# Optimising Area Under the ROC Curve Using Gradient Descent

**Alan Herschtal** ALAN.HERSCHTAL@TEAM.TELSTRA.COM
**Bhavani Raskutti** BHAVANI.RASKUTTI@TEAM.TELSTRA.COM
Telstra Corporation, 770 Blackburn Road, Clayton, Victoria, Australia

## Abstract

This paper introduces RankOpt, a linear binary classifier which optimises the area under the ROC curve (the AUC). Unlike standard binary classifiers, RankOpt adopts the AUC statistic as its objective function, and optimises it directly using gradient descent. The problems with using the AUC statistic as an objective function are that it is non-differentiable, and of complexity $O(n^2)$ in the number of data observations. RankOpt uses a differentiable approximation to the AUC which is accurate, and computationally efficient, being of complexity $O(n)$. This enables the gradient descent to be performed in reasonable time. The performance of RankOpt is compared with a number of other linear binary classifiers, over a number of different classification problems. In almost all cases it is found that the performance of RankOpt is significantly better than the other classifiers tested.

## 1. Introduction

In many binary classification tasks, the aim of the classification is to sort the observations into a list so that the minority class observations are concentrated towards the top of the list. That way, if, due to limited resources, only a small subset of all observations are acted upon, the sorting will enable a high percentage of the observations of interest (the minority class observations) to be included in this subset. In other words, for a given cut-off, or decision threshold, it is desirable to have as many as possible of the minority class observations above the threshold (high true positive rate) together with as few as possible of the majority class observations (low false positive rate). Graphing the true positive rate against the false positive rate as the decision threshold is varied yields the Receiver Operating Characteristic, or ROC curve. The area under the ROC curve, or the AUC, is a decision threshold independent measure of classifier goodness, and has often been used as such (Bradley, 1997; Weiss & Provost, 2001).

Most binary classifiers, however, have as their objective function some other measure, such as mean square error, or one-sided linear or square penalty. When the real objective is to optimise the sorting order, such classifiers are actually solving the wrong problem. Hence they are likely to perform sub-optimally when the performance measure is the AUC. This has been found to be the case empirically on a wide variety of datasets (Perlich et al., 2003). Similarly, if the classifier's objective function is closely related to the AUC then it yields models with better AUC (Yan et al., 2003; Cortes & Mohri, 2004).

In this paper, we introduce RankOpt, an algorithm that optimises the AUC directly. RankOpt searches for the linear model that is optimal for the AUC, using gradient descent to optimise the model coefficients. It is compared with a number of other linear binary classifiers, namely: linear regression (used as a classifier), an SVM with one-sided linear penalty (SVM-L1) (Cristianini & Shawe-Taylor, 2000), an SVM with one-sided square penalty (SVM-L2) (Joachims, 1998; Platt, 1998; Vapnik, 1998), and the centroid classifier (Rocchio, 1971).

Section 2 describes RankOpt's objective function. Section 3 discusses details of RankOpt's algorithm, and several issues of relevance to gradient descent, including the possible existence of local minima, and selection of a starting point. Section 4 describes the experimental procedure, and the datasets that RankOpt was tested on. It also includes the results. Related work is discussed in Section 5, followed by conclusions and future work in Section 6.

## 2. Objective Function

Consider a rectangular dataset of *iid* observations, drawn from a population. The dataset contains $P$ minority class and $Q$ majority class observations, $\vec{x}_j^+,\ j = 1 \ldots P$, and $\vec{x}_k^-,\ k = 1 \ldots Q$. It has $m$ predictor variables, so $\vec{x}_j^+ = \{x_{ij}^+,\ i = 1 \ldots m\}$, where $x_{ij}^+$ is the $j^{\text{th}}$ instance of random variable $X_i^+$. Likewise, $\vec{x}_j^+$ is the $j^{\text{th}}$ instance of vector r.v. $\vec{X}^+$. Equivalent definitions hold for the majority class. A single boolean valued target variable defines the class of

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. 

any observation. Consider an observation pair, consisting of one observation chosen at random from each class $\{\vec{x}_j^+,\ \vec{x}_k^-\}$. The AUC of a model on a given dataset can be expressed as the probability that for such a random observation pair, the score of the minority class observation is greater than that of the majority class observation (Bamber, 1975). If the model is linear, with the coefficients of the predictor variables given by vector $\vec{\beta}$, then ignoring ties,

$$AUC(\vec{\beta}) = \Pr(\vec{\beta}.\vec{X}^+ > \vec{\beta}.\vec{X}^-).$$

This is simply the Mann-Whitney statistic (Mann & Whitney, 1947) scaled by $\frac{1}{PQ}$ (Yan et al., 2003). If we take the heaviside function defined as

$$g(x) = \begin{cases} 0, & x < 0, \\ 0.5, & x = 0, \\ 1, & x > 0 \end{cases}$$

$$\text{then} \quad \widehat{AUC(\vec{\beta})} = \frac{1}{PQ}\sum_{j=1}^{P}\sum_{k=1}^{Q} g(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_k^-))$$

is an unbiased estimator of the AUC.

