# Learning Local Invariant Mahalanobis Distances

**Ethan Fetaya** ETHAN.FETAYA@WEIZMANN.AC.IL
Weizmann Institute of science

**Shimon Ullman** SHIMON.ULLMAN@WEIZMANN.AC.IL
Weizmann Institute of science

## Abstract

For many tasks and data types, there are natural transformations to which the data should be invariant or insensitive. For instance, in visual recognition, natural images should be insensitive to rotation and translation. This requirement and its implications have been important in many machine learning applications, and tolerance for image transformations was primarily achieved by using robust feature vectors. In this paper we propose a novel and computationally efficient way to learn a local Mahalanobis metric per datum, and show how we can learn a local invariant metric to any transformation in order to improve performance.

Metric learning is a machine learning task which learns a distance metric $d(x, y)$ between data points, based on data instances. As distances play an important role in many machine learning algorithms, e.g. k-Nearest Neighbor and k-Means clustering, finding an appropriate metric for the task can improve performance considerably. This approach has been applied successfully to many problems such as face identification (Guillaumin et al., 2009), image retrieval (Hoi et al., 2008; Chechik et al., 2010), ranking (Lim & Lanckriet, 2014) and clustering (Xiang et al., 2008) to name just a few.

A standard approach to metric learning is to learn a global Mahalanobis metric

$$d(x,y)^2_M = (x-y)^T M(x-y) \tag{1}$$

Where $M$ is a positive semi-definite matrix (PSD). The PSD constraint only assures this is a pseudometric , but for simplicity we will not make this distinction. Various algorithms (Weinberger & Saul, 2009; Bar-Hillel et al., 2005; Davis et al., 2007) differ by the objective through which they learn the matrix $M$ from the data. As $M$ is a PSD matrix, it can be written as $M = L^T L$ and therefore

$$d(x,y)^2_M = (x-y)^T M(x-y) = ||\tilde{x} - \tilde{y}||_2^2$$
$$\tilde{x} = Lx,\ \tilde{y} = Ly.$$

This means that finding an optimal Mahalanobis distance is equivalent to finding the optimal linear transformation on the data, and then using $L_2$ distance on the transformed data. This approach has two limitations, first it is limited to linear transformation. Second, it requires a large amount of labeled data.

One approach that can be used to overcome the first limitation is to use local distances (Frome et al., 2006) where we learn a unique distance function per training datum. Local approaches do not produce, in general, a global metric (as they are usually not symmetrical) but are commonly considered metric learning nonetheless. These methods, in general, need similar and dissimilar training data for each local metric.

In our current work we use a local approach inspired by the work on exemplar-SVM (Malisiewicz et al., 2012), that showed that using only negative examples can suffice for good performance. The intuition behind this is that objects of the same class do not necessarily have to be similar, but objects from different classes must be dissimilar. We will show how to learn a local Mahalanobis distance that for each datum tries to keep the non-class as far away as possible. This approach can use a large amount of weakly supervised data, as in many cases negative examples are easier than positive examples to acquire. For example, if we are interested in face identification, we can learn a local metric around a query face image given a bank of training face images, which we only assume does not belong to the queried person. Unlike other metric learning methods, we

*Proceedings of the $32^{nd}$ International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP volume 37. 

will not need any labels on which image belongs to which person in the negative set.

The intuition why Mahalanobis distances are the natural model for local metrics is simple. Assume we have some metric $d(x, y)$ on the dataset and assume that it is smooth (at least continuously twice differentiable). From the metric properties we know that if we fix $x$ and look at $f(y) = d(y, x)$ then $f$ has a global minimum at $y = x$. Applying second order Tylor approximation to $f$ around $x$ we get

$$d(y,x) = f(y) \approx f(x) + (y-x)^T \nabla f(x) + \\ (y-x)^T \nabla^2 f(x)(y-x) = (y-x)^T \nabla^2 f(x)(y-x) \quad (2)$$

The equality holds since $x$ is the global minimum with value $f(x) = d(x, x) = 0$, and this also implies that $\nabla^2 f(x)$ is positive semidefinite. While the Taylor approximation only holds for values of $y$ close to $x$, as metric methods such as k-NN focus on similar objects the approximation should be good at the points of interest. This observation leads us to look for local matrices that are of the form of a Mahalanobis distance.

We will first define our local Mahalanobis distance learning method as a semidefinite programming problem. We will then show how this problem can be solved efficiently without any costly matrix decompositions. This allows us to solve high dimensional problems that regular semidefinite solvers cannot handle.

