Max-margin classification of data with absent features

Gal Chechik GAL@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford CA, 94305

Geremy Heitz GAHEITZ@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford CA, 94305

Gal Elidan GALEL@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford CA, 94305

Pieter Abbeel PABBEEL@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford CA, 94305

Daphne Koller KOLLER@AI.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford CA, 94305

Editor: Nello Cristianini

Abstract

We consider the problem of learning classifiers in structured domains, where some objects have a subset of features that are inherently absent due to complex relationships between the features. Unlike the case where a feature exists but its value is not observed, here we focus on the case where a feature may not even exist (structurally absent) for some of the samples. The common approach for handling missing features in discriminative models is to first complete their unknown values, and then use a standard classification procedure over the completed data. This paper focuses on features that are known to be non-existing, rather than have an unknown value. We show how incomplete data can be classified *directly* without any completion of the missing features using a max-margin learning framework. We formulate an objective function, based on the geometric interpretation of the margin, that aims to maximize the margin of each sample in its own relevant subspace. In this formulation, the linearly separable case can be transformed into a second order cone program (SOCP), a convex problem that can be solved efficiently. We also describe two approaches for optimizing the general case: an approximation that can be solved as a standard quadratic program (QP) and an iterative approach for solving the exact problem. By avoiding the pre-processing phase in which the data is completed, both of these approaches could offer considerable computational savings. More importantly, we show that the elegant handling of missing values by our approach allows it to both outperform other methods when the missing values have non-trivial structure, and be competitive with other methods when the values are missing at random. We demonstrate our results on several standard benchmarks and two real-world problems: edge prediction in metabolic pathways, and automobile detection in natural images.

1. Introduction

In the traditional formulation of supervised learning, data instances are viewed as vectors of features in some high-dimensional space. However, in many real-world tasks, data instances have a complex pattern of missing features. Sometimes features are missing due to measurement noise or corruption, but in other cases samples have varying subsets of observable features due to the *inherent* properties of the instances.

Examples for such settings can be found in many domains. In visual object recognition, an object is often classified using a set of image patches corresponding to parts of the object (e.g., the license plate for cars); but some images may not contain all parts, simply because a specific instance does not have the part in the first place. In other scenarios, some features may not even be defined for some instances. Such cases arise when we are learning to predict attributes of interacting objects that are organized based on a known graph structure. For example, we might wish to classify the attributes of a web-page given the attributes of neighboring web-pages (Jensen and Neville, 2002), or characteristics of a person given properties of its family members. In analyzing genomic data, we may wish to predict the edge connectivity in networks of interacting proteins (Jaimovich et al., 2005) or predict networks of chemical reactions involving enzymes and other metabolic compounds (Kharchenko et al., 2003, Vert and Yamanishi, 2004). In these cases, the local neighborhood of an instance in the graph can vary drastically, and it has already been observed that this could introduce bias (Jensen and Neville, 2002). In the web-page task, for instance, one useful feature of a given page may be the most common topic of other sites that point to it. If this particular page has no such parents, however, the feature is meaningless, and should be considered *structurally absent*.

Common methods for classification with missing features assume that the features exist but their values are unknown. The approach usually taken is a two phase approach known as *imputation*. First the values of the missing attributes are filled in during a preprocessing phase, followed by the application of a standard classifier to the completed data (Little and Rubin, 1987, Roth, 1994). Imputation makes most sense when the features are known to exist, but their values are missing due to noise, especially in the setting of *missing at random* (when the missingness pattern is conditionally independent of the unobserved features given the observations), or *missing completely at random* (when it is independent of both observed and unobserved measurements).

In the common practice of imputation application, missing attributes in continuous data are often filled with zeros, with the average of all of the data instances, or using the *k nearest neighbors* (kNN) of each instance to find a plausible value of its missing features (in the limit of large k this becomes equivalent to using the mean of all the data).

A second family of imputation methods builds probabilistic generative models of the features (with or without the label) using raw maximum likelihood or algorithms such as expectation maximization (EM) to find the most probable completion (Ghahramani and Jordan, 1994). Such model-based methods allow the designer to introduce prior knowledge about the distribution of features, and are extremely useful when such knowledge can be explicitly modeled (Kapoor, 2006). These methods have been shown to work very well for *missing at random* (MAR) data settings, because they assume that the missing features are generated by the same model that generates the observed features. The distribution over missing features, can also be approximated using *multiple imputations*, where several missing values are filled in. This can be efficiently achieved using Markov-chain monte-carlo (MCMC) methods (Schafer, 1997). The imputed datasets can then be combined for classification using a Bayesian approach or ensemble methods (Schafer, 1997).

However, model-based approaches (using EM or MCMC, for instance) can be computationally expensive, and require significant prior knowledge about the domain. More importantly, they may produce meaningless completions for features that are *structurally absent*, which will likely decrease classification performance. As an extreme example, consider the case of two subpopulation of instances (e.g., animals and buildings) with few or no overlapping features (e.g., body parts, and architectural aspects). In such a case filling missing values (e.g., the body parts of buildings) is clearly meaningless and will not improve classification performance.

The core observation made in this paper is that, in some cases, features are known to be *non-existing*, rather than *have an unknown value*. In these cases it would be useful to avoid guessing values for hypothetical undefined features. This distinction motivates the development of different approaches for handling non-existing features.

The approach we propose here aims to classify data instances directly without filling hypothetical missing values, and is therefore different from imputation. Our interpretation is that each data instance reside in a lower dimensional subspace of the feature space, determined by its own existing features. To formulate the classification problem we go back to the geometric intuitions underlying max-margin classification: We try to maximize the worst-case margin of the separating hyperplane, while measuring the margin of each data instance *in its own lower-dimensional subspace*.

We show how the linearly separable case can be formalized as a *second order cone programming* (SOCP) problem, which is convex and can be solved efficiently. For the non-separable case, the objective turns out to be non-convex, and we propose an iterative procedure that in practice converges to a fixed point of this objective.

We evaluate the performance of our approach both in two real world applications with non-existing features, and in a set of standard benchmarks with randomly removed features. Although the intuitions underlying our method arise from cases with non-existing features, we show that the performance of our methods is not inferior to other simple imputation methods, even in the case of missing at random. We then evaluate our approach on the task of predicting missing enzymes in genetic pathways and detecting automobiles in natural images. In both of these domains, features may be inherently absent due to some or all of the mechanisms described above. Our approach significantly outperforms simple imputation techniques in these settings.

2. Max-Margin Formulation for Missing Features

We start by formalizing the problem of classification with non-existing features, and then explain why standard max-margin classification must be modified to handle missingness.

We are given a set of n labeled samples $\mathbf{x}_1 \dots \mathbf{x}_n$, each characterized by a subset of features from a full set $\mathcal{F}$ of size d. Let $\mathcal{F}_i$ denote the set of features of the i^{th} sample. A sample that has all features $\mathcal{F}_i = \mathcal{F}$, can be viewed as a vector in $\mathbb{R}^d$, where the j^{th} coordinate corresponds to the j^{th} feature. Each sample $\mathbf{x}_i$ can therefore be viewed as embedded in the relevant subspace $\mathbb{R}^{|\mathcal{F}_i|} \subseteq \mathbb{R}^d$. For simplicity of notation, sample $\mathbf{x}_i$ also has a binary class label $y_i \in \{-1, 1\}$. Importantly, since instances share features, the classifier we learn should have parameters that are consistent across instances, even if those instance do not lie in the same subspace.

