Chapter 1

Introduction

The need to automatically decide whether, and/or to what extent, two objects are similar arises in many areas of computer science. Sometimes it is explicit, for instance in nearest neighbor methods that rely on finding training instances similar to the input, or in information retrieval applications. In other cases, for instance in probabilistic models that use dissimilarity computations to derive model parameters, this need is implicit. The notion of similarity judgment has been also in the focus of a large body of research in cognitive science. It is known that people can perceive and judge similarity at different cognitive levels, and that the semantics of such judgments may depend on the task.

The basic idea explored in this thesis is that the notion of task-specific visual similarity can be, and should be, learned from examples of what is to be considered similar for a given task. Specifically, we develop an new approach that learns an *embedding* of the data into a metric space where a (possibly weighted) Hamming distance is highly faithful to the target similarity. A crucial practical advantage of this approach is that a search for examples similar to a given query is reduced to a standard search in the metric embedding space and thus may be done extremely quickly, leveraging an arsenal of randomized algorithms developed for that purpose. In some of the applications reported here we use our embedding approach in conjunction with locality sensitive hashing, and achieve state-of-the-art performance in sublinear time.

We develop a family of algorithms for learning such an embedding. The algorithms offer a trade-off between simplicity and speed of learning on the one hand and accuracy and flexibility of the learned similarity concept on the other hand. We then describe two applications of our similarity learning approach in computer vision: for a regression task of estimating articulated human pose from images and videos, and for a classification task of matching image regions by visual similarity. To our knowledge, this is the first example-based solution to these problems that affords a feasible implementation.

In the context of regression, the novelty of our approach is that it relies on learning an embedding that directly reflects similarity in the target space. We can use this embedding to retrieve training examples in which the target function with high probability has values similar to the value on the input point. We combine the embedding

with the search algorithm using randomized hashing and with a clustering step that allows for multi-modal estimation.

This Introduction is organized as follows. Section 1.1 gives defines more formally the task we are addressing. Section 1.2 outlines the basic ideas in our approach to learning similarity. The computer vision applications of this approach are briefly described in Section 1.3. Finally, Section 1.4 describes the organization of the remainder of the thesis.

1.1 Modeling equivalence

The central learning problem addressed in this thesis can be formulated as follows. Let $\mathcal{X}$ denote the *data space* in which the examples are represented. We will define an *equivalence similarity* concept as a binary relation $\mathcal{S}(\mathbf{x}, \mathbf{y}) \rightarrow \pm 1$, that specifies whether two objects $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{X}$ are similar (+1) or not (-1). We will assume, unless noted otherwise, that this relation is reflexive, i.e. $\mathcal{S}(\mathbf{x}, \mathbf{x}) = +1$, and symmetric, i.e. $\mathcal{S}(\mathbf{x}, \mathbf{y}) = \mathcal{S}(\mathbf{y}, \mathbf{x})$. However we will not require $\mathcal{S}$ to be transitive, and so it will not necessarily induce equivalence classes on $\mathcal{X}$.

We develop an approach to learning a model of such similarity relation *from examples* of pairs that would be labeled similar, and ones that would be labeled dissimilar, by $\mathcal{S}$. We also show how, under certain assumptions, such learning can be done in a scenario in which *only positive* examples are provided, in addition to some unlabeled data.

Our objective in learning similarity is dual:

- To develop a *similarity classifier*, that is, to build an estimator that given a novel pair of objects in $\mathcal{X}$ predicts, as accurately as possible, the label $\mathcal{S}$ would have assigned to it.

- To provide framework for a very efficient *similarity search*. Given a large database $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of examples and a query $\mathbf{x}_0$ we would like to have a method for retrieving examples in the database that are similar (with respect to $\mathcal{S}$) to the query, *without* having to apply the similarity classifier to every possible pair $(\mathbf{x}_0, \mathbf{x}_i)$.

The embedding approach developed in this thesis allows us to achieve both of these goals in a single learning framework.

1.1.1 Other notions of similarity

Similarity can be defined at two additional levels of "refinement", which we do not address here. However we describe these notions of similarity below in order to clarify the distinction from the problem outlined above.

Ranking One can define a relative similarity: for two pairs of examples, $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}, \mathbf{w}$ one can determine whether or not $\mathcal{S}(\mathbf{x}, \mathbf{y}) \leq \mathcal{S}(\mathbf{z}, \mathbf{w})$. In principle such similarity model defines a binary classification problem on *pairs of pairs* of examples.