### 2.1. The Rank Statistic

Since the heaviside function is undifferentiable, it is replaced by the sigmoid function $s(x) = 1/(1+e^{-x})$ (Yan et al., 2003), in order to apply gradient descent. We refer to the resulting approximation to the AUC as the sigmoid rank statistic, or simply the "rank statistic", $R(\vec{\beta})$, defined as

$$R(\vec{\beta}) = \frac{1}{PQ}\sum_{j,k}^{P,Q} s(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_k^-)). \tag{1}$$

Note that $\lim_{|x|\to\infty} s(x) = g(x)$, so that for large $\|\vec{\beta}\|$ the sigmoid rank statistic is a good approximation to the AUC. It is everywhere differentiable, and its first few derivatives are all tightly bounded. It is straightforward to verify that

$$\frac{ds}{dx} = s(1-s), \qquad \frac{d^2s}{dx^2} = s(1-s)(1-2s)$$

The value of the AUC statistic, which is the true objective function, depends on the direction of $\vec{\beta}$ only and not on its magnitude. The rank statistic, on the other hand, depends on both the magnitude and the direction of $\vec{\beta}$. However, at large $\|\vec{\beta}\|$ the rank statistic is a good approximation to the AUC, hence for $\|\vec{\beta}\|$ large enough, the rank statistic is also nearly independent of $\|\vec{\beta}\|$. Hence we may approximate a classifier which optimises the AUC by optimising $R(\vec{\beta})$, constrained to the hypersphere $\|\vec{\beta}\| = B$, in $\vec{\beta}$-space, with $B$ fixed and fairly large. Formally we are after $\vec{\beta}_{OPT} = \text{argmax}\, R(\vec{\beta})$, s.t. $\|\vec{\beta}\| = B$, where the value of $B$ is determined as described in Section 3.

### 2.2. Computational Efficiency

Note that the computational complexity of the rank statistic $R(\vec{\beta})$ is $O(n^2)$ in the number of observations. A calculation of this complexity must be carried out at every step of the gradient descent algorithm, which is prohibitive.

This calculation can be simplified as follows. Observe that the arguments to the sigmoid function in $R(\vec{\beta})$ have a high degree of interdependence. Specifically, for any two minority class observations $\vec{x}_{j_1}^+$ and $\vec{x}_{j_2}^+$, and majority class observations $\vec{x}_{k_1}^-$ and $\vec{x}_{k_2}^-$, the following relationship holds.

$$(\vec{x}_{j_1}^+ - \vec{x}_{k_1}^-) + (\vec{x}_{j_2}^+ - \vec{x}_{k_2}^-) = (\vec{x}_{j_1}^+ - \vec{x}_{k_2}^-) + (\vec{x}_{j_2}^+ - \vec{x}_{k_1}^-).$$

For the four observation pairs formed by combinations of $\vec{x}_{j_1}^+$, $\vec{x}_{j_2}^+$, $\vec{x}_{k_1}^-$, and $\vec{x}_{k_2}^-$, namely $\{\vec{x}_{j_1}^+,\ \vec{x}_{k_1}^-\}$, $\{\vec{x}_{j_1}^+,\ \vec{x}_{k_2}^-\}$, $\{\vec{x}_{j_2}^+,\ \vec{x}_{k_1}^-\}$, and $\{\vec{x}_{j_2}^+,\ \vec{x}_{k_2}^-\}$, the argument to the sigmoid function for any one of these is fully determined by the other three. This means that using all $PQ$ observation pairs to calculate the $R(\vec{\beta})$ is wasteful. The following alternative is therefore proposed.

Randomise the order of the observations. Then balance the data by recycling the $P$ minority class observations until both classes have $Q$ observations. Then in the rank statistic, only consider observation pairs which consist of the $(k \bmod P)$-th minority class observation paired with the $k$-th majority class observation, $k = 1 \ldots Q$. This gives the following linear rank statistic, which is $O(n)$ in the number of majority class observations,

$$R_l(\vec{\beta}) = \frac{1}{Q}\sum_{k=1}^{Q} s(\vec{\beta}.(\vec{x}_{k \bmod P}^+ - \vec{x}_k^-)). \tag{2}$$

Unlike $R(\vec{\beta})$, in $R_l(\vec{\beta})$ no argument to the sigmoid function can be fully determined by any others. Note that $R_l(\vec{\beta})$ is not uniquely defined, as it depends on the ordering of the observations. Nonetheless for any random ordering, the arguments in the remainder of this section hold.

Clearly, for any fixed $\vec{\beta}$, $E[R_l(\vec{\beta})] = E[R(\vec{\beta})]$, but how do the variances compare? We would like to know how much greater the variance of $R_l(\vec{\beta})$ is than that of $R(\vec{\beta})$. Specifically, we are interested in the value of $\frac{var(R_l(\vec{\beta}))}{var(R(\vec{\beta}))}$, which we refer to as the "efficiency loss", $L_e$. Consider firstly $var(R(\vec{\beta}))$.

$$var(R(\vec{\beta})) = var\Big(\frac{1}{PQ}\sum_{j,k}^{P,Q} s\big(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_k^-)\big)\Big)$$

$$= \frac{1}{(PQ)^2}\sum_{j_1,k_1}^{P,Q}\sum_{j_2,k_2}^{P,Q} cov(s(\vec{\beta}.(\vec{x}_{j_1}^+ - \vec{x}_{k_1}^-)), s(\vec{\beta}.(\vec{x}_{j_2}^+ - \vec{x}_{k_2}^-))).$$

This sum of $(PQ)^2$ covariance terms consists of:

i) PQ terms where $j_1 = j_2$ and $k_1 = k_2$, for which the covariance term simply reduces to $var(s(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_k^-))) = V$.

ii) $PQ(Q-1)$ terms where $j_1 = j_2$ and $k_1 \neq k_2$, for which the covariance is $cov(s(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_{k_1}^-)), s(\vec{\beta}.(\vec{x}_j^+ - \vec{x}_{k_2}^-)) = V_1$.

iii) $PQ(P-1)$ terms where $j_1 \neq j_2$ and $k_1 = k_2$. These have covariance $cov(s(\vec{\beta}.(\vec{x}_{j_1}^+ - \vec{x}_k^-)), s(\vec{\beta}.(\vec{x}_{j_2}^+ - \vec{x}_k^-)) = V_2$.

