Learning Transformations Between Directed Subspaces Online

Adam Smith
University of California, Santa Cruz
CMPS 290C, Advanced Topics in Machine Learning
amsmith@cs.ucsc.edu

June 15, 2007

Abstract

In this project we tackle the rotation problem (learning the rotation between instance and label vectors in an online setting) by developing a novel problem setting. We adopt a geometrically-motivated approach in the framework of the univeral geometric algebra, a powerful mathematical tool for manipulating directed subspaces. Using this approach we mechanically derive a remarkably simple and approximation-free algorithm for learning rotations in arbitrary spaces. Finally, we explore several standard transformations that can greatly expand the power of a solution to the rotation problem.

1 Introduction

In supervised learning, the problem is to infer a function from example data that performs desirably on new data. This problem can be approached in either a batch or online setting, and we will focus on the (seemingly harder) online setting here. In classification prorblems, we are told that the unknown function mapping the data has a codomain of class identifiers (possibly just 'yes' or 'no'). However, there is an important class of problems where the codomain of the unknown function, which we conventionally call the space of labels, is same as the space of data points, conventionally called the space

of instances. We call these problems subspace transformation problems because they are equivalent to learning a transformation from one subspace to another. In particular, we will focus on the rotation problem, learning an unknown rotation from instances to labels that may shift over time. To attack this problem, we develop a powerful new problem setting and a simple, geometrically-motivated approach that can be used to derive solutions to various problems in the new setting.

2 Transformed Subspaces Setting

First, let us consider a new concept, the *motherspace* $\mathcal{M}$. The motherspace is a construct we use to motivate our approach, but is not necesary for understanding out final results. The motherspace is an infinite dimensional vector space that includes every subspace of instances and labels we might consider. By formulating our problems in this single space, we can discuss a large space of problem classes all at once.

Next, consider the space of all linear functions $f(\cdot)$ on the motherspace. In a machine learning context, we can identify each one of these functions as mappings $\mathbf{y_t} = f(\mathbf{x_t})$ from instances to labels as the unknown function underlying a particular problem in our setting. We focus on the case where we are promised that these functions are chosen from specific, restricted classes of linear functions. Additionally, in the online setting, we assume the function used to map instances to labels may change over time (necesitating the need to develop online algorithms to track this behaviour).

2.1 Definition of Problem Setting

Now we can formulate a clean definition of general problems in the transformed subspaces setting. This solution will serve as a template for instantiating an algorithm for the rotation problem when we focus on it later in this paper.

Online Homogeneous Transformation Problem

1. Receive an instance vector $\mathbf{x_t}$ from the instance space.
2. Predict a label vector $\hat{\mathbf{y}}_\mathbf{t}$ in the motherspace.
3. Receive true label vector $\mathbf{y_t} = f_t(\mathbf{x_t})$ in label space with $f_t(\cdot)$ from a specific class of linear functions.

4. Incur a loss $L_{\mathbf{y_t}}(\hat{\mathbf{y}}_\mathbf{t})$.

Note that the predicted laabel vector $\hat{\mathbf{y}}_\mathbf{t}$ need not necesarily live in the label space. In particular the definition of the label space is not explicity provided (and may even be the focus of the problem as it is in principle component analysis). Also note that we are considering a noise-free case. That is, there is always a particular f that exactly explaints the transformation from instance to label vectors. Again, in the transformation from application-level geometry we may also relax this contraint. Finally, we call this the *homogenous* transformation problem because any *linear* function $f : \mathcal{M} \to \mathcal{M}$ must be homogeneous. Intuitively, we can identify f with a transformation on entire homogeneous linear subspaces of the motherspace.

