David Andrzejewski
CS 726 Final Project
Report on: Second Order Cone Programming Approaches for Handling Missing and Uncertain Data

Introduction

This paper [4] addresses the problem of doing regression or classification on patterns (x, y) in cases where components of x may be either uncertain, or missing entirely. The classification label or output value y is always still assumed to be known exactly.

This paper presents a set of techniques and error measures for dealing with both uncertain and missing x values in both the regression and classification tasks. The only necessary assumptions are that each component of x have finite mean and covariance, and that these can be obtained somehow (estimated from the other data points, for example). This extends [1], which assumed Gaussian x.

Background

Uncertain or missing observations can occur in many (if not all) practical applications of machine learning techniques. A principled strategy for dealing with this fact would therefore be very useful.

The traditional approach to dealing with missing values in x is to either discard those examples or perform imputation, where estimates are used to fill in the missing values.

In statistics, there seems to be a fairly well developed field of techniques and algorithms for performing this imputation and adjusting the rest of the analysis to take the imputation into account. This paper does not make any mention of this, and only asserts that their formulations outperform imputation in their experiments. This may be because much of the work on imputation in statistics seems to be focused on using imputation to do statistical inference, as opposed to the machine learning applications handled in this paper.

However, there was one idea from the statistics field which I found to be possibly quite relevant to the machine learning task. It consists of a warning against simply deleting or ignoring examples with missing values, on the grounds that those examples may not be representative of the dataset as a whole [2]. This means that you would be systematically underrepresenting some region of example space. In the machine learning context, it is clear that this would hurt the generalization performance of your learner and provides a strong argument for using a principled method to handle examples with missing values, rather than simply discarding them.

Classification with uncertainty

The classification problem is originally formulated as a standard linear classifier with slack variables and an ℓ_2-norm regularization term on the weights. The quadratic program is then generalized into an equivalent second-order cone program. The problem is then modified by making x a random variable and bounding the probability of error exceeding the slack by a user-defined probability κ.

$$Pr_{x_i}\{y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i\} \geq 1 - \kappa_i \tag{1}$$

This probability bound can be seen as somewhat similar to the machine learning concept of Probably Approximately Correct (PAC) theory.

Multivariate Chebyshev Inequality

One of the key results used in this paper is a multivariate extension of the Chebyshev inequality [3]. The original Chebyshev inequality uses the mean and variance of a random variable to give us a bound on the probability that the random variable strays from its mean by at least some value. A multivariate extension result then has obvious utility for their formulation of classification under uncertainty. This is also where the first and second-order moments of each component of x come into play. The multivariate bound always holds for all distributions with the same second order moments, and only achieves equality for the 'worst' distribution in this family.

Robust and normal formulations

The multivariate Chebyshev inequality is then used to derive two different formulations for classification under uncertainty, both of which reduce to the same SOCP and differ by the 'robustness' parameter γ.

The robust formulation assumes that x is drawn from the 'worst-case' distribution with respect to the multivariate Chebyshev bound. This means that the classifier will achieve the desired probability bound (or better), no matter what the true underlying distribution is.

The normal formulation simply assumes that the x is drawn from a Gaussian distribution with the given mean and variance.

For both formulations, the authors derive a SOCP with the modified constraint

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i + \gamma_i \|\Sigma_i^{1/2} w\| \tag{2}$$

γ is calculated as a function of the desired probability bound κ. The exact function differs for the normal case vs. the robust case, with the γ value for a given κ being strictly lower for the normal case. This clearly makes sense, as by assuming a Gaussian distribution you need less 'cushion' than if you were to assume the worst-case distribution.

Both formulations have an interesting geometric interpretation which the authors derive. Each data point x can be expanded into an 'uncertainty ellipsoid' centered at $\overline{x}$ with radius γ and metric Σ. This is intuitively reasonable in the sense that requiring a better probability bound κ causes the uncertainty ellipsoid to grow. It also means that the uncertainty ellipsoid will be 'stretched' further in directions corresponding to high-variance components of x.

The classification under uncertainty constraint then corresponds to requiring that all points within the uncertainty ellipsoid lie on the correct side of the classification hyperplane (or are pushed to the correct side by slack).

This geometric interpretation gives us two error measures for testset evaluation. Worst-case error is 1 if any part of the uncertainty ellipsoid lies on the wrong side, and expected error is calculated as the fraction of the volume of the uncertainty ellipsoid which lies on the wrong side. In practice this would be calculated by sampling uniformly from the ellipsoid and counting how many samples are on the correct side.