Distance At the most refined level, $\mathcal{S}$ could produce a non-negative real number for any pair of examples; the smaller this number the more similar the two examples are. Such a regression mapping $\mathcal{X}^2 \rightarrow \mathbb{R}_+$ corresponds to the standard notion of a distance between pairs. The distance values of course induce a ranking relation, as well. It may also be possible to obtain a consistent set of distances from ranking, by methods like multidimensional scaling (Section 2.3.3) but in general the information available at this level is more rich than the other two.

In this thesis, unless otherwise noted the term "similarity" will refer to equivalence. At the end of the thesis we will discuss how the approach we develop could be extended to the ranking notion of similarity. As for learning a real-valued, distance notion of similarity, we will not pursue it here.

1.1.2 Example-based methods

In some cases, the goal of an application is explicitly to predict the similarity judgment on two examples $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, under a particular similarity $\mathcal{S}$. This is a *classification* problem over the space of pairs $\mathcal{X} \times \mathcal{X}$. However, very often in machine learning the ability to automatically judge similarity of example pairs is important not in itself, but as part of an example-based method.

The distinction between model-based and example-based classification is often loose; here we attempt to frame it in terms of the manner in which training examples are used to predict the label of a new input.

Model-based methods use the training examples to build a model–of the class or of the target function. More often than not the model is parametric. A common example is to fit a parametric model of probability density to examples from each class; the very popular family of classification methods based on principal component analysis belongs to this kind. Sometimes it is non-parametric, for instance modeling the class-conditional density with a kernel estimate. The main defining characteristic of a model-based classification is that the input is not explicitly matched with (compared to) individual training examples but rather matched to the model. This is true whether the original training examples are kept around, like in the case of kernel-based non-parametric model, or are discarded after the model is constructed as in principal component analysis, or a mix of the two is used, as in a support vector machine (SVM) [103], where some of the training examples, namely the support vectors, are retained in addition to a set of parameters learned from the entire training set.

In contrast, in example-based methods classification or regression is based explicitly on comparing the input to individual training examples. Widely used example-based methods include nearest-neighbor classification and locally-weighted regression. A generic description of such a method is:

1. Store the training (sometimes called reference) data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and the associated labels $\ell_1, \ldots, \ell_n$.

2. Given a query $\mathbf{x}_0$, find examples $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ in X that are similar to $\mathbf{x}_0$.

3. Infer the label of the query ℓ_0 from $(\mathbf{x}_{i_1}, \ell_{i_1}), \ldots, (\mathbf{x}_{i_k}, \ell_{i_k})$.

The central computational task in the above description is *similarity search*: Given a query $\mathbf{x}_0 \in \mathcal{X}$ and a set of examples $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, find $\mathbf{x}_i$ such that $\mathcal{S}(\mathbf{x}_0, \mathbf{x}_i) = +1$. Technically, this may be equivalent to applying a similarity classifier on n pairs $(\mathbf{x}_0, \mathbf{x}_i)$, $i = 1, \ldots, n$. However, from a practical standpoint such a solution is unacceptable for large datasets, even with a relatively simple classifier. In order to make search feasible, it should be possible to complete in time sublinear in n.

We assume that no analytic expression exists for the "true" similarity concept $\mathcal{S}$, or that no access to such an expression is given to us. Thus we need to construct a *model* $\widehat{\mathcal{S}}$ of similarity, which will be used to predict the values of $\mathcal{S}$.

1.1.3 Why learn similarity?

Before we discuss the details of our approach to learning similarity from data, we briefly discuss some alternatives here, and a more detailed discussion is found in Chapter 2.

A reasonable approach may be to use a distance as a proxy for the desired similarity, namely,

$$\widehat{\mathcal{S}}_{\mathcal{D}}(\mathbf{x}, \mathbf{y}) \;=\; \begin{cases} +1 & \text{if } \mathcal{D}(\mathbf{x}, \mathbf{y}) \leq R, \\ -1 & \text{if } \mathcal{D}(\mathbf{x}, \mathbf{y}) > R. \end{cases} \tag{1.1}$$

The choice of the distance $\mathcal{D}$, and to some extent of the threshold R, may have critical impact on the success of such a model. The most commonly used distances are the L_p metrics, in particular the L_1 (or Manhattan) and the L_2 (Euclidean) distances. These distances account for a vast majority of example-based methods proposed in computer vision when the representation space $\mathcal{X}$ is a vector space of fixed dimension.[1] When the representation does not allow a meaningful application of L_p, the similarity is typically measured in one of two ways. One is to *embed* the data into a metric space and proceed using an L_p distance; the other is to apply a distance measure suitable for $\mathcal{X}$. For instance, when examples are sets of points in a vector space, a common distance to use is the Hausdorff distance [45] or the earth mover's distance [55]. Often one uses an embedding of $\mathcal{X}$ into another, usually higher-dimensional space, in which an L_p metric approximates the complex distance in the original space [5, 55].