For all other terms $j_1 \neq j_2$ and $k_1 \neq k_2$. Due to the *iid* nature of the data, these have zero covariance. This gives the following expression for the total covariance.

$$var(R(\vec{\beta})) \quad = \quad PQ\big(V + V_1(Q-1) + V_2(P-1)\big) \times \frac{1}{(PQ)^2}$$

Assuming that $P, Q \gg 1$, and that both $V_1 Q$ and $V_2 P$ are $\gg V$, this can be simplified to

$$var(R(\vec{\beta})) \approx V_1/P + V_2/Q.$$

The validity of this last assumption has been verified empirically on several datasets used in experimentation.

Now repeat this process for $var(R_l(\vec{\beta}))$.

$$var(R_l(\vec{\beta})) = var\Big(\frac{1}{Q}\sum_k^Q s\big(\vec{\beta}.(\vec{x}_{k \bmod P}^+ - \vec{x}_k^-)\big)\Big) = \frac{1}{Q^2}\sum_{k_1}^Q\sum_{k_2}^Q cov\big(s(\vec{\beta}.(\vec{x}_{k_1 \bmod P}^+ - \vec{x}_{k_1}^-)), s(\vec{\beta}.(\vec{x}_{k_2 \bmod P}^+ - \vec{x}_{k_2}^-))\big)$$

The calculation of this variance is similar to that of $R(\vec{\beta})$. This time we consider the 3 covariance components: i) $k_1 = k_2$, ii) $(k_1 \bmod P) = (k_2 \bmod P)$ but $k_1 \neq k_2$, and iii) $(k_1 \bmod P) \neq (k_2 \bmod P)$. This yields

$$var(R_l(\vec{\beta})) = \frac{1}{Q}\Big(V + \Big(\frac{Q}{P} - 1\Big)V_1\Big). \tag{3}$$

Let us now consider the ratio of these variances, $L_e$. For heavily imbalanced data ($Q \gg P$), so long as we don't have $V_2 \gg V_1$, both variances reduce to $\approx V_1/P$, and hence $L_e \approx 1$. Since it has been found empirically for several datasets used in our experimentation that $V_1 \approx V_2$, this last assumption appears quite safe. So $R(\vec{\beta})$ has no advantage over $R_l(\vec{\beta}))$ for heavily imbalanced data.

Consider, therefore, balanced data. For $P = Q$, cancelling common terms leaves the variance ratio to be $(V_1 + V_2)/V$. For small $\|\vec{\beta}\|$, the sigmoid is linear, so

$$\frac{V_1}{V} \quad = \quad \frac{cov(s(\vec{\beta}.\vec{x}_j^+ - \vec{\beta}.\vec{x}_{k_1}^-), s(\vec{\beta}.\vec{x}_j^+ - \vec{\beta}.\vec{x}_{k_2}^-))}{var(s(\vec{\beta}.\vec{x}_j^+ - \vec{\beta}.\vec{x}_k^-))}$$

$$= \quad \frac{var(\vec{\beta}.\vec{x}_j^+)}{var(\vec{\beta}.\vec{x}_j^+) + var(\vec{\beta}.\vec{x}_k^-)}$$

By symmetry, $\frac{V_2}{V} = \frac{var(\vec{\beta}.\vec{x}_k^-)}{var(\vec{\beta}.\vec{x}_j^+) + var(\vec{\beta}.\vec{x}_k^-)}$ so $L_e = 1$.

This leaves only the case of balanced data, with large $\|\vec{\beta}\|$. For this case, we determine the efficiency loss empirically. On several datasets used for experimentation, $L_e$ was measured for a value of $\vec{\beta}$ near the gradient descent solution. In each case it was found to lie in the range $[1.5, 2]$. This means that if we are using $R_l(\vec{\beta})$ instead of $R(\vec{\beta})$, one would need roughly 1.5 to 2 times as much data to get an estimator of the same variance. When one considers that $R_l(\vec{\beta})$ is $P$ times more efficient to calculate, we have an overall gain in computational efficiency of at least $P/2$. Since $P$ is typically at least in the hundreds, using $R_l(\vec{\beta})$ instead of $R(\vec{\beta})$ affords an enormous efficiency gain.

## 3. The RankOpt Algorithm

The various components of the RankOpt algorithm are discussed in detail in the following subsections. Pseudocode is also shown in Table 1.