Regression with uncertainty

The regression problem is developed in the context of a standard linear regression problem where the goal is to minimize the regularized risk with respect to some loss function. Two different formulations for robust regression are proposed: 'close to mean' (CTM) 'small residual' (SR). CTM puts a probability bound on the deviation from the error at $\overline{x}$

$$Pr_x\{|e(f(x), y) - e(f(\overline{x}), y)| \geq \theta\} \leq \eta \tag{3}$$

This bound should encourage overall smoothness of the regression estimate with respect to deviations from $\overline{x}$. On the other hand, the SR formulation only penalizes deviations that increase the residuals with respect to the true target output value y

$$Pr_x\{|e(f(x), y)| \geq \xi + \epsilon\} \leq \eta \tag{4}$$

The Chebyshev and Markov inequalities are then used to derive sufficient conditions for the CTM and SR requirements. These sufficient conditions can then be added as new constraints to the regression optimization problems. The CTM sufficient condition is

$$\|\Sigma^{1/2} w\| \leq \theta\sqrt{\eta} \tag{5}$$

The SR sufficient condition is

$$\sqrt{w^T \Sigma w + (\langle w, \overline{x} \rangle + b - y)^2} \leq (\xi + \epsilon)\sqrt{\eta} \tag{6}$$

These two conditions are used to formulate two separate regression optimization problems, one for CTM and one for SR. It is interesting to note that these conditions are never combined, nor is the possibility discussed. It

may be that the conditions could be incompatible, would make the problem too difficult, or that there would be little to gain by combining them.

Like classification under uncertainty, both the SR and CTM regression formulations have geometric interpretations with uncertainty ellipsoids. Likewise, they also have similar error measures: robustness, expected, and worst-case. Robustness error is the largest difference between the estimate at $\overline{x}$ and any x in the uncertainty ellipsoid and is applicable to CTM. Expected error is the square root of the expectation of the square of the residual and is applicable to SR. The worst-case error is the maximum residual for any x in the uncertainty ellipsoid, which makes it applicable for both CTM and SR.

Classification with missing values

The authors' approach to missing values still does use a form of imputation, albeit a somewhat more sophisticated version than simply imputing values based on the entire dataset. The approach consists of estimating separate x component means and covariances for each class. It is important to note that these values are estimated for all components of x, whether they have missing values or not. These estimates are then used to impute the missing x values, which are then used to re-estimate the means and covariances. This Expectation Maximization (EM) procedure is then iterated until convergence. The imputation itself uses a linear model with means for missing and observed values, and a covariance matrix for all values. The expectation of the missing values is then calculated from the estimated mean plus the covariance effects of the non-missing values.

The standard training optimization problem is then formulated to train the classifier, with the addition of the modified constraint (2) for examples with imputed values. For these training examples, the mean and covariance corresponding to the example's true class label are used.

For prediction on examples with missing values, we must decide which class' mean and covariance values to use for imputation. This is accomplished by doing imputation for each class, then seeing which results is furthest from the hyperplane. The mean and covariance associated with that class are then used to impute the missing values, and the point is then classified.

It seems like it is possible for the furthest point from the hyperplane to be on the 'wrong' side of the hyperplane, where 'wrong' means the side not corresponding to the class of the mean and covariance used for imputation. This would be a very strange result but I do not see why it is impossible, and the authors do not mention it.

Regression with missing values

Regression with missing values is somewhat simpler than classification. Only one set of mean and covariance values are estimated. These are used to impute the missing values as described above. The regression optimization problem is then set up in the standard way, except with additional CTM or SR constraints for the examples with imputed values.

Kernelized formulations

The kernelized formulation corresponds to doing SVM or SVR for the case where some examples have missing values. The key aspect of the derivation of the dual is that the weight vector is expressed in terms of example with no missing values and points from the uncertainty ellipsoids of examples with missing values. The resulting non-linear formulation has a form almost identical to the linear case, except with α weights in the place of w and $\tilde{K}(x_i)$ in the place of x_i. $\tilde{K}(x_i)$ is the vector of kernel function values for x_i and every other x (both full examples and ones with imputed values).