We address the problem of finding an optimal classifier, within the max-margin framework. In the classical SVM approach of (Vapnik, 1995, Schölkopf and Smola, 2002), we learn a linear classifier $\mathbf{w}$ by maximizing the margin, defined as $\rho \equiv \min_i y_i(\mathbf{w}\mathbf{x}_i + b)/\|\mathbf{w}\|$. In the standard setting, when each sample has all features (i.e., $\mathcal{F}_i = \mathcal{F}$ for each instance), the learning of such a classifier reduces to the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{w},\xi,b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.t. \quad & y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1 - \xi_i \ , \quad i = 1 \ldots n \end{aligned} \tag{1}$$

where b is a threshold, the ξ's are slack variables necessary for the case when the training instances are not linearly separable, and C is the trade-off between accuracy and model complexity. Eq. (1) defines a quadratic programming problem, which can be solved very efficiently, and can be extended to nonlinear classifiers using kernels (Schölkopf and Smola, 2002).

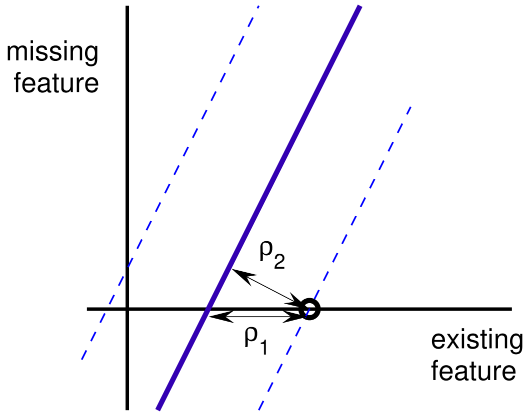

Figure 1: The margin is incorrectly scaled when a sample that has missing features is treated as if the missing feature has a value of zero. In this example, the margin of a sample that only has one feature (the x dimension) is measured both in the higher dimensional space (ρ_2) and the lower one (ρ_1). The lower dimensional margin is larger (unless the classifier is orthogonal to the sample's second feature), and therefore underestimates the margin, giving it too much weight.

Consider now learning such a classifier in the presence of missing data. At first glance, it may appear that since the $\mathbf{x}$'s only affect the optimization through inner products with $\mathbf{w}$, missing features can merely be skipped (or equivalently, replaced with zeros), thereby preserving the values of the inner product. However, this does not properly normalize the different entries in $\mathbf{w}$, and damages classification accuracy. The reason is illustrated in Fig. 1 where a single sample in $\mathbb{R}^2$ has one existing and one missing feature. Due to the missing feature, measuring the margin in the full space ρ_2, underestimates the correct geometric margin of the sample in the sub space ρ_1. In the next sections, we explore how Eq. (1) can be solved while properly taking this effect into account.

Geometric margins

The original derivation of the SVM classifier (Vapnik, 1995) is motivated by the goal of finding a hyperplane that maximally separates the positive examples from the negative, as measured by the geometric margin $\rho^{full}(\mathbf{w}) = \min_i \frac{y_i \mathbf{w}\mathbf{x}_i}{\|\mathbf{w}\|}$. In the case of missing features, each instance resides in a subspace of the joint feature space, therefore the margin $\frac{y_i \mathbf{w}\mathbf{x}_i}{\|\mathbf{w}\|}$ which measures the distance to a hyperplane in the full joint space is not well defined.

To address this problem we treat the margin of each instance in its own relevant subspace. We define the *instance margin* $\rho_i(\mathbf{w})$ for the i^{th} instance as

$$\rho_i(\mathbf{w}) = \frac{y_i \mathbf{w}^{(i)} \mathbf{x}_i}{\|\mathbf{w}^{(i)}\|} \tag{2}$$

where $\mathbf{w}^{(i)}$ is a vector obtained by taking the entries of $\mathbf{w}$ that are relevant for $\mathbf{x}_i$, namely those for which the sample $\mathbf{x}_i$ has valid features. Its norm is $\|\mathbf{w}^{(i)}\| = \sqrt{\sum_{k: f_k \in \mathcal{F}_i} w_k^2}$. We now consider our new geometric margin to be the minimum over all instance margins, $\rho(\mathbf{w}) = \min_i \rho_i(\mathbf{w})$, and arrive at a new optimization problem for the missing-features case

$$\max_{\mathbf{w}} \left(\min_i \frac{y_i \mathbf{w}^{(i)} \mathbf{x}_i}{\|\mathbf{w}^{(i)}\|} \right). \tag{3}$$

To solve this optimization problem, consider first the case of where no features are missing, as in standard SVM. There, maximizing the margin $\rho^{full}(\mathbf{w})$,

$$\max_{\mathbf{w}} \rho^{full}(\mathbf{w}) = \max_{\mathbf{w}} \left(\min_i \frac{y_i \mathbf{w}\mathbf{x}_i}{\|\mathbf{w}\|} \right) \tag{4}$$

is transformed into the quadratic programming problem of Eq. (1) in two steps. First, $\|\mathbf{w}\|$, which does not depend on the instance, can be taken out of the minimization, to produce $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|} \left(\min_i y_i \mathbf{w}\mathbf{x}_i\right)$. Then, the following invariance is used: for every solution $\mathbf{w}$, there exists a solution that achieves the same target function value, but with a margin that equals 1. This allows us to rewrite the above problem as a constrained optimization problem over $\mathbf{w}$

$$\begin{array}{ll} \max_{\mathbf{w}} & \dfrac{1}{\|\mathbf{w}\|} \\ s.t. & y_i(\mathbf{w}\mathbf{x}_i) \geq 1 \end{array} \quad , \tag{5}$$

which is equivalent to the problem of minimizing $\|\mathbf{w}\|^2$ with the same constraints, resulting in the slack-free version of the standard SVM learning optimization problem of Eq. (1).

Unfortunately, in the case of missing features, since different margin terms are normalized by different norms $\|\mathbf{w}^{(i)}\|$ in (Eq. (3)), the denominator can no longer be taken out of the minimization. In addition, each of the terms $y_i \mathbf{w}^{(i)} \mathbf{x}_i / \|\mathbf{w}^{(i)}\|$ is non-convex in $\mathbf{w}$, which makes it difficult to solve exactly in a direct way. The next section discusses how the geometric margin of Eq. (3) can be optimized.

3. Algorithms

We investigate three approaches for maximizing the geometric margin of Eq. (3). First, we show how the linearly-separable case can be rewritten as a series of convex optimization problems, and discuss the difficulties in extending this formulation to the non-separable case. Then, we describe two methods for solving the general case. The first uses a convex formulation to approximate the problem; the second uses a non-convex optimization formulation that we optimize iteratively.

3.1 A convex formulation for the linearly separable case

In the linearly separable case, we now show how to transform the optimization problem of Eq. (3) transformed into a series of convex optimization problem, in three steps. First, instead of maximizing $\left(\min_i \frac{y_i \mathbf{w}^{(i)}\mathbf{x}_i}{\|\mathbf{w}^{(i)}\|}\right)$ over $\mathbf{w}$, we maximize a lower bound

$$\begin{aligned} \max_{\mathbf{w},\gamma} \quad & \gamma \\ s.t. \quad & \min_i \frac{y_i \mathbf{w}^{(i)}\mathbf{x}_i}{\|\mathbf{w}^{(i)}\|} > \gamma \end{aligned}$$

This takes the minimization term out of the objective function into the constraints. It is equivalent to Eq. (3) since the bound γ can always be maximized until it is perfectly tight.