### 3.1. Gradient Descent

To simplify notation, we define new random variables, $Z_i = X_i^+ - X_i^-, \ \forall i = 1 \ldots m$, the difference between the minority and majority class r.v.'s. Then using the definition of the gradient of the sigmoid from Section 2, and the definition of $R_l(\vec{\beta})$ in Eq. 2, we have

$$\frac{\partial R_l(\vec{\beta})}{\partial \beta_i} = \frac{1}{Q}\sum_{k=1}^Q s(\vec{\beta}.\vec{z}_k)(1 - s(\vec{\beta}.\vec{z}_k))z_{ik}, \quad \forall i = 1 \ldots m \tag{4}$$

Since the gradient descent algorithm is constrained to the hypersphere $\sum_i \beta_i^2 = B$, first calculate the unconstrained gradient using Eq. 4, and then calculate its component in the direction of the hypersphere surface (i.e. perpendicular to $\vec{\beta}$). Then take a small step in this direction, and rescale to move back to the surface $\sum_i \beta_i^2 = B$. Iterate until a minimum is reached. (Since $R_l(\vec{\beta})$ is symmetric, i.e. $R_l(\vec{\beta}) = 1 - R_l(-\vec{\beta})$, it is immaterial whether we talk in terms of minimisation or maximisation.) The step size, or learning rate, is increased slightly at each iteration so long as the new value of $R_l(\vec{\beta})$ is smaller than its current value. Otherwise, the learning rate is decreased until the new value of $R_l(\vec{\beta})$ that will result from taking the step will be smaller than its current value, and only then is the step actually taken. Thus we guarantee that the value of $R_l(\vec{\beta})$ on training data will decrease at every iteration.

### 3.2. Selection of Hypersphere Radius Value

If the hypersphere radius, $\sqrt{B}$, is too small, then the arguments to the sigmoid function will generally be small, and the rank statistic will be a poor estimator of the true AUC.

Alternatively, if $B$ is too large, the rank statistic approaches a sum of step functions, and the rank surface starts to contain many small regions that are nearly flat, connected by extremely steep inclines, much like steppes on a mountainside. This tradeoff has been observed by Yan et al. (2003). Such a surface is hard to do gradient descent over. To avoid the problem of choosing a value for $B$, we use the data to calculate a series of increasing $B$ values as follows.

The rank statistic is a sum of sigmoids with different arguments, where the arguments depend on the data. Define "saturation" of the sigmoid function as being when the magnitude of its argument is $> 5$. These are the almost flat extrema of the sigmoid function. (That is the sigmoid returns a value in the range $[0, \sim 0.006]$ or $[\sim 0.994, 1]$.) Then define the probability of sigmoid saturation as being the probability that for a randomly chosen observation pair, $\{\vec{x}_j^+,\ \vec{x}_k^-\}$, the sigmoid function saturates, i.e. $Pr(|\vec{\beta}.(\vec{x}_j^+ - \vec{x}_k^-)| > 5)$. Then calculate a sequence of values of $B$, such that each value of $B$ corresponds to a probability of saturation (on the training data) from the increasing series $p$. Typically $p = (0.1, 0.3, 0.5, 0.7, 0.75, 0.8, \ldots)$, and we end with a probability of saturation $\approx 0.99$. At this point $R_l(\vec{\beta})$ is very close to the true AUC.

Having generated the sequence of $B$-values, start with the smallest one, and perform gradient descent as described above until the minimum $R_l(\vec{\beta})$ is reached. Then move to the next smallest value of $B$, which will change the rank surface, and hence the position of the minimum. Continue gradient descent, starting with $\vec{\beta}$ such that its direction is the same as that of the $\vec{\beta}$ where the previous gradient descent stopped. Repeat this process iteratively, increasing $B$ at each iteration, until the sequence of $B$'s has been exhausted. This way, by the time the problem of "steppes" begins to arise, almost all of the gradient descent has already been done.

### 3.3. Local Minima

In performing gradient descent, one must contend with the possibility of local minima on the error surface. In this section it will be demonstrated that for a wide variety of datasets local minima are unlikely to play a significant role. By the standard definition of expectation

$$E[AUC(\vec{\beta})] = E[g(\sum_i \beta_i Z_i)] = E[g(\vec{\beta}.\vec{Z})] = \int g(\vec{\beta}.\vec{z})\, f(\vec{z})\, d\vec{z}$$

where $f(\vec{z})$ is the joint p.d.f. of the $Z_i$'s.

Convert cartesian to spherical co-ordinates $\mathbb{R}^m \to S^m$,

$$\vec{\beta} \to (r, \vec{\theta}); \qquad \vec{z} \to (\rho, \vec{\alpha})$$

where $\vec{\theta} = (\theta_1, \theta_2, \ldots \theta_{m-1})$ and $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots \alpha_{m-1})$ are $(m-1)$-dimensional angle vectors. The integral becomes

$$E[AUC(r, \vec{\theta})] = \int_{\vec{\alpha}} \int_{\rho} J_m(\rho, \vec{\alpha}) g(r\rho \cos(\vec{\theta} - \vec{\alpha}))\, f_s(\rho, \vec{\alpha})\, d\rho\, d\vec{\alpha}$$

$$\text{where} \qquad J_m(\rho, \vec{\alpha}) = \rho^{m-1} \prod_{l=1}^{m-2} (\sin \alpha_l)^{m-1-l}$$

is the Jacobian for the change of co-ordinates $\mathbb{R}^m \to S^m$, and $f_s(\rho, \vec{\alpha})$ is the p.d.f. of $(r, \vec{\theta})$, expressed in the $m$-dimensional spherical co-ordinate system. Since only the sign of the argument to $g()$ matters,

$$E[AUC(\vec{\theta})] = \int_{\vec{\alpha}} g(\cos(\vec{\theta} - \vec{\alpha})) \int_{\rho} J_m(\rho, \vec{\alpha})\, f_s(\rho, \vec{\alpha})\, d\rho\, d\vec{\alpha}$$

$$\text{Then denoting} \qquad h(\vec{\alpha}) = \int_{\rho} J_m(\rho, \vec{\alpha})\, f_s(\rho, \vec{\alpha})\, d\rho$$