However, it is usually possible to provide *examples* of similarity values. The source of such examples depends on the circumstances in which similarity modeling is required. In some cases, similarity values for example pairs may be provided directly, either by manual labeling or via an automated data generation or gathering process. In colloquial terms, this means that a human has a particular concept of similarity in mind, such as "these two image patches look similar", or "these two people have a similar body pose", that allows him/her to serve as an oracle and provide values of $\mathcal{S}(\mathbf{x}, \mathbf{y})$ for some pairs $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$. These values are considered the "ground truth" and the goal of the similarity modeler is to construct an estimate, $\widehat{\mathcal{S}}$, that optimizes

[1] The distances under L_1 and L_2 will of course differ, however the ranking, and consequently the set of nearest neighbors, are typically very similar; see, e.g., [52] for a discussion.

the chosen measure of agreement with $\mathcal{S}$. In other words, the goal is to "uncover" the similarity judgment $\mathcal{S}$ used to assign the training labels.

On the other hand, in the context of example-based methods similarity between objects in $\mathcal{X}$ is in effect a "latent concept". Each training example $\mathbf{x}$ in $\mathcal{X}$ is associated with a *label* $\ell(\mathbf{x})$ in a target space $\mathcal{Y}$. Usually, a well-defined similarity $\mathcal{S}_\mathcal{Y}$ exists over $\mathcal{Y}$ and can usually be computed analytically. For instance, in a classification scenario $\mathcal{Y}$ is the finite set of class labels, and two labels are similar if they are identical. In a regression setting $\mathcal{Y}$ contains the values of the target function, and similarity may be defined by two values falling within a certain distance from each other. We suggest a natural protocol for defining a similarity over $\mathcal{X}^2$: two examples in $\mathcal{X}$ are considered to be similar under $\mathcal{S}$ if their labels are similar under $\mathcal{S}_\mathcal{Y}$. This provides us with a method for inferring values of $\mathcal{S}$ from the labels. The basic challenge remains unchanged: to be able to predict $\mathcal{S}(\mathbf{x}, \mathbf{y})$ without access to the labels $\ell(\mathbf{x}), \ell(\mathbf{y})$ and thus to the ground truth similarity.

A crucial property of similarity is that it can be *task-specific*: the same two examples may be judged similar for one purpose and dissimilar for another. This is illustrated by the following "toy" example. Consider a set of 2D points, with two different notions of similarity illustrated in Figure 1-1 (analyzed in more detail in Chapter 3.) Under the first similarity (top row), two points are similar if their Euclidean norms are close (within a given threshold). Under the second, two points are similar if the angles in their polar coordinates (modulo π) are close. Clearly, Euclidean norms, Manhattan or Mahalanobis distances are not adequate here. The proposed algorithm uses a few hundred examples of pairs similar under the relevant similarity and produces an embedding which recovers the target concept quite well, as shown on the right.

1.2 Learning embeddings that reflect similarity

In the most basic form, our approach can be summarized as follows. We construct an embedding of $\mathcal{X}$ into an M-dimensional space $\mathcal{H}$, each dimension m of which is given by a separate function h_m:

$$H \,:\, \mathbf{x} \in \mathcal{X} \;\rightarrow\; [\alpha_1 h_1(\mathbf{x}), \ldots, \alpha_M h_M(\mathbf{x})], \qquad h_m(\mathbf{x}) \in \{0, 1\}. \tag{1.2}$$

The value of $\alpha_m > 0$ depends on the specific algorithm, but in all algorithms the h_m are chosen in such a way that the L_1 distance

$$\|H(\mathbf{x}) - H(\mathbf{y})\| \;=\; \sum_{m=1}^{M} |\alpha_m h_m(\mathbf{x}) - \alpha_m h_m(\mathbf{y})|.$$

reflect the underlying similarity. That is, the lower the distance $\|H(\mathbf{x}), H(\mathbf{y})\|$, the higher the certainty of $\mathcal{S}(\mathbf{x}, \mathbf{y}) = +1$. Thus, we follow the paradigm of distance as proxy for similarity (1.1), however the representation, the distance and the threshold R are explicitly chosen with the objective of maximizing the prediction accuracy.