As a second step, the single constraint that requires that the minimum is larger than γ, is replaced by multiple constraints, one for each term in the minimization

$$\begin{aligned} \max_{\mathbf{w},\gamma} \quad & \gamma \\ s.t. \quad & \frac{y_i \mathbf{w}^{(i)}\mathbf{x}_i}{\|\mathbf{w}^{(i)}\|} \geq \gamma \qquad i = 1 \dots n. \end{aligned}$$

Finally, we write

$$\begin{aligned} \max_{\mathbf{w},\gamma} \quad & \gamma \\ s.t. \quad & y_i \mathbf{w}^{(i)}\mathbf{x}_i \geq \gamma \|\mathbf{w}^{(i)}\| \qquad i = 1 \dots n \quad , \end{aligned} \tag{6}$$

To see how this could be solved efficiently, assume first that γ is given. For any fixed value of γ, this problem obeys the general structure of a problem known as *second order cone program* (SOCP) (Lobo et al., 1998). An SOCP problem is convex, and can be solved efficiently even for large scale problems.

To find the optimal value of γ, note that Eq. (6) is a feasibility problem: we need to find the maximal γ value that still allows for a feasible solution. We therefore can solve Eq. (6) efficiently by doing a bisection search over $\gamma \in \mathbb{R}^+$, where in each iteration we solve an SOCP feasibility problem, for which standard and efficient solvers exist.

One problem with this formulation, is that any rescaled version of a solution is also a solution. This is because each constraint is invariant to a rescaling of $\mathbf{w}$. As a result, the null case $\mathbf{w} \equiv 0$, is always a solution, and is usually the solution that is found by the standard solvers. Note that adding a constraint that enforces a non vanishing norm $\|\mathbf{w}\| > const$ is no longer convex. However, adding a constraint on a single entry in w, say $w_k > const$, preserves the convexity of the problem. To ensure that the solution is not missed by the additional constraint, we solve the SOCP twice for each entry of $\mathbf{w}$, once while adding the constraint $w_k > 1$, and once with $w_k < -1$.

Unfortunately, extending this convex formulation to the non-separable case is difficult. When adding slack variables ξ's, we obtain

$$\begin{aligned} \max_{\mathbf{w},\gamma} \quad & \gamma + C\sum_{i=1}^{n}\xi_i \\ s.t. \quad & y_i\mathbf{w}^{(i)}\mathbf{x}_i \geq (\gamma - \xi_i)\,\|\mathbf{w}^{(i)}\| \qquad i = 1\ldots n. \end{aligned} \tag{7}$$

which is not jointly convex in $\mathbf{w}$ and ξ. We could however, solve a related problem, where the slack variables are not normalized by $\|\mathbf{w}^{(i)}\|$

$$\begin{aligned} & \max_{\mathbf{w},\gamma} \quad \gamma - C\sum_{i=1}^{n}\xi_i \\ & \text{such that: } y_i\mathbf{w}^{(i)}\mathbf{x}_i \geq \gamma\|\mathbf{w}^{(i)}\| - \xi_i \qquad i = 1\ldots n. \end{aligned} \tag{8}$$

Unfortunately, the problem of vanishing solutions ($\mathbf{w} \equiv 0$), discussed in the separable case, is also encountered here. Unlike the separable case, using constraints of the form $w_k > 1$, we can no longer guarantee that the solutions of the different modified problems will coincide. Indeed, we found empirically that different solutions are obtained for each modified problem. As a result, the non-separable formulation above is not likely to be of practical use.

3.2 Average norm

We now consider an alternative solution to the classification problem, based on an approximation of the margin Eq. (3). While applying standard SVM with zero-filled values can be viewed as an approximation that ignores any instance-specific components of the margin, we provide a finer approximation. We keep instance-specific numerators of the instance margin $\|\mathbf{w}\|$ and approximate only the denominator by a common term that does not depend on the instance. This allows us to take the denominator out of the minimization part of the objective. While this is only an approximation, it also has an interesting interpretation as a parameter-sharing approach.

In particular, we explore the effects of replacing each of the low-dimensional norms by the root-mean-square norm over all instances, $\overline{\|\mathbf{w}\|} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{w}^{(i)}\|^2}$, yielding

$$\max_{\mathbf{w}} \min_{i} \frac{y_i\mathbf{w}^{(i)}\mathbf{x}_i}{\overline{\|\mathbf{w}\|}} \ . \tag{9}$$

When all samples have all features, this reduces to the original problem Eq. (4). However, even with many missing features, the denominator will also be a good approximation of the denominator in Eq. (4) if all the norm terms are equal. In this case, the approximated problem differs from the exact problem by a constant factor. We therefore expect that in cases where the norms are near equal, we will find nearly optimal solutions of the original problem. As we show below, the advantage of this approximation is that it yields a tractable optimization problem.

We now follow similar steps to those of equations (4)-(5), and obtain the minimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|\mathbf{w}^{(i)}\|^2 \\ s.t. \quad & y_i(\mathbf{w}^{(i)}\mathbf{x}_i) \geq 1 \ , i = 1 \ldots n \end{aligned}$$

Adding the threshold b and slack variables ξ_i to handle the non-separable cases, we obtain

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|\mathbf{w}^{(i)}\|^2 + C\xi_i\right) \\ s.t. \quad & y_i(\mathbf{w}^{(i)}\mathbf{x}_i + b) \geq 1 - \xi_i \ , i = 1 \ldots n \end{aligned} \tag{10}$$

which is, as expected, a quadratic programming problem that can be solved using the same techniques as standard SVM.

Interestingly, this optimization problem can also be interpreted as an approach for sharing a classifier across multiple tasks. In this interpretation, we seek to solve multiple classification problems: one for each sample, but the parameters of all classifiers are shared.

3.3 Instance-specific margins

The global approach described in the last section is not expected to perform well if the individual norms $\|\mathbf{w}^{(i)}\|$'s vary strongly. A second approach for solving Eq. (3) is to treat each instance individually by representing each of the norms $\|\mathbf{w}^{(i)}\|$ as a scaling of the full norm by defining scaling coefficients $s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\|$, and rewriting Eq. (3) as

$$\max_{\mathbf{w}}\left(\min_i \frac{y_i\mathbf{w}^{(i)}\mathbf{x}_i}{s_i\|\mathbf{w}\|}\right) = \max_{\mathbf{w}}\frac{1}{\|\mathbf{w}\|}\left(\min_i \frac{y_i\mathbf{w}^{(i)}\mathbf{x}_i}{s_i}\right) \ , \quad s_i = \frac{\|\mathbf{w}^{(i)}\|}{\|\mathbf{w}\|}. \tag{11}$$

This allows us to use again the invariance to $\|\mathbf{w}\|$ and rewrite the optimization as a constrained optimization problem over s_i and $\mathbf{w}$. Eq. (11) therefore translates into

$$\begin{aligned} \min_{\mathbf{w},\mathbf{s}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ s.t. \quad & \frac{1}{s_i}y_i\mathbf{w}^{(i)}\mathbf{x}_i \geq 1 \quad \forall i \\ & s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\| \quad \forall i \end{aligned} \tag{12}$$

and in the non-separable case

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\mathbf{s}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ s.t. \quad & \frac{1}{s_i}\left(y_i(\mathbf{w}^{(i)}\mathbf{x}_i + b)\right) \geq 1 - \xi_i \ , \quad i = 1 \ldots n \\ & s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\| \ , \quad i = 1 \ldots n \end{aligned} \tag{13}$$

This constrained optimization problem is no longer a QP. In fact, due to the normalization constraints it is not even convex in $\mathbf{w}$. One approach for solving it is a *projected gradient* approach, in which one iterates between steps in the direction of the gradient of the Lagrangian $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_{i=1}^n \alpha_i \frac{y_i}{s_i}\mathbf{w}^{(i)}\mathbf{x}_i$ and projections to the constrained space, by calculating $s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\|$. With the right choices of step sizes, such approaches are guaranteed to converge to local minima.