The second major contribution of this paper will be to show how invariant local matrices can be learned. In many cases we know there are simple transformations that our metric should not be sensitive to. For example, small translation and rotation on natural images. We know a priori that if $x' = T(x)$, where $T$ is the said transformation, then $d(x, x') \approx 0$. We will show how this prior knowledge about our data can by incorporated by learning a local invariant metric. This also can be done in an efficient manner, and we will show that this improves performance in our experiments.

## 1. Related work

Metric learning is an active research field with many algorithms, generally divided into linear (Weinberger & Saul, 2009) which learn a Mahalanobis distance, non-linear (Kedem et al., 2014) that learn a nonlinear transformation and use $L_2$ distance on the transformed space, and local which learn a metric per datum. The LMNN and MLMM (Weinberger & Saul, 2009) algorithm are considered the leading metric learning method. For a recent comprehensive survey that covers linear, non-linear and local methods see (Bellet et al., 2014).

The exemplar-SVM algorithm (Malisiewicz et al., 2012) can be seen as a local similarity measure. This is obtained by maximizing margins, with a linear model, and is weakly supervised as our work. Unlike exemplar-SVM, we learn a Mahalanobis matrix and can learn an invariant metric. Another related work is PMLM (Wang et al., 2012), which also finds a local Mahalanobis metric for each data point. However, this method uses global constraints, and therefore cannot work with weakly supervised data, i.e. a single positive example. All the techniques above do not learn local invariant metrics.

The most common way to achieve invariance, or at least insensitivity, to a transformation in computer vision applications is by using hand-crafted descriptors such as SIFT (Lowe, 2004) or HOG (Dalal & Triggs, 2005). Another way, used in convolutional networks (Lecun et al., 1998), is by adding pooling and subsampling forcing the net to be insensitive to small translations. It is important to note that transformations such as rotations have a global behaviour, i.e. there is a global consistency between the pixel movement. This global consistency is not totally captured by the pooling and subsampling. As we will see in our experiments, using an invariant metric can be useful even when working with robust features such as HOG.

## 2. Local Mahalanobis

In this section we will show how a local Mahalanobis distance with maximal margin can be learned in a fast and simple way.

We will assume that we are given a single query image $x_0$ that belong to some class, e.g. a face of a person. We will also be given a set of negative data $x_1, ..., x_n$ that do not belong to that class, e.g. a set of face images of various other people. We will learn a local Mahalanobis metric for $x_0$, $M(x_0) \succeq 0$, where $M \succeq 0$ means $M$ is positive semi-definite. For matrices $M, N$, we will denote by $||M||$ the Frobinous norm $||M||^2 = \sum_{ij} M_{ij}^2$ and by $\langle M, N \rangle$ the standard inner product $\langle M, N \rangle = \sum_{ij} M_{ij} N_{ij}$.

We wish to find a Mahalanobis matrix $M$ given the positive datum $x_0$ and the negative data $x_1, ..., x_n$. Large margin methods have been very successful in metric learning (Weinberger & Saul, 2009), and more generally in machine learning, therefore, our algorithm will look for the PSD matrix $M$ that maximizes the distance to the closest negative

example

$$M = \arg\max_{M \succeq 0} ( \min_{1 \leq i \leq n} (x_i - x_0)^T M (x_i - x_0)) \quad (3)$$

The optimization cannot be solved as it is not bounded, since multiplying $M$ by a scalar multiplies the minimum distance by the same scalar. This can be solved by normalizing $M$ to have $||M|| = 1$. As normally done with margin methods, we can minimize the norm under fixed margin constraint instead of maximizing the margin under fixed norm constraint. The resulting objective is

$$\begin{aligned} M(x_0) &= \arg\min_M \frac{1}{2}||M||^2 \\ subject\ to: (x_i - x_0)^T M (x_i - x_0) &\geq 2 \quad \forall i \in \{1, ..., n\} \\ M &\succeq 0 \end{aligned} \quad (4)$$

Where the constant 2 is arbitrary and will be convenient later on. While this is a convex semidefinite programming task, it is very slow for reasonable dimensional data (in the thousands) even for state of the art solvers. This is because PSD solvers apply a projection to the semidefinite cone, performing an expensive singular value or eigen decomposition at each iteration.

