0.8361	0.9149	0.8714
SVM (RBF)	1.0000	0.9650	0.9602	0.9423	0.9508
Ensemble	-	0.9539	0.9322	0.9483	0.9402
Ensemble (Oversampling)	-	0.9276	0.8556	0.9565	0.9032

Table 1: Performance Metrics

SVM was tested with several different kernels. The top performing kernels were linear, RBF, and high degree polynomial ($10 \leq p \leq 15$). We found a large drop off for $p \geq 16$. SVM with RBF was the best performing classifier we tested. While many of it's values were only slightly above logistic regression and SVM with a linear kernel, it had the highest recall. In our case, because of the consequences of misclassifying a malignant mass, this is the most important statistic. SVM with the linear kernel had near perfect precision, which would lead to very few overtreatments. It is possible that some combination of these two would be the best classifer.

We also tested combining logistic regression and SVM through AdaBoost. This ensemble did not perform any better than the best classifiers (it was roughly equal in performance to SVM with RBF). Additionally, we tried incorportating oversampling into this ensemble to combat the class imbalance. This slightly improved recall, the most important metric. However, there was a large drop in precision.

The UCIMLR also lists the validation accuracy and the precision of models used in papers that cite the dataset. Our SVM implementations (with linear and RBF kernels) beats both, while our logistic regression implementation performs slightly worse.

4 Conclusion

Throughout the project, we studied multiple classifiers and how they performed in diagnosing cancer masses. Because cancer diagnosis is invasive and costly, these classifiers are incredibly important tools. However, for any of these tools to be useful, they need to correctly predict whether the observed mass is malignant with high probability. We tested both logistic regression and support vector machine models, along with an ensemble. Logistic regression and SVM equipped with linear and RBF kernels performed particularly well. As seen in Table 1, each had at least 93% validation accuracy, 90% precision, and 90% recall. This means that the models had neither high false positive nor high false negative rates. However, because recall is the most important statistic (the ramifications of marking a malignant mass as benign are catastrophic), one might opt

to use AdaBoost with oversampling, as this had nearly 96% recall. Overall, each of our models has it's own advantages and disadvantages that a clinician would choose to optimize for.

4.1 Future Work

As mentioned above, finer tuning of the hyperparameters could have led to better results. In testing, logistic regression was very sensitive to the number of epochs and learning rate. Additionally, it would be interesting to see if the models performed better if we used duplicated samples to offset the class imbalance. We did not notice a significant increase in the ensemble's performance from oversampling, but that could be because we only used a single training set instead of cross validation.

Another direction to consider is how these models would perform in classifying other forms of cancer and other diseases. For different datasets, it is likely that a different subset of the models would be optimal.

5 Our Contributions

Josh: I implemented the algorithms used in our script, as well as tested them with various hyperparameters. I also wrote the introduction, data, methods, results, and conclusions sections of the paper.

Amir: I reviewed the algorithms used in our script. Also, I wrote the introduction, related works, and methods sections of the paper.

References

Poh Lian Choong and C.J.S. de Silva. A comparison of maximum entropy estimation and multivariate logistic regression in the prediction of axillary lymph node metastasis in early breast cancer patients. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 3, pages 1468–1473 vol.3, 1996. doi: 10.1109/ICNN.1996.549116.

D.R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society*, 1958.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

Mithun Das Gupta and Thomas S. Huang. Bregman distance to l1 regularized logistic regression. *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

Yuangui Li, Zhonghui Hu, Yunze Cai, and Weidong Zhang. Support vector based prototype selection method for nearest neighbor rules. In *International Conference on Computing, Networking and Communications*, 2005.

Jian Pan, Hua Yiang, Xing-Long Liu, Zhiqiang Chen, and Zhaofeng Yan. Bagging-based logistic regression with spark: A medical data mining method. 2016. URL https://api.semanticscholar.org/CorpusID:55949362.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 1974.

William Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Electronic imaging*, 1993. URL https://api.semanticscholar.org/CorpusID:14922543.

Lev V. Utkin, Yulia A. Zhuk, and Anatoly I. Chekh. An ensemble-based feature selection algorithm using combination of support vector machine and filter methods for solving classification problems. 2013. URL https://api.semanticscholar.org/CorpusID:55415889.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Wikipedia contributors. Sigmoid function — Wikipedia, The Free Encyclopedia, 2024. [Online; accessed 8-December-2024].

William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.