2.2 Solution Template

From our problem statement, we can formulate a general solution to homogeneous transformation problems that echos the standard skeleton for online algorithms. First, we should pick a class of linear functions (that may or may not be more specific than the class of unknown functions for the problem) and identify a parameterized form of this class. Initially, the algorithm should initialize its parameter $\mathbf{P}$ to some chosen initial value. These are our only creative choices in the derivation of an algorithm to solve the problem at hand. On each instance $\mathbf{x_t}$ recieved, the algorithm should predict $\hat{\mathbf{y}}_\mathbf{t} = f_\mathbf{P}(\mathbf{x_t})$. After a true label $\mathbf{y_t}$ is received, the algorithm should update the parameter in such a way as to minimize the tradeoff of the divergence of the parameter to its last (presumably succesfull) value and and the loss the new parameter would incur on the previous example. That is, we have the following equation where η is a tradeoff parameter conventionally called the learning rate.

$$\mathbf{P_{t+1}} = \inf_{\mathbf{P} \in \mathcal{P}} \{\Delta(\mathbf{P}, \mathbf{P_t}) \ + \ \eta L_{\mathbf{y_t}}(f_\mathbf{P}(\mathbf{x_t}))\} \tag{1}$$

3 Approach

So far, we have only setup the machine learning commitments of our algoritms and not yet commited to a specific mathematical framework in which we can actually peform computations. As such, we exercise our freedom to

choose an somewhat obscure but immensely powerful framework called geometric algebra. As we will see, this choice lends a healthy dose of clarity and expressiveness in the transformed subspaces setting. Geometric algebra is sometimes billed as algebra of directed subspaces, an ideal tool for our problem.

3.1 Geometric Algebra

The *universal geometric algebra* $\mathcal{G}$ (as defined in [1]) is an infinite-dimensional, real, Clifford algebra. The generating set for the algebra is an infinte dimensional vector space (the motherspace for us), and the basic operations are a geometric sum and a geometric product. The geometric sum is exactly the familiar add-the-components sum in vector algebra. The geometric product, however, is of a less familiar variety. The geometric product is an information preserving bilinear operator that captures the parallel and perpendicular parts of its operands in one quantity. The closure of the sum and product operations on the given vector space produces a space of objects called multivectors that (similar to quaternions) have a scalar part, a vector part, and higher order (bivector, trivector, etc.) parts that each have a clear geometric interpretation in the motherspace. Interestingly, all real algebras are subalgebras of $\mathcal{G}$ (complex algebras are embedded as purely real algebras as well). Thus, $\mathcal{G}$ contains a subalgebra just for our problem!

In the rest of this section we will describe familiar mathematical tools in the framework of geometric algebra by only introducing concepts as needed to reduce the overhead required to understand our solution in this largely unfamiliar framework. All of the concise expressions we work with have beautiful geometric intepretations that are, unfortunately, outside of the scope of this paper to discuss in detail. Before continuing, absorb just a few bits of notation. The justaposition $\mathbf{AB}$ signifies the (non-commutative) geometric product between two multivectors. The expression $\mathbf{A^{-1}}$ signfies the inverse of a multivector (always defined except when $\mathbf{A} = 0$). The inner ($\cdot$) and outer ($\wedge$) products are derivable from the definition of the geometric product and, for vectors, coincide with the definition of the familiar Euclidean dot product and Grassmann (progressive) outer product. Finally $\langle \mathbf{A} \rangle$ denotes an operation that selects only the scalar part of a multivector (related to a trace).

3.2 Transformations

We claimed that deriving an algorithm from our template in the previous section requires selecting a parameterized family of linear functions. Here we lay out the form a few restricted classes of transformations.

First, consider reflections. A reflection can be parameterized by a single vector $\mathbf{n}$ which can be interpreted as the normal to the *hyperplane* of reflection. The function takes the form $f_{\mathbf{n}}(\mathbf{x}) = -\mathbf{nxn^{-1}}$.

Rotations (our focus) are parameterized by a rotor $\mathbf{R}$. Rotors are a quaternion-like normalized package of a scalar and a bivector (in general, the geometric product of two unit vectors). Rotors concisely encode the two-dimensional plane of rotation and angle of rotation. Note that the notion of "axis of rotation" does not generalize to non-three-dimensional spaces. A rotation tranformation takes the form $f_{\mathbf{R}}(\mathbf{x}) = \mathbf{RxR^{-1}}$ (the similarity to the reflection case is due to rotations really being a pair of nested reflection operations).

We can also represent another kind of transformation, projection onto a subspace. The parameter of this transformation is $\mathbf{S}$, the outer product of linearly indepdent vectors spanning the space onto which the projection occurs. A projection transformation takes the form $f_{\mathbf{S}} = \mathbf{x} \cdot \mathbf{SS^{-1}}$ (conventionally, the inner product binds tighter than the geometric product). Note that presence of the inner product in this expression indicates that this tranformation is capable of destroying information (geometric algebra often leads to self-documenting expressions like this), a property that is actually desirable for projection.