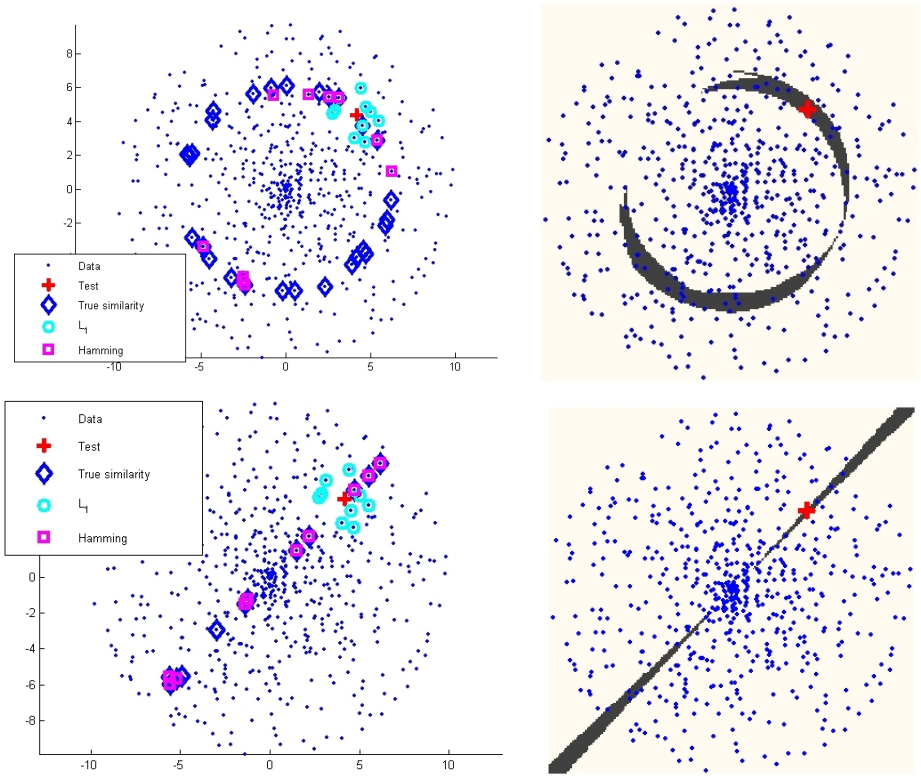


Figure 1-1: Illustration of task-specific similarity modeling on a toy 2D data set. Left: ground truth showing, for one query (cross), examples similar to it (diamonds). Examples found by the BoostPro (Chapter 3) algorithm are shown by squares. Right: similarity regions induced by the query and the embedding learned with BoostPro (200 bits), for a particular distance. Top row: norm similarity, bottom row: angle similarity.

1.2.1 Motivation: hashing and boosting

This approach is inspired by, and to a large extent has evolved from ideas developed in the last decade in two areas of research: randomized search algorithms in computational geometry and ensemble methods in machine learning. Here we briefly describe them, and a more detailed survey can be found in Chapters 2 and 3.

Locality sensitive hashing (LSH) The LSH [65, 52, 31] is a scheme for approximate similarity search under the L_p metric for $p \in [0, 2]$. It works by indexing the data in a set of l hash tables with independently constructed randomized hash functions, each using a key of k bits. Each bit in the hashing key is computed by projecting the data onto a random vector and thresholding the value. With a suitable setting of parameters l and k, this hashing scheme finds, for a value R, a ϵ-R neighbor of $\mathbf{x}_0$, i.e., an example $\mathbf{x}$ such that $\|\mathbf{x}_0 - \mathbf{x}\| \leq (1 + \epsilon)R$. Its lookup time is $O\left(n^{1/(1+\epsilon)}\right)$, and arbitrarily high probability of success can be achieved. The building block of LSH which provides this guarantee is the notion of a *locality sensitive* hash function, under which the probability of collision is related to the distance in $\mathcal{X}$. When the L_p metric over $\mathcal{X}$ is used as a proxy for the underlying similarity $\mathcal{S}$, the LSH achieves our goal as formulated: the union of distinct bits used in the hash keys defines an embedding in which L_1 distance (in this case equivalent to the Hamming distance) reflects $\mathcal{S}$. A natural question, then, is how to extend the LSH framework to reflect the distance in the unknown embedding space. Our solution is, essentially, to *learn* the locality-sensitive bits and let the bits define the embedding.