To solve this optimization in a fast manner we will first relax the PSD constraint and look at the following objective

$$\begin{aligned} M(x_0) &= \arg\min_M \frac{1}{2}||M||^2 \\ subject\ to: (x_i - x_0)^T M (x_i - x_0) &\geq 2 \quad \forall i \in \{1, ..., n\} \end{aligned} \quad (5)$$

We will then see how this is equivalent to a kernel SVM problem with a quadratic kernel, and therefore can be solved easily with off-the-shelf SVM solvers such as LIBSVM (Chang & Lin, 2011). Finally we will show how the solution of objective 5 is in fact the solution of objective 4 resulting in a fast solution to objective 4 without any matrix decomposition.

**Theorem 1.** *The solution of objective 5 is given by running kernel SVM with kernel $k(x, y) = \langle x, y \rangle^2$ on inputs $\tilde{x}_0, \tilde{x}_1, ..., \tilde{x}_n$ where $\tilde{x}_i = x_i - x_0$*

*Proof.* Define $\varphi(x) = x \cdot x^T$, a function that maps a column vector to a matrix. This function has the following simple properties:

- $k(x, y) = \langle x, y \rangle^2 = \langle \varphi(x), \varphi(y) \rangle$, i.e. the function $\varphi$ is the mapping associated with the quadratic kernel.
- For any matrix $W$, we have $\langle W, \varphi(x) \rangle = x^T W x$.

which can be easily verified using $\varphi(x)_{ij} = x_i x_j$. We can define auxiliary labelling $y_0 = -1$ and $y_i = 1$ for $1 \leq i \leq n$. Combining everything objective 5 can be rewritten as

$$\begin{aligned} M(x_0) &= \arg\min_M \frac{1}{2}||M||^2 \\ subject\ to: y_i \cdot (\langle M, \varphi(\tilde{x}_i) \rangle - 1) &\geq 1 \quad \forall i \in \{0, 1, ..., n\} \end{aligned} \quad (6)$$

where we can include $i = 0$ as $\langle M, \varphi(\tilde{x}_0) \rangle = 0$ for any matrix.

Objective 6 is exactly an SVM problem with quadratic kernel, with bias fixed to one, given inputs $\tilde{x}_0, ..., \tilde{x_n}$, proving the theorem. Notice also that for the identity matrix $M = I$ we have $\langle M, \varphi(\tilde{x_i}) \rangle > 0$ for $i \geq 1$ and $\langle M, \varphi(\tilde{x_i}) \rangle = 0$ for $i = 0$, therefore the data is separable by $M = I$ and the optimization is feasible. □

Now that we have shown how objective 5 can be converted into a standard SVM form, for which efficient solvers exists, we will show how it is the solution to objective 4.

**Theorem 2.** *The solution to objective 5 is the solution to objective 4.*

*Proof.* To prove the theorem it suffices to show that the solution is indeed positive semidefinite. A well known observation arrising from the dual formulation of the SVM objective (Burges, 1998) is that the optimal solution $M$ has the form

$$M = \sum_{i=0}^{n} \alpha_i y_i \varphi(\tilde{x}_i), \quad \alpha_i \geq 0. \quad (7)$$

Since $\varphi(x) \succeq 0$ for any $x$, as its only nonzero eigenvalue is $||x||$, $\varphi(\tilde{x}_0) = \varphi(0) = 0$, and $y_i = 1$ for $i \geq 1$ we get

$$M = \sum_{i=0}^{n} \alpha_i y_i \varphi(\tilde{x}_i) = \sum_{i=1}^{n} \alpha_i \varphi(\tilde{x}_i) \succeq 0, \quad (8)$$

where the positive semidefiniteness in eq. 8 is assured due to the set of PSD matrices being a convex cone. □

Combining theorem 1 with theorem 2 we get that in order to solve objective 4 it is enough to run an SVM solver with a quadratic kernel function, thus avoiding any matrix decomposition.

Looking at this as a SVM problem has further benefits. The SVM solvers do not compute $M$ directly, but return the set of support vectors $\tilde{x}_{i_1}, ..., \tilde{x}_{i_k}$ and coefficients $\alpha_{i_1}, ..., \alpha_{i_k}$ such that $M = \sum_k \alpha_{i_k} \varphi(\tilde{x}_{i_k})$. This allows us to work in high dimension $d$, where the $\mathcal{O}(d^2)$ memory needed to

store the matrix can be a problem, and can slow computations further. As the rank of the matrix is bounded by the number of support vectors, one can see that in many applications we get a relatively low rank matrix. This bound on the rank can be improved by using sparse-SVM algorithms (Cotter et al., 2013). In practice we got low rank matrices without resorting to sparse $SVM$ solvers.