The robust formulation now requires values for the mean and covariance of $\tilde{K}(x_i)$. Estimating the covariance may be tricky, and the authors suggest the two possible approaches. One is to make the simplified assumption of spherical uncertainty in feature space (an identity covariance matrix), and the other is to use the $\tilde{K}(x)$ values of the k nearest neighbors in example (that is, x) space to estimate the covariance of a given $\tilde{K}(x)$.

Once these aspects of the formulation have been addressed, both the classification and regression tasks are pretty much identical to the linear case with the addition of robust classification constraints (2) for classification or CTM or SR constraints for regression. Again, these additional constraints only apply to examples with imputed values.

Experimental results

Most of the experiments consist of taking a dataset, randomly deleting values, and then comparing the robust approach to a nominal classifier which simply uses imputed values estimated by linear regression. A 'control' nominal classifier is also run on the full dataset (no missing values). The authors also make it a point to use a toy dataset which is not linearly separable to show the need for the kernelized formulation.

The experimental results (unsurprisingly) validate the effectiveness of the authors' approach. Their robust formulation outperforms traditional imputation, when suitable γ values are chosen (see below). In fact, the robust formulation is often able to approach the accuracy of the 'control' nominal classifier. Also, the kernelized version outperforms the linear version on nonlinear datasets.

One interesting aspect of the experimental results is that the performance advantage of the robust formulations often does not appear until the robustness parameter γ is increased beyond a certain value. This seems to show that the robustness formulation may not be advantageous for lower values of κ.

It is also interesting to note that the simpler 'uncertain' formulation was not used in any experiments. All example feature values were either totally correct, or missing. While the strategy for addressing missing values used the uncertain formulation, it would have also been interesting to examine the performance of the robust classifier in the presence of various types of uncertainty in x, for example Gaussian noise with mean zero, or systematic bias modeled as Gaussian noise with a non-zero mean.

Questions, issues, and discussion

After reading the paper, a number of questions and issues occurred to me.

- They never directly address this issue, but presumably the first and second order moments for each x_i are estimated from the rest of the training data. If the examples with missing values seem to be a non-representative subset of all examples, could this lead to bad estimates? Should complete examples which are more similar in some sense (nearest neighbors ignoring directions corresponding to missing values, or same label) be weighted more heavily for mean and covariance estimation?
- What happens in the case of overlapping uncertainty ellipsoids? With different labels? Will this result in an infeasible optimization problem?
- Why not combine the CTM and SR constraints? Are they overlapping,redundant, or conflicting in some sense?
- What is a good way to choose γ (or κ)? In the experiments larger γ values seemed to lead to more robust classification (up to a point). But if γ is too large can an infeasible optimization problem result?
- The schemes for estimating $\tilde{K}(x_i)$ do not seem very principled. Are there better general techniques, or a kernel-dependent method for doing this?

It seems that this approach has an interesting interpretation with respect to maximum-margin classification techniques. The formulation presented in this paper takes advantage of the fact that all directions are not equally important when trying to maximize the margin. That is, you may be willing to trade off a slightly smaller margin overall in order to get a wider margin in the direction of a high-variance x component. Figure 2 in the paper is an excellent illustration of this idea.

The core insight of this paper is that robustness in the results can be acheived by integrating information about the underlying x distribution directly into the optimization formulation in the form of modified constratiants. The resulting formulation can (very roughly) be thought of as 'uncertainty normalized' or 'uncertainty rescaled' with respect to the variance of different components of x.

Overall, this approach seems to be an effective and principled way to build missing or uncertain x values directly into the optimization problem formulation. The geometric interpretations provide a very intuitive way to understand the formulations, and the probability bounds and error functions provide a meaningful framework in which to understand the problem and the approach. Also, the requirement of being able to compute the first two moments of the x distribution is pretty minimal and should allow this approach to be applied to a wide variety of problem settings.

References

[1] C. Bhattacharyya, K. S. Pannagadatta, and A. J. Smola.
A second order cone programming formulation for classifying missing data.
Advances in Neural Information Processing Systems (NIPS 17), 2004b.

[2] Little, R. J. A. and D. B. Rubin (1987).
Statistical analysis with missing data.
John Wiley and Sons, New York.

[3] A.W.Marshall and I. Olkin.
Multivariate Chebyshev Inequalities.
Annals of Mathematical Statistics, 31(4):10011014, 1960.

[4] Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, Alexander J. Smola.
Second Order Cone Programming Approaches for Handling Missing and Uncertain Data.
JMLR (Special Topic on Machine Learning and Optimization) 7(Jul):1283–1314, 2006.