$E[AUC(\vec{\theta})]$ can be expressed as a convolution in $m-1$ dimensions, as follows.

$$E[AUC(\vec{\theta})] = \int_{\vec{\alpha}} g(\cos(\vec{\theta} - \vec{\alpha}))\, h(\vec{\alpha})\, d\vec{\alpha}$$

$h(\vec{\alpha})$ is the marginal p.d.f. of the $(m-1)$-dimensional solid angle $\vec{\alpha}$. Since $g(cos())$ is a rectangle function, and convolution with a rectangle function has a smoothing effect which will tend to eliminate local extrema, we can assert that if $h(\vec{\alpha})$ is unimodal, then $E[AUC(\vec{\theta})]$ must be unimodal. Further, even if $h(\vec{\alpha})$ is multi-modal, the local extrema may well be eliminated by convolution with $g(cos())$.

This, however, holds for the *expected* rank error, which is what the surface $AUC(\vec{\theta})$ approaches as the amount of data approaches infinity. In reality the amount of available data is finite, so some local minima may arise due to noise - i.e. $AUC(\vec{\theta})$ fluctuates around its expected value. Given sufficient data, these fluctuations will be small, and hence should be close to the location of the expected minimum, where the gradient of $E[AUC(\vec{\theta})]$ is small. This has been verified empirically by graphing the error surface for a variety of three dimensional problems – as the amount of data increases, the local minima move closer to one another and become fewer, until there is only one.

### 3.4. Starting Point

Since local minima do not play a significant role, the selection of a starting point for gradient descent should have minimal impact on the final solution. It may however have an impact on computational efficiency. Hence it is desirable to find a starting point that is likely to be close to the final solution. Intuitively, if a predictor has a large difference in class specific means, it should be an important predictor in the model. Also if the class specific variances for a given predictor are low, it should also be an important predictor in

**Table 1: Pseudocode for RankOpt algorithm with PSF**

```
for i = 1 to # of PSF sub-folds { (Section 3.7)
    initialise saturation probability, Pr(sat) (Section 3.2)
    while (Pr(sat) < threshold) {
        calculate sphere radius, √B (Section 3.2)
        set starting point (Section 3.4)
        perform gradient descent over the hyper-surface
defined by R_{l,i-1}(β⃗), constrained to the sphere of radius
√B (Section 3.1)
        increase Pr(sat)
    }
    select β⃗ for sub-fold i and save as β⃗_i (Section 3.6)
}
calculate the final β⃗ value by averaging β⃗_i over all i
```

the model. Defining $\Delta\mu_i = E[X_i^+] - E[X_i^-]$ to be the difference in class specific means, or the "class separation" for predictor $i$, and $V_i = var(X_i^+) + var(X_i^-)$ to be the "class specific variance". we select as a starting point

$$\beta_i = \frac{\Delta\mu_i}{V_i}, \quad \forall i = 1 \ldots m$$

In practice, we linearly scale every predictor variable $x_i$ by $\Delta\mu_i/V_i$, so that in the scaled space our starting point becomes simply $\vec{\beta} = (1, 1, \ldots, 1)$.

### 3.5. Plateaus

Plotting the $R_l(\vec{\beta})$ surface for a wide variety of three dimensional datasets showed that indeed local minima only occur where they are explainable by the finite nature of the data. However the underlying minimum is often very broad, with a low gradient "plateau" around it. It is difficult to do gradient descent across these plateaus, since even a small amount of noise creates local minima. Handling these is a subject of its own, beyond the scope of this paper. For now we settle for developing a heuristic based linear rescaling, to partly eliminate plateaus. This is applied in addition to the scaling of Section 3.4.

### 3.6. Model Selection Rule

Recall that in Eq. 2 an $O(n)$ rank statistic was defined. Note that one could easily offset either the minority or majority class data relative to the other by $a$ observations, $a = 1 \ldots P-1$, yielding the equivalent statistic

$$R_{l,a}(\vec{\beta}) = \frac{1}{Q}\sum_k s\left(\vec{\beta}.(\vec{x}^+_{(k+a) \bmod P} - \vec{x}^-_k)\right)$$

$R_{l,a}(\vec{\beta})$ has the same mean and variance for all values of $a$. For any $a \neq 0$ it is clearly somewhat correlated with $R_l(\vec{\beta})$, but it nonetheless yields a convenient validation set which is at least partly independent of the training set, and comes at no extra cost (in the sense that it comes wholly from within the training data). To see exactly to what extent this validation statistic is independent, we would like to find the value of $corr(R_l(\vec{\beta}), R_{l,a}(\vec{\beta}))$, $a \neq 0$. We note that

$$\begin{aligned} R(\vec{\beta}) &= \frac{1}{P}\sum_{a=1}^{P} R_{l,a}(\vec{\beta}) \\ \Rightarrow var(R(\vec{\beta})) &= \frac{1}{P^2} var\Big(\sum_{a=1}^{P} R_{l,a}(\vec{\beta})\Big) \\ = \frac{1}{P^2}\Big(\sum_{a=1}^{P} var(R_{l,a}(\vec{\beta})) &+ \sum_{a_1=1}^{P}\sum_{a_2=1}^{P} cov(R_{l,a_1}(\vec{\beta}), R_{l,a_2}(\vec{\beta}))\Big), \end{aligned}$$

where $a_1 \neq a_2$. The covariance term has the same value for all $a_1, a_2$, so simplify the above expression as follows.