Fortunately, we can use the fact that the problem is a QP for any given set of s_i's, and devise a faster iterative algorithm. For a given tuple of s_i's, we solve a QP for $\mathbf{w}$, and then use the resulting $\mathbf{w}$ to calculate new s_i's. To solve the QP, we derive the dual for given s_i's

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i \frac{y_i}{s_i}\langle \mathbf{x}_i, \mathbf{x}_j\rangle \frac{y_j}{s_j}\alpha_j \\ s.t. \quad & 0 \le \alpha_i \le C\ , \quad i = 1 \dots n \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned} \tag{14}$$

where the inner product $\langle \cdot, \cdot \rangle$ is taken only over features that are valid for both $\mathbf{x}_i$ and $\mathbf{x}_j$ (kernels are discussed in the next section). Fig. 2 provides a pseudo code of this iterative algorithm.

The dual solution is used to find the optimal classifier as in standard SVM, by setting $\mathbf{w} = \sum_{i=1}^n \alpha_i \frac{y_i}{s_i}\mathbf{x}_i$. This separating hyper plane can be used to predict the label of any new sample x_i by calculating the sign of the thresholded dot product $sign(\mathbf{w}^{(i)}\mathbf{x}_i - b)$ where b is set as in standard SVM (see for example p. 203 in Schölkopf and Smola, 2002) .

The primary difference between this algorithm and the projected gradient approach is that, instead of a series of small gradient steps, we take bigger leaps and follow each by a similar projection back into the constrained space. It is easy to see that a fixed point of this iterative procedure achieves an optimal solution for Eq. (13), since it achieves a minimal $\|\mathbf{w}\|$ while obeying the s_i constraints. As a result, when this algorithm converges, it is guaranteed that we reach a (local) optimum of the original problem Eq. (13). Unfortunately, the convergence of this iterative algorithm is not always guaranteed, and therefore it must be stopped when some stopping criterion is reached. In practice, we used cross validation to choose an early stopping point, and found that the best solutions were obtained within very few steps (2-5). This means that, in practice, the running time of this algorithm is expected to be at most five times of that of running a single SVM. Our specific implementation used matlab scripts for data manipulation and CPLEX for optimization, yielding a total runtime of 8 seconds for a set of 200 samples and 500 features on a 2.4 Ghz Intel Xeon CPU. It is possible to extend this approach and add multiple restarts with randomized starting points, and choose the best one using cross validation. One implementation advantage of this algorithm is that the effect of the s_i factors can be absorbed into the $\frac{y_i}{s_i}$ term (see Eq. (14)), rendering the $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$ term (and later the kernel matrix, see next section) independent of $\mathbf{s}$ and allowing it to be calculated in advance.

Iterative optimization/projection

Initialization:

Initialize ${s_i}^0 = 1$ for all $i = 1 \dots n$.

Iterations

repeat

Find optimal $\mathbf{w}^t(\mathbf{s}^{t-1})$. Solve Eq. (14) for the current $\mathbf{s}$.

Update $\mathbf{s}^t(\mathbf{w}^t)$, using $s_i = \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\|$.

until (stopping criterion)

Figure 2: Pseudo-code of the iterative projections algorithm.

We also tested two other approaches for optimizing Eq. (13). In the first, we replaced the above update of $\mathbf{s}$ with an optimization step. In this step we minimized $(s_i - \|\mathbf{w}^{(i)}\|/\|\mathbf{w}\|)^2$ subject to the $y_i \mathbf{w}^{(i)} \mathbf{x}_i \geq (1 - \xi_i)\mathbf{s}_i$ constraints. The solution to this problem can be derived analytically. This approach performed slightly worse than the iterative approach of Fig. 2. Second, we took a hybrid approach, combining gradient ascent over s and QP for w. Specifically, we iterated between solving Eq. (14), and making small steps in the direction of the gradient of the dual w.r.t. $\mathbf{s}$. This approach also did not perform as well as the iterative approach of Fig. 2.

4. Kernels for missing features

The power of the SVM approach can be largely attributed to the flexibility and efficiency of nonlinear classification allowed through the use of kernels. Because the dual objective of Eq. (14) depends only on the inner products of the data instances, we can consider the features to lie in a space of much higher dimension than is feasible in the primal formulation of Eq. (13). As a result, these objectives can be kernelized in the same manner as in standard SVMs. For example, for Eq. (14) we write

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \frac{y_i}{s_i} K\left(\mathbf{x}_i, \mathbf{x}_j\right) \frac{y_j}{s_j} \alpha_j \\ s.t. \quad & 0 \leq \alpha_i \leq C \;, \quad i = 1 \dots n \quad ; \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad . \end{aligned} \tag{15}$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function that simulates an inner product in the higher dimensional feature space. Classification of new samples are obtained as in standard SVM by calculating the margin $\rho(x_{new}) = \sum_j y_j \alpha_j \frac{1}{s_j} K(x_j, x_{new}) \frac{1}{s_{new}}$.

Importantly, kernels in this formulation operate over vectors with missing features, and we therefore have to develop kernels that handle this case correctly. Fortunately, for some standard kernels, there is an easy procedure to construct a modified kernel that takes such missing values into account.

We focus on kernels whose dependence on their inputs is through their inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Two examples for such kernels are polynomial and Sigmoidal kernels. A kernel in this class can be easily transformed into a kernel that handles missing features, by considering only features that are valid for both samples, and skipping the rest. This is not equivalent to ignoring missing features (as in filling with zeros), since the algorithm applied on Eq. (15) using this kernel does already correct for the missing information by taking into account the s_i terms.

For instance, for a polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d$, define the modified kernel

$$K'(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{F}} + 1)^d \quad ,$$

with the inner product calculated over valid features $\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{F}} = \sum_{k: f_k \in \mathcal{F}(j) \cap \mathcal{F}_i} \langle \mathbf{x}_{ik}, \mathbf{x}_{jk} \rangle$. It is straightforward to see that K' is a kernel, since instances with missing values can be filled with zeros: Formally, define $x' = h(\mathbf{x})$, where $h : \{\mathbb{R}, \mathrm{NaN}\}^d \to \mathbb{R}^d$ replaces invalid entries (missing features) with zeros. Then we have $K'(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}'_i, \mathbf{x}'_j)$, simply since multiplying by zero is equivalent to skipping the missing values. Hence K' is a kernel.

5. Experiments

We evaluate the performance of our approaches in four different missingness scenarios. First, as a sanity check, we explore performance when features are missing at random. In this setting our methods are not expected to provide considerable gain over other methods, but we hope they will perform equally well. Second, we study a visual object recognition application where some features are missing because they cannot be located in the image. Finally, we apply our methods to a problem of biological network completion, where missingness patterns of features is determined by the known structure of the network.