Boosting The idea of boosting [99, 23] is to create an ensemble classifier (or regressor) by greedily collecting simple classifiers that improve the ensemble performance. Each simple classifier only has to be better than chance, hence it is often referred to as a "weak" classifier. A number of variants of boosting have been published so far; in Chapter 3 we review the specific boosting algorithms relevant to our work. The general strategy shared by boosting methods is to assign weights to the training examples and manipulate these weights in order to steer the iterative greedy selection process towards improving the desired properties of the ensemble.

The learning approach in this thesis was inspired by these ideas, and has adapted them for the purpose of constructing a similarity-reflecting embedding. The algorithms outlined below and described in detail in Chapter 3. The order in which they are presented corresponds to the evolution of the underlying ideas and to trading off simplicity of learning for representational power of the resulting embeddings.

1.2.2 Similarity sensitive coding

The first algorithm[2] is essentially a modification of the original LSH approach in which the hashing bits correspond to axis-parallel decision stumps. The operating assumption behind it is that a reasonable approximation to $\mathcal{S}$ may be obtained by calculating the L_1 distance in the data space $\mathcal{X}$, when the following "corrections":

[2]Published in [105].

1. Some dimensions of $\mathcal{X}$ may be irrelevant for determining $\mathcal{S}$. These dimensions serve as noise when distance is computed, and are better ignored.

2. For a given dimension, some thresholds (decision stumps) are much more effective (i.e. similarity-sensitive–see Section 2.4.2) than others. Using these thresholds in constructing LSH keys will optimize the properties of the hashing scheme for a given size of the data structure (and thus for a given lookup time.)

3. The determination of the dimensions and thresholds described above is to be guided by the available training data in the form of similar and dissimilar pairs of points in $\mathcal{X}$. The training true positive (TP) rate correspond to the percentage of similar pairs in which both examples (projected on the dimension at hand) fall on the same side of the threshold. The false positive (FP) rate is evaluated similarly by looking at the dissimilar pairs.

This leads to the algorithm called similarity sensitive coding (SSC), first presented in [105] under the name of PSH (parameter-sensitive hashing). For each dimension of $\mathcal{X}$, SSC evaluates the thresholds and selects the ones with acceptable combination of TP and FP rate. The criteria of acceptability depend on the precision/recall rates appropriate for the application at hand, and are formulated as an upper bound on the FP and a lower bound on the TP rates. The data are then indexed by LSH, which uses only the selected stumps as hash key bits. This is equivalent to embedding $\mathcal{X}$ into a binary space

$$H^{\mathrm{SSC}}(\mathbf{x}) \;=\; \left[h_1^{\mathrm{SSC}}(\mathbf{x}), \ldots, h_M^{\mathrm{SSC}}(\mathbf{x})\right], \tag{1.3}$$

where each bit $h_m^{\mathrm{SSC}}(\mathbf{x})$ is obtained by quantizing a single dimension i_m in $\mathcal{X}$ into a single bit by thresholding:

$$h_m^{\mathrm{SSC}}(\mathbf{x}) \;=\; \begin{cases} 1 & \text{if } x_{i_m} \leq T_m, \\ 0 & \text{if } x_{i_m} > T_m. \end{cases}$$

1.2.3 Boosting the embedding bits

Learning of the decision stumps in SSC is straightforward, and the algorithm has produced good results in the pose estimation domain [105, 35]. However, SSC leaves room for a major improvement: it ignores dependencies between the dimensions of $\mathcal{X}$. The second algorithm of Chapter 3 addresses these issues and employs a boosting algorithm (AdaBoost) which takes the dependencies into account. The boosting algorithm yields an ensemble classifier,

$$C^{\mathrm{AB}}(\mathbf{x}, \mathbf{y}) = \operatorname{sgn}\left[\sum_{m=1}^{M} \alpha_m \left(h_m^{\mathrm{AB}}(\mathbf{x}) - 1/2\right)\left(h_m^{\mathrm{AB}}(\mathbf{y}) - 1/2\right)\right] \tag{1.4}$$

where the single bit functions h_m^{AB} are of the same form as h_m^{SSC}. The resulting embedding is into a weighted binary space

$$H^{\mathrm{AB}}(\mathbf{x}) \;=\; \left[\alpha_1 h_1^{\mathrm{AB}}, \ldots, \alpha_M h_M^{\mathrm{AB}}\right]. \tag{1.5}$$