Finally, consider a scaling tranformation. This tranformation has a single real scalar parameter λ and takes the form $f_\lambda(\mathbf{x}) = \lambda\mathbf{x}$. Scaling is not a very powerful transformation, but we include it here for variation. General linear transformations are indeed expressable, but we will not make use of them.

3.3 Divergences and Losses

For any of the parameters that we might choose of the above, it is necesary to select a parameter divergence $\Delta(\mathbf{P}, \mathbf{P_t})$. An obvious, geometrically-motivated options is the "norm squared of difference" $|\mathbf{P} - \mathbf{P_t}|^2$ which is derivable as a Bregman divergence from the norm operation. The norm is defined for all multivectors and for multivectors of similar grade (either only scalars, only vectors, only bivectors, etc.) it corresponds exactly to the fa-

miliar notion of Euclidean distance.

Another option, that we have not developed enough for general use, is an attempt at a "geometric relative entropy". The expression $\Delta(\mathbf{A}) = \frac{1}{2}\langle \mathbf{A} \ln \mathbf{A} - \mathbf{A} + 1 \rangle$ is also defined for all multivectors (in terms of the Taylor expansion). For rotors, this yields strange entropic-distance-from-the-identity-rotation that is proportional to $-\theta \sin\theta - 2\cos\theta$ that is convex on $[-\pi, +\pi]$. We have not yet derived a Bregman divergence for general multivectors based on this expression. As such, we will focus on the norm squared as the generator for our divergences in this discussion.

Finally, part of problem definition, the loss of the prediction, deserves some discussion with respect to our geometric algebra approach. In selecting a loss, we just need some notion of distance between vectors in the motherspace. Again, an obvious choice is the norm squared of difference. A geometric relative entropy may someday be suitable. In this discussion we will focus on the norm squared as the generator for our losses as well.

4 Rotation Problem

With the above notions formalized, we may now take any problem and derive an algorithm to solve it. Here, as an example, we focus our attention on the rotation problem. In the online rotation problem, instance as well as label vectors come from a specific n-dimensional vector space $\mathcal{V}$. At each step, a hidden rotation $\mathbf{C_t}$ correctly maps the instance in the example. The hidden rotation may shift over time.

4.1 Attack

As a summary of our solution, we will use rotations as the class of linear functions (exactly mirroring the problem definition). The norm squared of difference will serve as our parameter divergence as well as our loss function. Finally, we will use the indentity rotation as our initial parameter state.

In the language of geometric algebra, the above approach is codified as follows. Our parameter, the rotor $\mathbf{R}$ is an object with $\binom{n}{2} = n(n-1)/2$ degrees of freedom after normalization. The initial setting, is simply the concisely expressed identiy rotation: $\mathbf{R_1} = 1$. To predict, we will use a function of the form $f_{\mathbf{R}}(\mathbf{x}) = \mathbf{R}\mathbf{x}$. Note, this is called a one-sided rotation and has slightly different properties than the standard two sided rotation.

We have chosen it here to easier exploration of a new framework and our result should be reformulated for the two-sided case before use. The parameter update, $\mathbf{R_{t+1}} = \inf_{|\mathbf{R}|^2=1} \{|\mathbf{R} - \mathbf{R_t}|^2 + \eta|\mathbf{R}\mathbf{x_t} - \mathbf{y_t}|^2\}$ follows the solution template. After every update, we may interpret the new state of parameter by decomposing $\mathbf{R_{t+1}} = \exp(-\mathbf{B}\theta)$ where $\mathbf{B}$ is a bivector that encodes the plane (with handedness) of rotation and θ is the angle of rotation in radians.