For all applications, we compare the two variants of our method with five common approaches for filling missing features.

- **Zero**. Missing values were set to zero.
- **Mean**. Missing values were set to the average value of the feature over all data (training+testing sets).
- **Flag**. Additional features ("flags") were added, explicitly denoting whether a feature is missing for a given instance. To minimize the number of added flags, they were shared between features in the following way: all features that had the same pattern of missingness across samples were assigned a single flag. [1]
- **kNN**. Missing features were set with the mean value obtained from the K nearest neighbors instances. Neighborhood was measured using a Euclidean distance in the subspace relevant to each pair of samples. The number of neighbors was varied across $K = 3, 5, 10, 20$, and the best result of these four (on test data) is shown (possibly overestimating performance.)

1. In practice, features are often structured such that validity is determined at the level of groups of features, that is, all features in a group are either valid or invalid together. This happens, for example, if a part of an object is missing in an image, since all its corresponding features are invalid. Using a single flag to signal the validity of each group, can reduce the number of added flags significantly.

- **EM**. Generative model in the spirit of Ghahramani and Jordan (1994). A Gaussian mixture model is learned by iterating between (1) learning a GMM model of the *filled* data (2) re-filling missing values using cluster means, weighted by the posterior probability that a cluster generated the sample. Covariances were assumed spherical. The number of clusters was varied across $K = 3, 5, 10, 15, 20$, and the best result is reported.

- **Averaged norm** (*avg* $|w|$). Our approximate convex approach described in section Sec. 3.2.

- **Geometric margin** (*geom*). Our exact non-convex approach described in section Sec. 2, and solved iteratively.

In all of the experiments, we used a five-fold cross validation procedure. That is, reported performance was evaluated on a testing set that was not used during training. In addition, 20% of the training set was used for choosing optimization parameters, such as the type of kernel and its parameters. We compared polynomial kernels (dim=1,2,3), and the reported result is the one that obtained the best result over the validation set, which was almost always a second order polynomial kernel. This set was used to to choose an early stopping point of the *Geometric* method to prevent over-fitting. In all applications below, we found that the best stopping time was between 2 and 5 iterations of the algorithm.

5.1 Missing at Random

As a first sanity check, we evaluate performance when feature values are missing at random. Recall that in this case, our method is not expected to perform better than alternative approaches. We do, however, want to make sure that our performance is not worse in this scenario. We therefore evaluated the performance in a setting where we can control the mechanism that governs missing features.

We first used the well known datasets from the UCI repository. We downloaded the datasets in Matlab format from $www.datalab.uci.edu/data/mldb-sgi/mat/$ that have two classes and no nominal features. To make the effect of missing feature prominent, we removed 90% of the features of each sample randomly, and applied the competing methods described above for each of the tasks. Tests with lower missingness values yielded similar results, usually with lower magnitude.

The results are summarized in Table 1, showing that most methods performed similarly, with little significant advantage to any of the methods. In only one case (*sick*) we found a significant difference, and in that case the (*geom*) method out-performed the other methods. In four out of six cases, geom achieved the best performance, often together with some other method.

Most UCI datasets have no more than a few dozen features, a small number when compared to real world applications. To test a *missing-at-random* scenario with more features we used 1200 images of the digits 5 and 6 from MNIST (LeCun, 1998). 1000 images were taken from the predefined training data and 200 from the test data. We used the central $28 \times 28 = 484$ pixels of each image and removed a square patch of pixels from each image that covered 25% of the total number of pixels. The location of the patch was uniformly sampled for each image, and typical examples are given in Fig. 3. This procedure was

	geom	avg w	none	mean	flag	EM	kNN
sick	**.90±.01	.81±.03	.81±.03	.63±.29	.84±.02	.60±.30	.70±.25
pima	*.66±.05	.64±.05	*.66±.04	.65±.04	.65±.04	.65±.03	.59±.07
hepatitis	*.78±.05	.77±.06	*.78±.05	*.78±.06	*.78±.05	*.78±.06	.77±.05
echo	.66±.08	.63±.06	.63±.08	*.67±.07	.64±.07	*.67±.07	*.67±.07
iono	.67±.06	.66±.06	.66±.06	*.71±.04	.63±.06	.70±.05	.67±.06
hypo	*.95±.01	.91±.03	.93±.02	.65±.34	.94±.02	.67±.33	.81±.22
mnist	*.95±.005	*.95±.005	*.95±.005	.94±.007	.93±.006	*.95±.004	.94±.006

Table 1: Mean accuracies (± standard errors over five random partitions, each separating the data into five cross validation sets) for classification of UCI datasets with randomly missing values. Bottom line is for the MNIST dataset. Best method for each dataset is marked with an asterisk. Significant improvement marked with two asterisks.

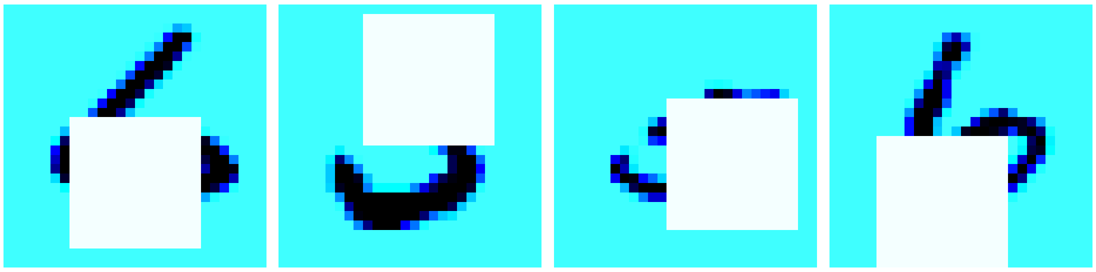

Figure 3: Examples of MNIST images of the digits '5' and '6' after a square patch of pixels is removed.

repeated five times to create five training and testing sets. We then applied the algorithms described above. The results were similar to the UCI case, and most methods performed at a comparable level of 95 percent accuracy.

5.2 Visual object recognition

We now turn to evaluating our methods in an application where instances have structurally missing features. We consider a visual object recognition task, where we attempt to determine if an object from a certain class (automobiles) is present in a given input image. This task has seen much work in recent years, see for example Berg et al. (2005), Grauman and Darrell (2005), Fergus et al. (2003), and discriminative approaches have typically produced very good results (Grauman and Darrell, 2005, Quattoni et al., 2005).

Features in these methods are commonly constructed from *regions of interest* (patches) in the image. These patches typically include several candidates for any object part, and some images may have more candidates for a given part than others. For example, a feature related to the trunk of a car may not be found at all in a picture of a hatch-back car. In this case, we consider the corresponding features to be structurally missing.

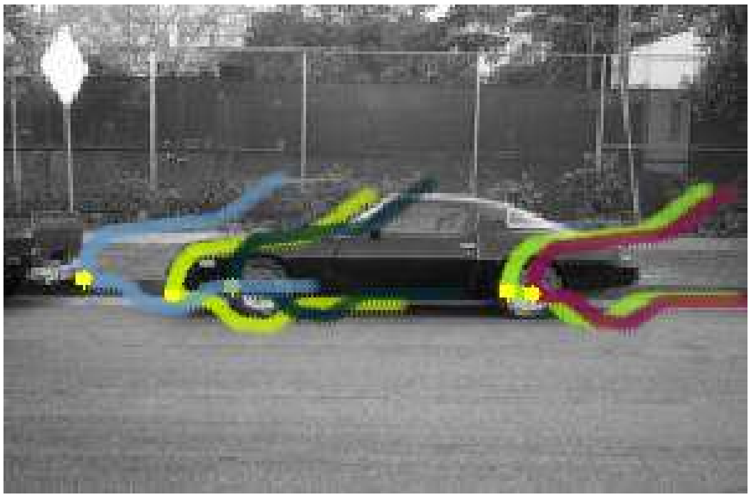

Figure 4: Landmark candidates are extracted by matching local edge information to the generative model and choosing the best K candidates for each landmark. Here we show the best 5 candidates matched to the edge feature of the front bumper. For each candidate, we extract a square (21×21) patch. Using a PCA codebook computed from a large set of positive and negative training examples, we then extract the first 10 principal components from the patch. These values are appended to create the feature vector for the image.