Interestingly, the L_1 (Hamming) distance in this space between $H^{\mathrm{AB}}(\mathbf{x})$ and $H^{\mathrm{AB}}(\mathbf{y})$ is proportional to the *margin* of the AdaBoost classifier,

$$\sum_{m=1}^{M} \alpha_m \left(h_m^{\mathrm{AB}}(\mathbf{x}) - 1/2\right) \left(h_m^{\mathrm{AB}}(\mathbf{y}) - 1/2\right).$$

In practice, this algorithms may outperform SSC for a number of reasons:

- The embedding is less redundant and more directly optimized for the underlying similarity prediction task.
- The weights produced by AdaBoost allow for an additional "tuning" of the embedding.

While in principle this is a straightforward application of AdaBoost, a number of interesting practical problems arise when the algorithm is applied to a large amount of data. In particular, under the assumption mentioned in Section 2.1.4 that similarity is a "rare event", the class distribution is very unbalanced. We discuss this issue in Chapter 3.

1.2.4 BoostPro: boosting optimized projections

The final algorithm of Chapter 3, called BoostPro, further advances our approach towards making the embedding more flexible. We leave the realm of axis-parallel decision stumps, and instead propose to use arbitrary *projections* of the data. By a projection we mean any function $f : \mathcal{X} \to \mathbb{R}$; in all the experiments described in this thesis we have used polynomial projections,

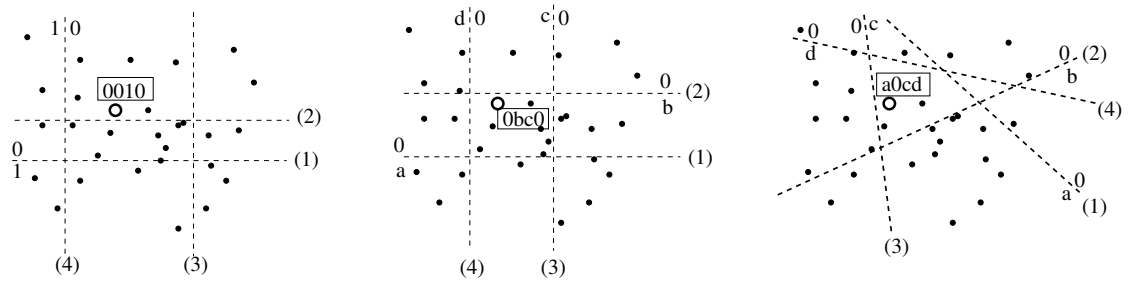
$$f(\mathbf{x}) \;=\; \sum_{j=1}^{d} \theta_j x_{i_j}^{p_j}, \qquad p_j \in \{1, 2, \ldots\}, \qquad i_j \in \{1, \ldots, dim(\mathcal{X})\}.$$

In contrast to the algorithm outlined in the previous section (where the weak learners only select the threshold), BoostPro uses a gradient-based optimization procedure in the weak learners to improve projection coefficients as well as thresholds, given the current ensemble and the weights on the training data. Furthermore, we introduce a modification of AdaBoost algorithm for the special case of learning similarity, in which learning is done from positive examples only.

Figure 1-2 provides a cartoon illustration of the forms of embedding attainable with each of the algorithms.

1.2.5 Relationship to other similarity learning methods

In Chapter 2 we discuss in some detail the significant body of literature devoted to related topics. Here we attempt to broadly categorize the prior work and emphasize its main differences from the learning approach developed in this thesis.



(a) SSC (b) Boosted SSC (c) BoostPro

Figure 1-2: Illustration of embeddings obtained with the learning algorithms. Dotted lines show 0/1 boundaries for each bit. Letters correspond to weights, numbers in parenthesis to the order of the bits. Shown in the box is the embedding of the query point (circle). In (c), the case of linear projections is illustrated; for polynomial projections of higher order the boundaries would be nonlinear.