## 3. Local Invariant Mahalanobis

For some applications, we know a priori that certain transformations should have a small effect on the metric. We will show how to include this knowledge into the local metric we learn, learning locally invariant metrices. In section 4 we will see this has a major effect on performance.

Assume we know a set of functions $T_1, ..., T_k$ that the desired metric should be insensitive to, i.e. $d(x, T_i(x))$ should be small for all $x$ and $i$. A canonical example is small rotations and translations on natural images. One of the major issues in computer vision arises from the instability of the pixel representation to these transformations. Various descriptors such as SIFT (Lowe, 2004) and HOG (Dalal & Triggs, 2005) offer a more robust representation, and have been highly successful in many computer vision applications. We will show in section 4 that even when using a relatively robust representation such as HOG, learning an invariant metric has a significant impact.

A natural way to mathematically formulate the idea of being insensitive to a transformation, is to require the leading term of the approximation to vanish in that direction. In our case this means

$$(T(x_0) - x_0)^T \nabla_y^2 d(x_0, y)(T(x_0) - x_0) = 0. \quad (9)$$

If we return to our basic intuition of the local Mahalanobis matrix as the Hessian matrix $\nabla_y^2 d(x_0, y)$, we can now state the new local invariant Mahalanobis objective

$$\begin{aligned} M(x_0) = \arg\min_M \frac{1}{2}||M||^2 \\ subject\ to: \\ (x_i - x_0)^T M(x_i - x_0) \geq 2 \quad \forall i \in \{1, ..., n\} \\ (T_j(x_0) - x_0)^T M(T_j(x_0) - x_0) = 0 \quad \forall j \in \{1, ..., k\} \\ M \succeq 0 \end{aligned} \quad (10)$$

We will show how by applying a small transformation to the data, we can reduce this to objective 4 which we can solved easily.

**Theorem 3.** *Define $V = span\{T_1(x_0) - x_0, ..., T_k(x_0) - x_0\}$, then the minimizer of objective 4 with $x_i - x_0$ replaced by $z_i$, its projection to $V^\perp$ is the minimizer of objective 10.*

*Proof.* For PSD matrices, the constraint that $(T_j(x_0) - x_0)^T M(T_j(x_0) - x_0) = 0$ is equivalent to $M(T_j(x_0) - x_0) = 0$. This can be seen if we write the vector in the basis of $M$ eigenvectors, and notice that components with positive eigenvalues have a positive contribution to the quadratic form. This means that $Mv = 0$ for all $v \in V$. Each vector $x_i - x_0$ can be split into two orthogonal elements, $x_i - x_0 = z_i + v_i$ where $v_i$ is its projection onto $V$ and $z_i$ is its projection onto $V^\perp$. Our equality constraints $(T_j(x_0) - x_0)^T M(T_j(x_0) - x_0) = 0$ now imply

$$(x_i - x_0)^T M(x_i - x_0) = (z_i + v_i)^T M(z_i + v_i) = z_i^T M z_i \quad (11)$$

since all the other terms vanish. We can now rewrite objective 10 as

$$\begin{aligned} M(x_0) = \arg\min_M \frac{1}{2}||M||^2 \\ subject\ to: z_i^T M z_i \geq 2 \quad \forall i \in \{1, ..., n\} \\ Mv = 0 \quad \forall v \in V \\ M \succeq 0 \end{aligned} \quad (12)$$

If we forget the equality constrains we get objective 4 with $x_i - x_0$ replaced by $z_i$. To finish the proof we need to show that the solution to the optimization without the equality constraints, does indeed satisfy them.

As we have already seen in the proof of theorem 2 the optimal solution is of the form $M = \sum \alpha_i \varphi(z_i) = \sum \alpha_i z_i \cdot z_i^T$. The vector $z_i$ is a member of $V^T$ so for $v \in V$

$$Mv = \left(\sum \alpha_i z_i \cdot z_i^T\right) v = \sum \alpha_i z_i \cdot (z_i^T v) = 0 \quad (13)$$

Proving that the solution satisfies the equality constraints. □

A few comments are worth noting about this formulation. First the problem may not be linearly separable, although in our experiments with real data we did not encounter any unseperable case. This can be easily solved, if needed, by the standard method of adding slack variables. Second, the algorithm just adds a simple preprocessing step to the previous algorithm and runs in approximately the same time.