Our object model contains a set of "landmarks", defining the outline of an object, for which we find several matches in a given image (see Elidan et al., 2006, for details). Fig. 4 shows examples of matches for the front windshield landmark. Because of the noisy matches, the highest scoring candidate often does not match the true landmark, and the number of high-quality matches (features) varies in practice. It is in precisely such a scenario that we expect our proposed algorithm to be effective. The detailed description of this model is out of the scope of our current work and is described in Elidan et al. (2006).

Concretely, we located up to 10 candidate patches (21×21 pixels) that were promising (likelihood above a given threshold) for each of the 19 landmarks in the car model. For each candidate, we compute the first 10 principal component coefficients of the image patch and concatenate these descriptors to form the image feature vector with 1900 features per image. If the number of descriptors for a given landmark is less than ten, we consider the rest to be structurally absent.

Fig. 5 compares the accuracy of the different methods. The geometric approach was found to be significantly superior to all other methods. To further evaluate our method, we qualitatively examined the classification results for several images across the various classification methods. Fig. 6 shows the top 10 matches for the front windshield landmark for four test instances. In the first (upper left) example, we see that some of the matches are correctly localized near the true landmark location, while some are far off in the upper right of the image. In this case, we would expect that our thresholding technique will reject the bad matches, allowing only a subset to be present in the image feature vector. Because

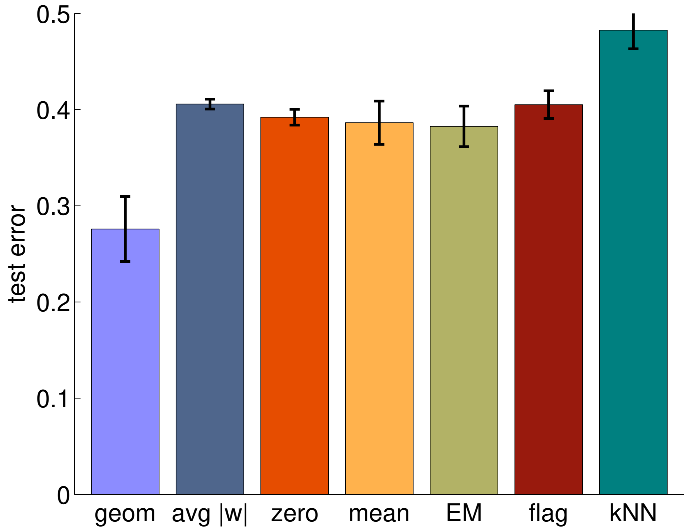

Figure 5: Classification accuracy of the different methods for the Object recognition in natural images. Error bars are standard error of the mean over the five cross validation sets.

we only include "good" matches in the feature vector, we would expect a classifier learned using our method to outperform methods that do not handle these missing features as well. This is indeed what we observe, and our classifier correctly labels both this image and the upper left instance, while the other methods get it wrong.

In contrast, we can look at the two instances in the bottom row of Fig. 6. In the first case, all landmark matches seem to be good, while the in second, all matches appear to be incorrect. Due to the consistency of the match quality, we should not expect our method to perform differently from the other methods. Indeed, we find that, in these particular instances, the final classification is the same across all classifiers. This analysis provides strong evidence that our method is leveraging a proper handling of missing features to obtain a boost in performance for this application.

Different approaches for handling low confidence matches could be taken in this application, for example, by providing explicit confidence measurements as additional features. Preliminary experiments that we performed demonstrated that using explicit confidence values could further improve classification accuracy. Systematic experiments with this approach are not in the scope of the current manuscript. The classification approach described in this paper should handle best the case of truly non-existing features, as described in the next section.

5.3 Metabolic pathway reconstruction

As a final application, we consider the problem of predicting missing enzymes in metabolic pathways, a long-standing and important challenge in computational biology (Vert and

a. b. c. d.

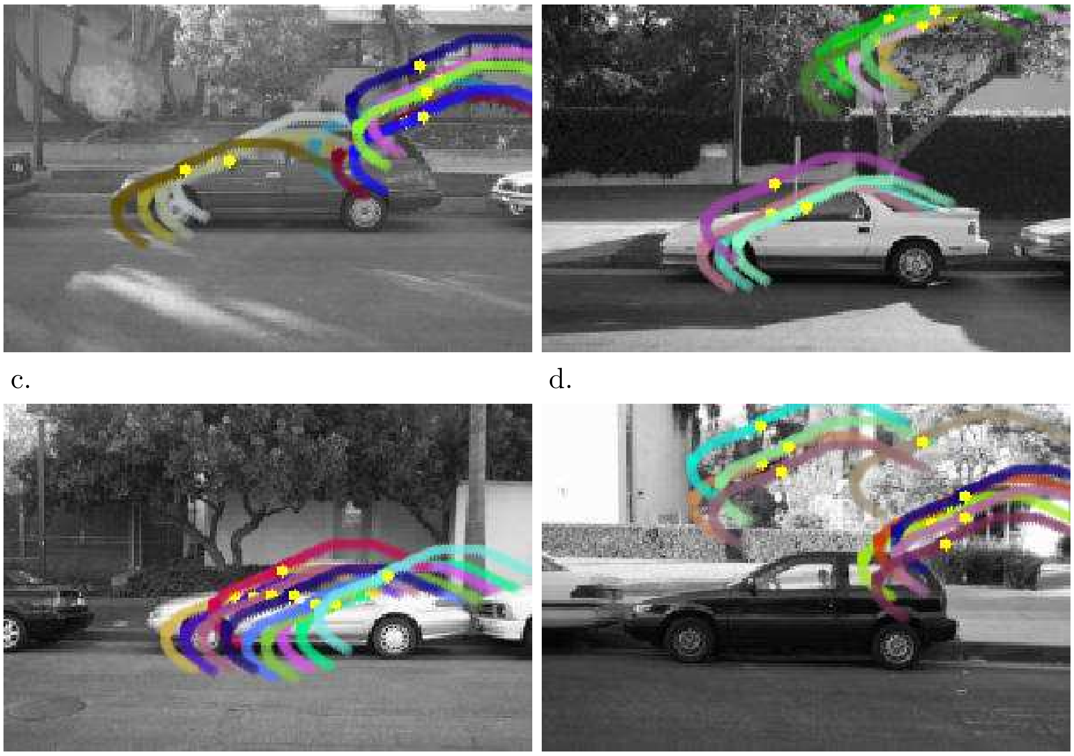

Figure 6: Examples of correct and misclassified instances. (a) and (b) Five of the candidate matches miss the car, leading to irrelevant features for this instance. These instances were correctly classified by the 'geometric margin' approach but misclassified by all other methods.(c) All candidate matches successfully match the car, making all features valid. This instance was correctly classified by almost all methods. (d) All candidate matches miss the car, this instance was misclassified by all methods as not containing a car.