Metric learning Much work has been done on learning a *metric* $\mathcal{D}$ on $\mathcal{X}$ that is optimized for the use in a particular learning machine, typically a NN classifier. The requirement that the distance be a metric (including transitivity and compliance with triangle inequality) stands as a major difference with our approach. Furthermore, typically the learned metric is constrained to a particular parametric form, usually described by a quadratic form [118, 53]. Thus the class of similarity concepts attainable by these methods is significantly more limited in comparison to our embeddings.

Optimal distance learning For certain classification tasks there have been proposed algorithms that learn a distance (as a measure of dissimilarity) which is not necessarily a metric, optimized for a particular task—classification or clustering. Among recent work in this direction, [79] and [60, 61] are the closest in spirit to ours. However, the transitivity requirement is retained in these approaches, and it is not clear how to extend them effectively beyond problems with finite label sets.

Manifold learning Many algorithms have been proposed for learning a low-dimensional structure in data, under the assumption that the data lie on a (possible non-linear) manifold: multidimensional scaling (MDS) [27], Isomap [112], local linear embedding [96] and others (see [12] for a unifying perspective on these and other manifold learning algorithms.) These algorithms usually obtain an embedding of the training data by manipulating the eigenvectors of the pairwise distance matrix. A related family of methods deals with embedding a graph, whose vertices represent examples and edges are weighted by (dis)similarity, in a space where the similarities are preserved.

In contrast to the manifold learning algorithms, our approach does not make an implicit assumption regarding structure in the data, nor does it limit the dimension-

ality of the embedding by the dimensionality of $\mathcal{X}$.[3] A more important difference, however, has to do with extending the embedding to new examples. The MDS and related algorithms do not yield a *mapping function*, which could be applied to a previously unseen example. While some extensions to out-of-sample examples have been proposed [12, 33], they typically rely on the (Euclidean) distance in $\mathcal{X}$ and the ability to find neighbors efficiently among training data–an undesirably circular dependency in the context we are considering here.

1.3 Applications in computer vision

1.3.1 Levels of visual similarity

Visual similarity can be defined at a number of perceptual levels, which differ in the amount of semantic complexity, the dependence on the task at hand, and the potential stages in the visual pathway at which they may be implemented in biological vision systems.

Low-level similarity Two image regions (*patches*) are considered visually similar if they correspond to similar physical scenes. A simple example of this occurs under small motions (translation and rotation) of a camera pointed at a given scene: in most cases, unless there is a discontinuity in appearance due, for examples, to sharp edges, images of the scene in subsequent frames will be similar. The framework developed in this thesis will be applied to learn such similarity – specifically, to predict when two image patches are transformed versions of each other. In essence, the goal is to obtain transformation-invariant similarity on top of non-invariant representation. The learning for this kind of similarity can occur with no human supervision: given a set of natural images, pairs of similar patches can be extracted automatically.

Mid-level similarity On a higher perceptual level (which may be associated with later stages in the visual pathway) visual elements are deemed similar if they share some simple semantic property. An example of such similarity that arises in the object categorization domain is the notion of *parts* - elements that are repeatable in a particular visual category, albeit with some appearance variation. This level of similarity may be important, in particular in an object classification architecture with multiple feature levels.

High-level similarity On an even higher perceptual level, similarity is defined primarily by semantics. These properties that make two objects similar are themselves not visual, but can be inferred (by human perception) from visual information. Two examples of such *task-specific* similarity that we consider in this thesis are object category (do the two objects belong to the same category?) and articulated human

[3] Although directly comparing dimensionalities is somewhat inappropriate, since our embedding space is (possibly weighted) binary, as opposed to Euclidean $\mathcal{X}$.