$$var(R(\vec{\beta})) = \frac{1}{P} var(R_l(\vec{\beta})) + \frac{P-1}{P} cov(R_l(\vec{\beta}), R_{l,a}(\vec{\beta}))$$

dividing by $\frac{1}{P} var(R_l(\vec{\beta}))$ and then rearranging gives

$$corr(R_l(\vec{\beta}), R_{l,a}(\vec{\beta})) = \frac{1}{P-1}\Big(\frac{P}{L_e} - 1\Big)$$

Given the range of values of $L_e$ found empirically in Section 2.2, we can safely assume that $\frac{P}{L_e} \gg 1$ so we have

$$corr(R_l(\vec{\beta}), R_{l,a}(\vec{\beta})) \approx 1/L_e$$

This gives values of $corr(R_l(\vec{\beta}), R_{l,a}(\vec{\beta}))$ in the range $[\frac{1}{2}, \frac{2}{3}]$.

Recall (Section 3.2) that training involves a series of gradient descent runs with increasing values of $B$, each one converging to its own local minimum. Which one of these minima do we select as being closest to optimal? A first guess might be that the $\vec{\beta}$ corresponding to the largest value of $B$ should be selected, but this tends to cause overtraining. So we select the $\vec{\beta}$ which minimises $R_{l,a}(\vec{\beta})$, for some arbitrarily chosen value of $a$. It was found empirically that despite the correlation between the training and validation statistics, such a selection rule usually results in a model that is very close to optimal on the test set.

### 3.7. Sub-Folds

The existence of a semi-independent rank statistic, $R_{l,a}(\vec{\beta})$, $a \neq 0$, can be used to advantage in a way that goes beyond development of a model selection rule. It can be used to supply a second, albeit correlated, training set, which can be used to augment the training itself.

One can execute two totally separate gradient descent training runs, one in which the error surface is defined by the statistic $R_l(\vec{\beta})$, and the other in which it is defined by

$R_{l,a}(\vec{\beta})$, $a \neq 0$. These yield two separate estimates of the optimal $\vec{\beta}$, which are at least partially independent. Intuitively, averaging these two estimates is likely to yield a better test result, because any error components that are in opposite directions will cancel.

We refer to this technique as pseudo sub-folding (PSF). PSF can be extended, of course, to more than two sub-folds.

## 4. Experimental Description and Results

Two sets of experiments were performed. The first tests the performance of RankOpt with various settings of PSF, namely: no PSF; PSF with two sub-folds (PSF2); and PSF with three sub-folds (PSF3). The second set of experiments compares the best of these with the other linear classifiers.

For the SVM classifiers and the centroid classifier, the scaling of the data can significantly impact the classifier's performance. Further, the SVM classifiers' performance can be significantly affected by the penalty parameter of the error term. The SVM's and the centroid classifier were therefore run with both the same scaling as RankOpt, and with no scaling at all. Further, the SVM's were run with three different values of penalty parameter – 10, $10^3$, and $10^5$. In each case the best result only is quoted.

### 4.1. Datasets

Experiments were performed on eight datasets, from different domains and of different levels of difficulty. The minority class was typically between 10% and 40% of the data.

**Forest Cover Type (forest)**: Data was downloaded from the UCI KDD repository. It classifies $30 \times 30$ metre cells of forest into one of seven cover types based on the cell's dominant tree type. The two most populous classes (Spruce-Fir and Lodgepole Pine) were extracted, and the binary classification task consisted of distinguishing between these classes. A total of 10 predictors were used, these being the 10 continuous predictors supplied for the data.

**Housing Mortgage (housing)**: Data was downloaded from the U.S. Census Bureau 5% Public Use Microdata Sample (PUMS) containing individual records of the characteristics of a 5% sample of housing units for the state of Florida. Amongst all housing units which had a mortgage, the binary classification task was to distinguish between those for which the mortgage had been paid off and those for which it hadn't. The 12 continuous or ordinal predictor variables included the total household income, the room and bedroom counts, rate costs (electricity, water and gas), the property tax rate, insurance rate and property value.

**Telecommunications Churn (churn10 and churn31)**: Data on mobile phone customers of a large telecommunications carrier was used to learn to distinguish between those that churned to a competitor in the following three months and those that didn't. After rebalancing,[1] the minority class was $\sim 40\%$ of the data. A set of 31 continuous and ordinal variables was used for prediction, including bill and product information. Further, a subset of 10 of these predictors was selected, none of which were particularly predictive, resulting in a difficult to learn task. This made up a second telecommunications binary classification task.

**Marital Status (married)**: As for the housing mortgage dataset, data was downloaded from the U.S. Census Bureau PUMS. From this dataset a 1% sample of individual records from the state of California was extracted. The binary classification task was to distinguish between individuals who have been married (whether currently married or not), with individuals who have never been married. The predictors were 11 continuous variables, including ones relating to age, education level, income, and working hours.