Yamanishi, 2004, Kharchenko et al., 2003). Instances in this task have missing features due to the structure of the biochemical network.

To explain why our representation leads to non-existing features, we start with a brief domain background. Cells use a complex network of chemical reactions to produce their chemical building blocks Fig. 7. Each reaction transforms a set of molecular compounds into another set of molecules and requires the presence of a specific *enzyme*, to catalyze the reaction. For example, in Fig. 7 the enzyme *ARO7* catalyzes a reaction that turns *chorismate* to *prephanate*. For many reactions, the enzyme responsible for their catalysis is unknown, making it an important computational task to predict the identity of such missing enzymes.

Our approach for predicting missing enzymes is based on the observation that enzymes in local network neighborhoods usually participate in related functions. As a result, neighboring enzyme pairs have non trivial correlations over their features that reflect their functional

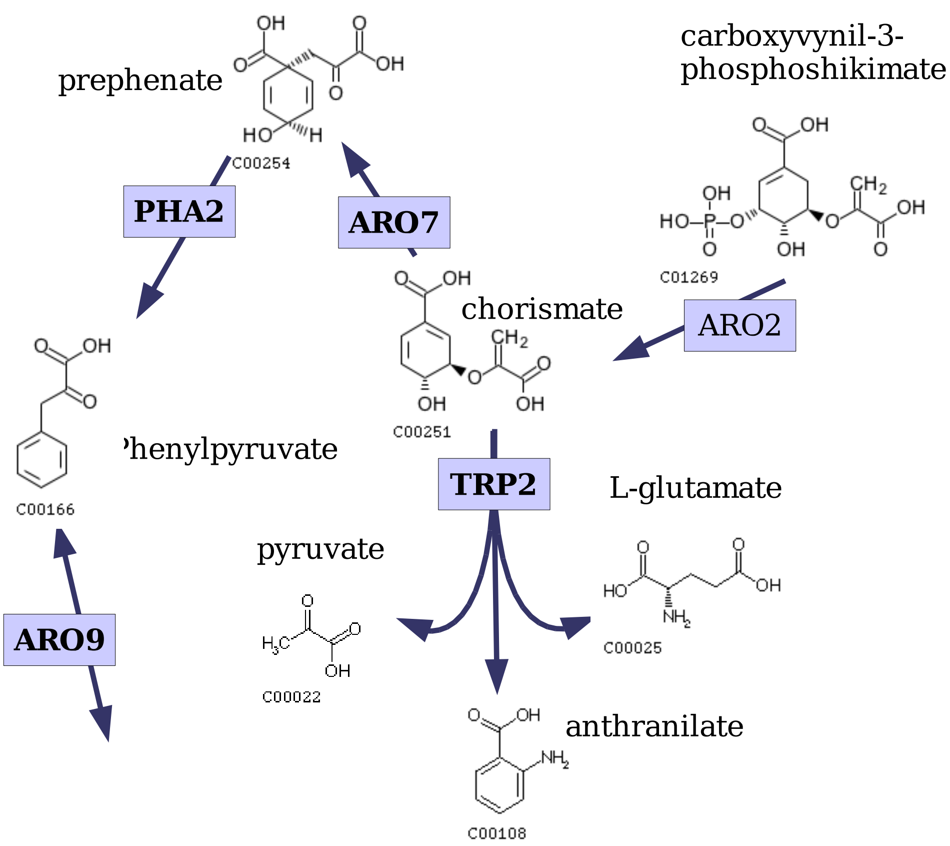

Figure 7: A fragment of the full metabolic pathway network in S. Cerevisiae. Chemical reactions (arrows) transform a set of molecular compounds into other compounds. Small molecules like CO_2 were omitted from this drawing for clarity. Reactions are catalyzed by enzymes (boxed names, e.g., ARO7), but in some cases these enzymes are unknown. The network imposes various neighborhood relations between enzymes assigned to reactions, like linear chains (ARO7,PHA2), forks (TRP2,ARO7) and funnels (ARO9,PHA2)

relations. Importantly, different types of network neighborhood relations between enzyme pairs lead to different relations of their properties. For example, an enzyme in a linear chain depends on the outputs of the preceding reaction as its own inputs. Hence, the genes that code for these enzymes are expected to be co-expressed (Kharchenko et al., 2003, Vert and Yamanishi, 2004, Kharchenko et al., 2005). On the other hand, enzymes in forking motifs (using a shared input compound, but creating different outputs) often have anti-correlated expression profiles (Ihmels et al., 2003).

To preserve the distinction between different neighbor relations, we defined a set of network motifs, including *forks* (same input, different outputs), *funnels* (same output, different inputs), *linear chains*. Each enzyme is represented as a vector of features that measure its relatedness to each of its different neighbors, across different data types. A feature vector will have structurally missing entries if the enzyme does not have all types of neighbors. For example, the enzyme PHA2, which transforms *prephenate* to *phenylpyruvate* in Fig. 7, does not have a neighbor of type *fork*, and therefore all features assigned to such a neighbor are absent in the representation of the reaction *"Prephanate → Phenylpyruvate"*.

We used the metabolic network of budding yeast (S. Cerevisiae), as reconstructed by Forster et al. (2003), after removing the 14 metabolites of highest degree (metabolic currencies), and reactions with unknown enzymes. The resulting network had 1265 directed reactions. We used three types of data for enzymes attributes: A compendium of gene expression assays, protein domains content of enzymes, and the cellular localization of proteins. Each of these datasets was used to compute a measure of similarity between an enzyme and each of its neighbors. Specific details are given in Appendix A. All similarity features were concatenated, yielding a feature vector of length ~ 3900. As explained above, features in this vector are often missing since some enzymes do not have all types of neighbors.

We created positive examples from the reactions with known enzymes (~ 520 reactions), and negative examples by plugging a random impostor genes into each neighborhood. Negative genes were uniformly chosen from the set of other metabolic enzymes. The result is a balanced training set. We trained an SVM classifier using a 5-fold cross validation procedure, as described above.

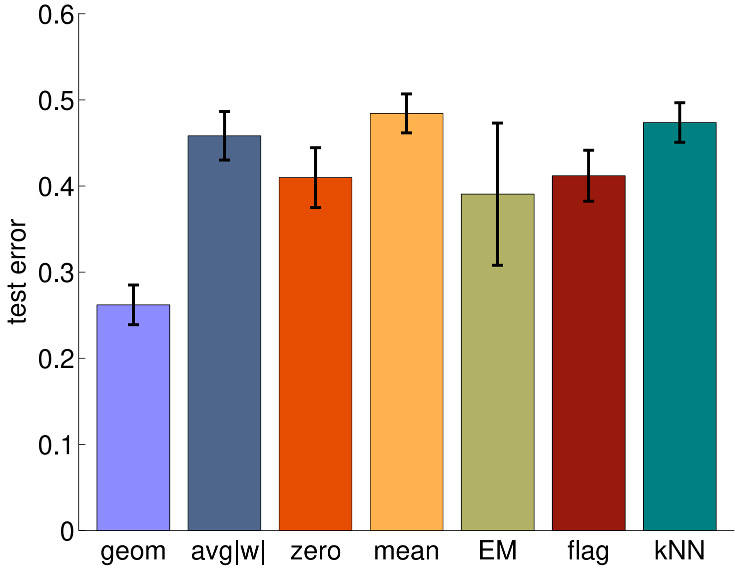

Figure 8: Classification accuracy for compared methods. The classification task is to identify if a candidate enzyme is in the right "neighborhood". Error bars are standard errors of the mean, over 5 cross validation sets.