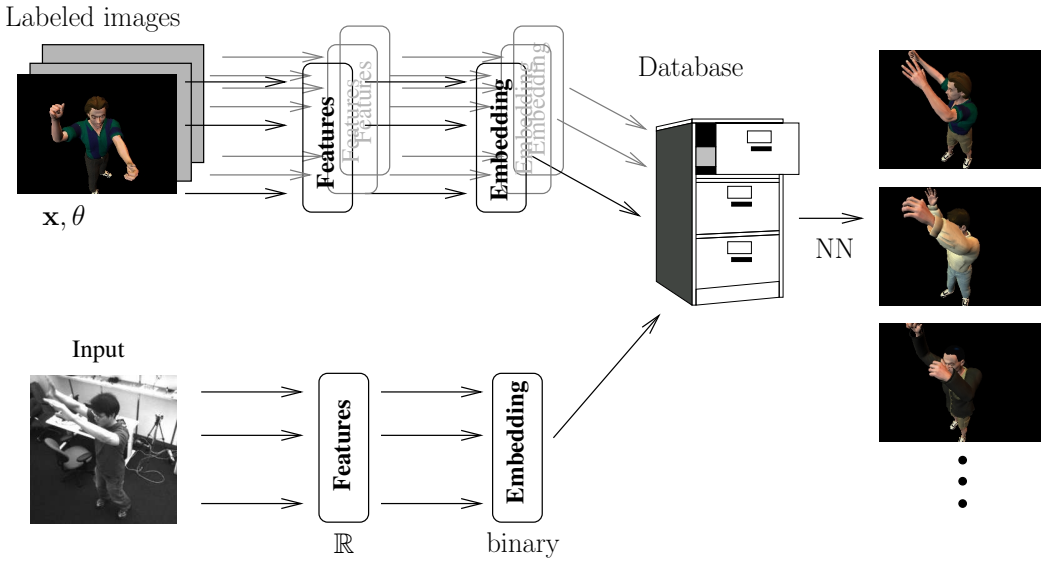


Figure 1-3: A cartoon of the example-based pose estimation approach. The embedding is learned to reflect similarity of the entire pose.

pose estimation (is the body configuration of the two human figures similar?). Note that in the latter case, there is an additional level of dependency on the exact task: two poses that may be judged similar if one only needs to classify a gesture (pointing versus raising one's hand) would not be considered similar if the goal is to recover the 3D location of every body joint with maximal precision.

1.3.2 Example-based pose estimation

Previous model-based approaches have shown that the task of modeling the global relationship between the image and the pose is very difficult. In the proposed approach, we instead model a simpler concept: similarity between the poses that appear in two images. This leads to an example-based estimation algorithm: given an image, find in a large database of images (labeled with the underlying articulated poses) examples classified as similar to the input. This scheme, illustrated in Figure 1-3, relies on the performance of the similarity classifier. Its high true positive (TP) rate provides that with high probability, the unknown pose is close to the poses in the retrieved examples. On the other hand, the low false positive (FP) rate means that not many spurious examples will be retrieved.

A preliminary work in this direction, using SSC, has been presented in [105]. In this thesis we present new experiments with a very large database of poses, obtained with motion capture system, using BoostPro to learn an embedding of images that reflects pose similarity.

1.3.3 Learning visual similarity of image regions

Comparing image regions is a basic task which arises in many computer vision problems: analysis of stereo, image denoising, scene recognition, object categorization etc. Recently, methods that operate by comparing image regions have established themselves as state-of-the-art in some of these problems. Conceptually, there are usually four steps in such methods that directly operate on image regions:

1. *Interest operator*: selecting a set of regions from the given image that are considered "interesting". This is an attention mechanism, and a number of such operators have been proposed. While some appear to be particularly successful in certain cases [77, 84], the choice of interest operator and even its utility is still far from obvious [80, 14], and we will remain agnostic regarding this issue.

2. *Descriptor* The next step is to compute the representation of the selected patches. Ideally, the representation should capture the features that are important to the application that uses the matching method, while being invariant to features that are unimportant. We will consider two representations, that have been the subject of much work in the vision community: the shift-invariant feature transform (SIFT) [77] and the sparse overcomplete codes [89].

3. *Matching* Once the descriptor for a region is computed, it is matched to the descriptors of regions in the database (the training data).

4. *Inference* Depending on the specific task and the method at hand, the results of the matching across the test image are combined to produce an answer.

The matching step clearly provides a natural grounds for applying our learning approach. In Chapter 6 we describe an experiment in which we learn to match patches obtained by transforming an image in certain ways (rotations and mild translations), and show how whereas standard distance-based similarity models fail, the embedding learned by our algorithm allows to detect similarity between transformed versions of the same patches.

1.4 Thesis organization

Chapter 2 provides the background for the thesis research. It describes the prior work in related areas, with particular emphasis on the two ideas that inspired our learning approach: locality sensitive hashing and the boosting. Chapter 3 describes the core machine learning contribution of the thesis–a family of algorithms that produce similarity-reflecting embeddings of the data. Armed with these algorithms we develop example-based approaches for two computer vision domains. In Chapter 4 we describe a method for estimating articulated pose of human figure from a single image, and in Chapter 6 a method for matching image regions based on visual similarity under certain class of transformations. Chapter 7 contains a discussion of the presented approach, and outlines the most important directions for future work.