## 4. Experiments

### 4.1. Running time

We compared running the optimization with an SVM solver (Chang & Lin, 2011), to solving it as a semidefinite problem and as a quadratic problem (relaxing the

*Table 1.* Classification error for MNIST dataset.

| METHOD | ERROR |
|---|---|
| ESVM | 1.75% |
| ESVM+SHIFTS | 1.59% |
| LOCAL MAHAL | 1.69% |
| QUADSVM+SHIFTS | 1.50% |
| INV-MAHAL (OUR METHOD) | 1.26% |
| LMNN | 1.69% |
| MLMNN | 1.18% |

semidefinite constraint). The main limitation when running off-the-shelf solvers is memory. Quadratic or semidefinite solvers need a constraint matrix, which in our case is a full matrix of size $n \times d^2$ where $n$ is the number of samples and $d$ is the data dimension. We tested all three approaches on the MNIST dataset of dimension 784 using only 5000 negative examples, as this already resulted in a matrix of size 24.6Gb.

Currently first order methods, such as ADMM (Boyd et al., 2010), are the leading approaches to solving problems such as quadratic and semidefinite programming for large matrices. We used YALMIP for modeling and solved using SCS (O'Donoghue et al., 2013). The time to run this as an semidefinite program was $1152 \pm 417sec$. The time it took to run this as a quadratic program was $545 \pm 74sec$. In comparison, when we run this as an SVM problem it took at most $0.36sec$. We excluded the time needed to build the $n \times d^2$ constraint matrix for the quadratic and semidefinite solvers.

This order of magnitude improvement should not be a surprise, due to the high memory usage of standard quadratic solvers for this task. While it is possible to write quadratic solver for this task without the huge memory overhead, the experience with SVM has shown that solver designed specifically for SVM work much faster then generic quadratic solvers

### 4.2. MNIST

The MNIST dataset is a well known digit recognition dataset, comprising of $28 \times 28$ grayscale images on which we perform deskewing preprocessing. For each of the $60,000$ training images we computed a local Mahalanobis distance and local invariant Mahalanobis, by training each image separately using all the non-class training images as negative examples (ignoring all same-class images). For local invariance, we used 8 one-pixel translation and 6 small rotations as our transformations. At test time we performed $knn$ classification with $k = 3$ using the local metrics, where the distance of the test image to a training image $x_i$ is computed using the pre-learned Mahalanobis metric $M_i$. We show some examples of nearest neighbours in Figure 4.2. We compared this with exemplar-SVM, as the leading technique most similar to ours. We also compared our scores to exemplar-SVM where we add the tansformed images as positive training data. To show the importance of the invariance objective, we compare also to SVM with quadratic kernel to which we add the transformed data as positive training data (unlike the way we use the shifted data). Finally, we compared our results to the state-of-the-art metric learning LMNN method (linear metric), and to MLMNN, a local version of LMNN, which learns multiple metrics (but not one per datum).

As can be seen in table 1, we perform much better than exemplar SVM and are comparable with MLMNN. It is important to note that unlike MLMNN, we compare each datum only to negatives, so our method is applicable in scenarios where MLMNN is not.

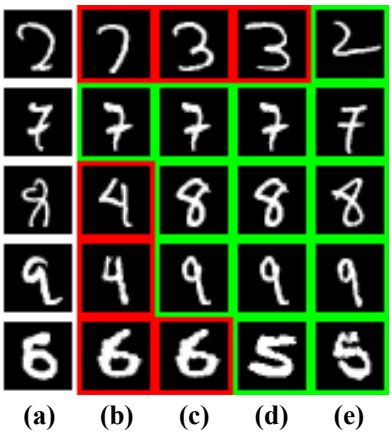


*Figure 1.* Nearest neighbour for various matrices. (a) original image (b) $L_2$ distance (c) exemplar-SVM (d) local-Mahalanobis (e) local invariant Mahalanobis

Another key observation is the difference between the invariant-Mahalanobis and the quadratic-SVM with shifts. While very similar functionally, we see that looking at the problem as a local Mahalanobis matrix gives important intuition, i.e. the way to use the shifted images, that leads to better performance.

### 4.3. Labeling faces in the wild (LFW)

LFW is a challenging dataset containing 13,233 face images of 5749 different individuals with a high level of variability. The LFW dataset is divided into 10 subsets, when the task is to classify 600 pairs of images from

one subset to same/not-same using the other 9 subsets as training data. We perform the unsupervised LFW task, where we do not use any labelling inside the training images we get, besides the fact that they are different than both test images.