Figure 8 shows the test classification error of the different methods in the gene filling task. The *geometric margin approach* achieves significantly better performance in this task.

kNN achieved very poor performance compared to all other methods. One reason could be that the Euclidean distance is inappropriate for the current task and that a more elaborate distance measure needs to developed for this data. Learning metrics is a complicated task in general, and more so in the current problem since the feature vectors contain entries of several different types, making it unlikely that a naive distance measure would work well in this task. The approach of adding explicit flags to denote the validity of each block of features ('flag') performed extremely poorly, and was therefore removed from this figure. It is likely that this is a consequence of the large increase in dimensionality of the feature vector that results from using this method.

6. Discussion

We presented a novel method for max-margin training of classifiers in the presence of missing features, where the pattern of missing features is an inherent part of the domain. Instead of completing missing features in a preprocessing phase, we developed a max-margin learning objective based on a geometric interpretation of the margin when different instances essentially lie in different spaces. This allows us to directly classify instances by skipping the non-existing features, rather than filling them with hypothetical values.

Our geometric method was found to be significantly superior in two challenging real-life problems with non-existing features. It was also found to be competitive with a range of imputation approaches when tested in missing-at-random settings. We found that a second approach ($avg\ |w|$), which approximates the instance-specific geometric margins, did not perform better than simple imputation methods. A simple approach of adding flags that explicitly denote the existence of a feature, often performed well. It does however increase the number of features and is expected to perform better in dataset with low feature-to-samples ratio. Adding those flags allow the classifier to use information in the missingness pattern, which could become useful when this is highly correlated with the class label. It remains an interesting question to study the relation between informative missingness patterns and the direct classification discussed here.

The standard treatment of missing features is based on the concept that missing features exist but are unobserved due to noise or insufficient measurements. This assumption underlies the approach of completing features before the data is used in classification. This paper focuses on a different scenario in which features are inherently absent, and discusses several domains where this could happen. In such scenarios, it is not clear why we should guess hypothetical values for undefined features, as feature values are filled based on other observed values, and therefore cannot add information to our classifiers. In fact, by completing features that are not supposed to be part of an instance, we may be confounding the learning algorithm by presenting it with problem which may be harder than the problem we are actually interested in.

This paper presents an alternative to imputation, which can become useful in other scenarios where imputation is not practical. For example, applications like collaborative filtering for movie recommendation, operate in very high-dimensional spaces that are also very sparse. Typical datasets have millions of users, tens of thousands of movies, but only a tiny fraction of the potential entries are known. While imputation methods may be impractical in such a setting, approaches based on skipping unknown entries could reduce the computation costs significantly, by operating on known entries only.

Interestingly, the problem of classifying with missing features is related to another problem, where individual reliability measures are available for features at each instance separately. This is a common case in scientific data, where the reliability of each experiment could be provided separately. For example, DNA micro-array experiments have inherent measures of experimental noise levels, and biological variability is often estimated using replicates. This problem can be viewed in the same framework described in this paper: the geometric margin must be defined separately for each instance since the different noise levels distort the relative scale of each coordinate of each instance separately. Relative to this setting, the completely missing and fully valid features discussed in this paper are extreme points on the spectrum of reliability. It will be interesting to see which aspects of the geometric formulation discussed in this paper can be extended to this new problem.

Acknowledgments

We thank Aviv Regev for insightful discussions of metabolic pathways, and Amir Globerson for discussing early versions of this work. This work was supported by a grant from the National Science Foundation, and by the DARPA transfer learning program under contract FA8750-05-2-0249.

References

A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

G. Elidan, G. Heitz, and D. Koller. Learning object shape: From drawings to images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271, 2003.

J. Forster, I. Famili, P. Fu, B.. Palsson, and J. Nielsen. Genome-scale reconstruction of the saccharomyces cerevisiae metabolic network. *Genome Research*, 13(2):244–253, February 2003.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, October 2005.

W.K. Huh, J.V. Falvo, L.C. Gerke, A.S. Carroll, R.W. Howson, J.S. Weissman, and E.K. O'Shea. Global analysis of protein localization in budding yeast. *Nature*, 425:686–691, 2003.

N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P.S. Langendijk-Genevaux, M. Pagni, and C.J.A. Sigrist. The prosite database. *Nucleic Acids Res.*, 34:D227–D230, 2006.

J. Ihmels, R. Levy, and N. Barkai. Principles of transcriptional control in the metabolic network of saccharomyces cerevisiae. *Nature Biotechnology*, 22:86–92, 2003.

A. Jaimovich, G. Elidan, H. Margalit, and N. Friedman. Towards an integrated protein-protein interaction network. In *RECOMB 2005*, 2005.

D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *19th international confernce on Machine learning (ICML 2002)*, 2002.

A. Kapoor. *Learning Discriminative Models with Incomplete Data*. PhD thesis, MIT Media Lab, Feb 2006 2006.

P. Kharchenko, D. Vitkup, and GM Church. Filling gaps in a metabolic network using expression information. *Bioinformatics*, 2003.

P. Kharchenko, GM Church, and D. Vitkup. Expression dynamics of a cellular metabolic network. *Molecular Systems Biology*, 2005.

Y. LeCun. *MNIST Database of Handwritten Digits*. NEC Research, 1998. URL `http://yann.lecun.com/exdb/mnist/`.

R.J.A. Little and D.B. Rubin. *Statistical Analysis with Missing Data*. NY wiley, 1987.

M. Sousa Lobo, L. Vandenbergheb, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, 1998.

A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1097–1104, Cambridge, MA, 2005. MIT Press.

P. Roth. Missing data: A conceptual review for applied psychologists. *Personnel Psychology*, 47(3): 537–560, 1994.

J. L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, 1997.

B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization Optimization and Beyond*. MIT Press, Cambridge, MA, 2002.

V.N. Vapnik. *The nature of statistical learning theory*. SpringerVerlag, 1995.

J. P. Vert and Y. Yamanishi. Supervised graph inference. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1433–1440, Cambridge, MA, 2004. MIT Press.

Appendix A. Similarity measures for enzyme predictions

We used three types of data, and defined a relevant similarity measure for each one.

- A compendium of 645 gene expression experiments with RNA expression levels of each gene; each experimental condition k contributed one feature, the point-wise Pearson correlation $\frac{\mathbf{x}_i(k)\mathbf{x}_j(k)}{\|\mathbf{x}_i\|\|x_j\|}$.

- The protein-domain content of each enzyme as found by the Prosite database (Hulo et al., 2006). Each domain k contributed one feature, measured as the point-wise symmetric D_{KL} measure

$$\mathbf{x}_i(k)\log\left(\frac{\mathbf{x}_i(k)}{\bar{x}(k)}\right)+\mathbf{x}_j(k)\log\left(\frac{\mathbf{x}_j(k)}{\bar{x}(k)}\right) \quad ; \quad \bar{x}(k)=\frac{\mathbf{x}_j(k)+\mathbf{x}_i(k)}{2} \quad .$$

- The cellular localization of the protein Huh et al. (2003); each cellular localization contributed one feature, the point-wise Hamming distance: 0 if the same and 1 otherwise.