**Intrusion Detection (intrusion)**: This dataset consists of a random sample of the intrusion detection data used for the 1999 KDD Cup competition. The classification task was to distinguish between normal use and intrusion. The 10 predictors used were a subset of all continuous predictors available with the data, as certain continuous predictors were omitted to make the problem more challenging.

**Handwritten Digit Recognition (digit)**: Data was downloaded from the MNIST handwritten digit database. Each observation in this dataset consists of a bitmap of $28 \times 28$ grayscale values, representing a handwritten digit. Each observation was converted to lower resolution ($7 \times 7$ pixels) to reduce the dimensionality of the problem. The classification task was to distinguish between the digit '0' and all other digits. To make the problem more challenging, only the top 3 rows of pixels (21 pixels) were used. Further, pixels near the corners which contain almost no information were discarded. The result was a 17 dimensional dataset.

**Weather Season Prediction (weather)**: There is a grid of weather buoys in the equatorial region of the Pacific Ocean. These take meteorological measurements, including wind speed and direction, air and sea temperature, and humidity at regular intervals. The resulting data is available at the website of the Tropical Atmosphere Ocean project. Hourly measurements for all buoys over the period from May 1999 to April 2000 were downloaded. The classification task was to distinguish meteorological readings made during the northern hemisphere Autumn months (October, November and December) from those made in other months.

[1] The proportion of minority class observations for this dataset is very small. Hence the data was rebalanced so as to include sufficient minority class observations without using a prohibitively large amount of majority class data. The RankOpt algorithm does not require this rebalancing, but does require sufficient observations of each class for training.

### 4.2. Experimental Procedure

To lend statistical significance to our results, it is desirable to apply each classification method to a large number of independent training sets, and average the AUCs of the resulting models on the test sets. This necessitates that the training sets for multiple runs of each algorithm be mutually exclusive of one another. Hence each dataset was split into $n$ mutually exclusive folds, and in each run of each classifier, training was performed using one fold, and the resulting model was tested on the remaining $n-1$. With the exception of the digit dataset, which contains 60,000 observations, each of the datasets contains a minimum of 180,000 observations. So from each of these datasets 180,000 observations were randomly selected. These were split randomly into 60 mutually exclusive folds ($n = 60$), with 3000 observations each. 60 runs of each classification algorithm (RankOpt, linear regression, SVM-L1, SVM-L2, and centroid ) were then performed, each using a different one of the 60 folds for training (and hence validation), and the other 59 for testing, as described above. This yielded 60 test results for each classifier for each dataset. This experimental procedure was then repeated using different amounts of training data – 30 folds of 6,000 observations, $15 \times 12{,}000$, $9 \times 20{,}000$, and $6 \times 30{,}000$. The experimental procedure for the digit dataset was identical except that the number of folds was divided by 3 in each case.

### 4.3. Effect of PSF

Figure 1 shows the results of the first set of experiments, namely, measuring the effect of PSF on RankOpt's performance. We measure the mean of the AUC on the test sets (y-axis), at each training data quantity (x-axis), for all eight datasets, for all three PSF settings. Vertical bars show a one standard error confidence interval. The standard error did not vary much with training set size, hence it is shown for one training set size only. It appears that PSF has a beneficial effect for all datasets except for churn10, intrusion, and perhaps weather, where it has no significant impact. However the difference between PSF2 and PSF3 is minimal for all eight datasets. Therefore, for the purposes of comparing with other classifiers, we select RankOpt with PSF2.

### 4.4. Comparison with Other Linear Classifiers

Figure 2 shows how RankOpt with PSF2 compares with the other classifiers. The meaning of the axes is as in Figure 1. The SVM-L1 algorithm is computationally intensive, and results could not be generated in reasonable time for any more than 6000 training observations, so only these are shown. Error bars are not shown as they are mostly insignificant relative to the difference in classifier performance. For six of the eight datasets, RankOpt is a clear winner. For the forest data, linear regression is comparable to RankOpt, even outperforming it when the amount of training data is limited, and SVM-L1 performs only marginally worse than RankOpt. For the digit data, SVM-L2 performs only marginally worse than RankOpt. The simple centroid classifier is usually by far the worst, and often doesn't even make it onto the chart. Note that there is no clear "runner up" to RankOpt amongst the other classifiers, with each being far worse than RankOpt at least occasionally – linear regression fails to make it on to the chart for the housing dataset, and the SVM's fail to make it on to the chart for the weather dataset.

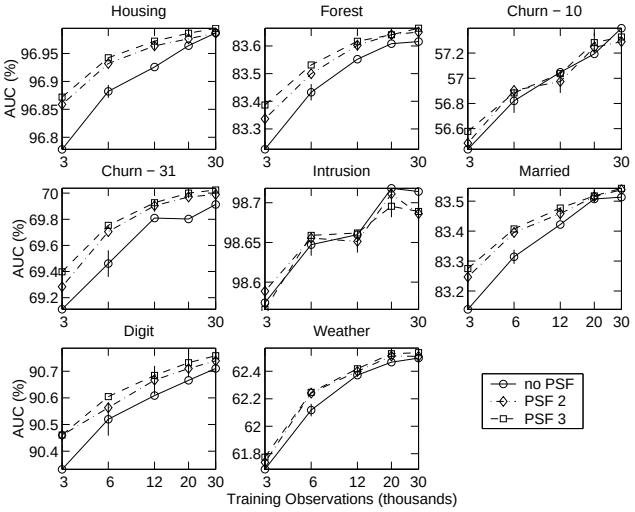


*Figure 1.* Rank statistic using RankOpt optimisation, with three different levels of PSF.