We used the aligned images (Huang et al., 2012) which we represented using HOG features (Dalal & Triggs, 2005). For each test pair $(x_1, x_2)$ we compute the their local Mahalanobis matrices, $M_1$ and $M_2$, using the training set and use $(x_2-x_1)^T M_1(x_2-x_1)+(x_1-x_2)^T M_2(x_1-x_2)$ as their similarity score.

We compared our results to a cosine similarity baseline, to exemplar-SVM and exemplar-SVM with shifts. We note that we cannot use LMNN or MLMNN on this data, as we only have negative images with a single positive image.

*Table 2.* Classification error for LFW dataset.

| METHOD | ERROR |
|---|---|
| COSINE SIMILARITY | 30.57± 1.4% |
| ESVM | 26.90±2.2% |
| ESVM+SHIFTS | 27.12±2.3% |
| LOCAL MAHAL | 19.85±1.3% |
| INV-MAHAL (OUR METHOD) | 19.48±1.5% |

As we can see from table 2, the local Mahalanobis greatly out-performs the exemplar-SVM. We also see that even when using robust features such as HOG, learning an invariant metric improves performance, albeit to a lesser degree.

## 5. Summary

We showed an efficient way to learn a local Mahalanobis metric given a query datum and a set of negative data points. We have also shown how to incorporate prior knowledge about our data, in particular the transformations to which it should be robust, and use it to learn locally invariant metrics. We have shown that our methods are competitive with leading methods while being applicable to other scenarios where methods such as LMNN and MLMNN cannot be used.

## References


Bar-Hillel, Aharon, Hertz, Tomer, Shental, Noam, and Weinshall, Daphna. Learning a mahalanobis metric from equivalence constraints. *JMLR*, 2005.

Bellet, Aurlien, Habrard, Amaury, and Sebban, Marc. A survey on metric learning for feature vectors and structured data. *arXiv:1306.6709*, 2014.

Boyd, Stephen, Parikh, Neal, Chu, Eric Peleato, Borja, and Eckstein, Jonathan. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2010.

Burges, Christopher. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998.

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

Chechik, Gal, Sharma, Varun, Shalit, Uri, and Bengio, Samy. Large scale online learning of image similarity through ranking. *JMLR*, 2010.

Cotter, Andrew, Shalev-Shwartz, Shai, and Srebro, Nathan. Learning optimally sparse support vector machines. *ICML*, 2013.

Dalal, Navneet and Triggs, Bill. Histograms of oriented gradients for human detection. *CVPR*, 2005.

Davis, Jason, Kulis, Brian, Jain, Prateek, Sra, Suvrit, and Dhillon, Inderjit. Information-theoretic metric learning. *ICML*, 2007.

Frome, Andrea, Singer, Yoram, and Malik, Jitendra . Image retrieval and classification using local distance functions. *NIPS*, 2006.

Guillaumin, Matthieu, Verbeek, Jakob, and Schmid, Cordelia. Is that you? metric learning approaches for face identification. *ICCV*, 2009.

Hoi, Steven, Liu, Wei, and Chang, Shih-Fu. Semi-supervised distance metric learning for collaborative image retrieval. *CVPR*, 2008.

Huang, Gary, Mattar, Marwan A., Lee, Honglak, and Learned-Miller, Erik. Learning to align from scratch. *NIPS*, 2012.

Kedem, Dor, Tyree, Stephen, Weinberger, Kilian, and Sha, Fei. Non-linear metric learning. *NIPS*, 2014.

Lecun, Yann, Bottou, Leon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.

Lim, Daryl and Lanckriet, Gert. Efficient learning of mahalanobis metrics for ranking. *ICML*, 2014.

Lowe, David. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

Malisiewicz, Tomasz, Gupta, Abhinav, and Efros, Alexei. Ensemble of exemplar-svms for object detection and beyond. *ICCV*, 2012.

O'Donoghue, Brendan, Chu, Eric, Parikh, Neal, and Boyd, Stephen. Operator splitting for conic optimization via homogeneous self-dual embedding. *arXiv:1312.3039*, 2013.

Wang, Jun, Woznica, Adam, and Kalousis, Alexandros. Parametric local metric learning for nearest neighbor classification. *NIPS*, 2012.

Weinberger, Kilian and Saul, Lawrence. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009.

Xiang, Shiming, Nie, Feiping, and Zhang, Changshui. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 2008.