It is noteworthy that the SVM and centroid classifiers are highly sensitive to how the original data are scaled. SVM's also have the drawback that they require a penalty parameter to be set, and results can be quite sensitive to this. For the churn dataset with 31 dimensions, the linear regression package occasionally reported a warning that the matrix was ill-conditioned, and hence results may be unreliable. It is expected that this problem would arise with increasing frequency as the number of predictors increases.

In terms of computational efficiency, it is worth noting that unlike some other linear classifiers, RankOpt is linear in both the number of training observations and in the number of predictors. Since neither the RankOpt code nor the SVM code used to this point has been optimised, it would be difficult to draw any conclusions from a direct comparison of execution times of the various classifiers.

## 5. Comparison with Related Work

We note that other algorithms have been developed in which the objective function closely approximates the AUC (Yan et al., 2003; Cortes & Mohri, 2004). These differ from ours in several important ways. In particular, they yield

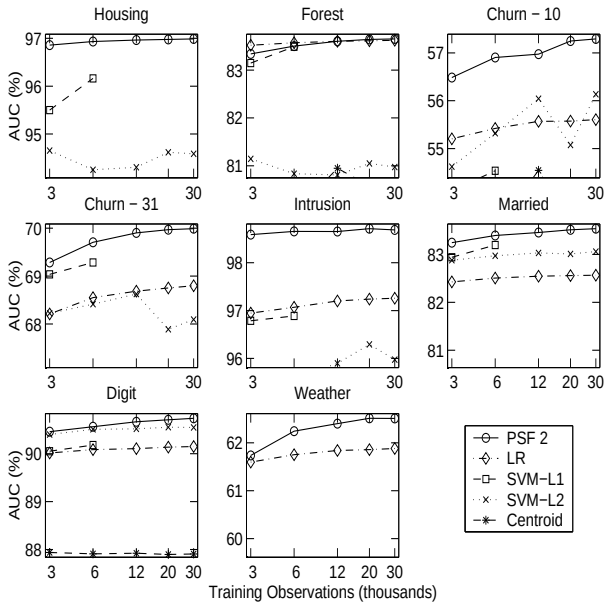


*Figure 2.* RankOpt with PSF measured against other classifiers

non-linear models, whereas RankOpt's models are strictly linear. Therefore, direct comparison between RankOpt and these other techniques would be appropriate only after extending RankOpt to non-linear space. (Section 6).

In Yan et al. (2003), the sigmoid approximation of the AUC is considered, but rejected in favour of a polynomial approximation. The claim is made that the sigmoid approximation with a small $\beta$ is not accurate enough, and with a large $\beta$ one creates too many steep gradients. Although these observations are true, we have shown that by using a series of increasing $\beta$ values, this trade off can be avoided.

Cortes & Mohri (2004) use boosted decision stumps to optimise the AUC. This method is quite different from RankOpt's gradient descent over the rank statistic surface. Comparison between their method and a non-linear extension of RankOpt would be of interest.

## 6. Conclusion and Future Work

We have introduced RankOpt, a linear binary classifier which optimises AUC. RankOpt was compared to a number of other linear binary classifiers, and in almost all cases was found to significantly outperform them.

This work has focussed on prediction tasks in which the predictors are all either continuous or ordinal. It is planned that this will be extended to include binary valued predictors, enabling the development of a non-linear classifier via binarisation of continuous and ordinal predictors.

Scaling of the data has been found to significantly affect RankOpt's performance. This is an issue that we plan to explore more thoroughly in the future.

## Acknowledgements

The permission of the Managing Director, Telstra Research Laboratories (TRL) to publish this paper is gratefully acknowledged. The technical advice and feedback of Herman Ferra, TRL, is gratefully acknowledged.

## References

Bamber, D. (1975). The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *J. Math. Psych., 12*, 387 – 415.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine laerning algorithms. *Pattern Recognition, 30(7)*, 1145 – 1159.

Cortes, C., & Mohri, M. (2004). AUC Optimization vs. Error Rate Minimization. In *Advances in neural information processing systems 16*. Cambridge MA: MIT Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the Tenth European Conference on Machine Learning ECML98*.

Mann, H. B., & Whitney, D. R. (1947). On a test whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist., 18*, 50 – 60.

Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree Induction vs. Logistic Regression: A Learning Curve Analysis. *Journal of Machine Learning Research, 4*, 211–255.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization.

Rocchio, J. J. (1971). Relevance Feedback in Information Retrieval. *The Smart System - Experiments in Automatic Document Processing* (pp. 313–323). Englewood Cliffs, N J: Prentice-Hall Inc.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Weiss, G. M., & Provost, F. (2001). *The effect of class distribution on classifier learning: an empirical study* (Technical Report). Rutgers University.

Yan, L., Dodier, R., Mozer, M. C., & Wolniewicz, R. (2003). Optimizing classifier performance via approximation to the wilcoxon-mann-witney statistic. *Proceedings of the Twentieth Intl. Conf. on Machine Learning* (pp. 848 – 855). AAAI Press, Menlo Park, CA.