4.2 Geometric Update

The derivation of the of the update will held our until the next section to keep the discussion transparent. However, to skip to the exciting part, we will now present the resulting update expression in its clean, compact form. Note, z is a normalizing scalar.

$$\boxed{\mathbf{R_{t+1}} = \frac{\mathbf{R_t} + \eta \mathbf{y_t}\mathbf{x_t}}{z}} \tag{2}$$

Without any leaps of creativity, this update can be read in English as the following: *The best new rotor is simply the normalized sum of the previous rotor and the shortest rotor that explains the mapping the last example weighted by the tradeoff factor.*

4.3 Discussion

Technically, the subexpression $\mathbf{y_t}\mathbf{x_t}$ is not a rotor but instead a spinor because it is unnormalized (it encodes a dilation as well). An effect of this is that the algorithm will effectively learn more from pairs of vectors that are longer. This is to be expected, however, because our tradeoff is in terms of distance between the prediction and the true label vectors and not in terms of angles. If the hidden function was not strictly a rotation as the problem defined (possibly with the addition of noise), this length sensitivity can be interpreted as the intuition that examples of longer vectors being transformed gives us greater confidence in the apparent rotation.

This algorithm is very simply to explain yet was mechanically derived from the problem setting and our solution template. It does not depend on dimensionality nor the statement in the problem definition that instances and labels come from the same space. As such, we may use this solution in much more general settings. Furthermore, we predict that it may have

distinct advantages over other algorithms that the problem statement might suggest if we had not already developed our approach seated in geometric algebra.

First, consider if we had used a rotation matrix the parameter to our algorithm. Matrices are explicitly defined in terms of their components, so necesarily the update step of the algorithm could not be expressed in a coordinate-free manner. After somehow combining the previous rotation matrix and a rotation matrix that explains the last rotation, it is not clear that the result is again a rotation matrix. This value may not have a clear geometric meaning until it is projected back to have the right properties. This projection would likely be operationalized as a Graham-Schmidt orthonormalization process could possibly undo learning progress the algorithm has made. In our approach, spinor normalization has no effect on the direction or magnitude of the rotation action. This stems from the fact that a $n \times n$ matrix has too many degrees of freedom to only represent rotations.

Next, consider an algorithm based on Euler angles. Euler angles usually refer to rotations about the three principle axes in three dimensions, but we can consider a generalizable version that tracked a rotation about each principle plane (the spans of any two different basis vectors). The update in this formulate, again, is necesarily not coordinate-free. It is impossible to interpret the state of the parameter without establishing a orthonormal reference frame. Furthermore, this approach would likely make heavy use of trigonometric functions in its update. Well known problems of gimbal-lock and non-uniform sensitivity to noise are also likely. Despite having the ideal number of degrees of freedom in their representation, Euler angles are too ridgid in their structure.

Finally, a common answer to the issues of gimbal-lock with Euler angles is quaternions. Quaternions, when normalized, have the ideal number of degrees of freedom and store very similar information to that of Euler angles. However, because quaternions have an extra component they are more flexible during manipulation. Quaternions are actually equivalent (up to naming conventions) to rotors in three dimensions. So, while it is exactly the properties of the quaternion that make it so convenient for working with rotations in three dimensions that make the rotor a suitable parameter in higher dimensions, the use of a literal quaternion would immediately lock us into only very low dimensional applications.

5 Derivation of the Update

With our context, motivation, and initial results clearly laid out, it is now safe to inspect the process by which we arrived at the parameter update for the rotor problem. The following derivation should be prefaced with the comment that while we have been touting the benefits of coordinate-free expressions for simpler interpretation, we nonetheless revert to coordinates here. This is not because this problem (or any in this setting) requires working with coordinates, but because we are not experienced enough with geometric calculus to carry out these computations with confidence in another way.

To begin, recall the expression for the ideal new rotor as translated from our solution template.

$$\mathbf{R_{t+1}} = \inf_{|\mathbf{R}|^2=1} \left\{ |\mathbf{R} - \mathbf{R_t}|^2 + \eta |\mathbf{R}\mathbf{x_t} - \mathbf{y_t}|^2 \right\} \tag{3}$$

Now, form the Lagrangian of this extremization problem. Recall that $|\mathbf{R_t}|^2 = 1$ because it is normalized after each update. Let $\mathbf{x} = \mathbf{x_t}$ and $\mathbf{y} = \mathbf{y_t}$ for now.

$$L(\mathbf{R}, \alpha) = |\mathbf{R} - \mathbf{R_t}|^2 + \eta |\mathbf{R}\mathbf{x} - \mathbf{y}|^2 + \alpha(|\mathbf{R}|^2 - 1) \tag{4}$$

Expand the distances using a general form of the law of cosines.

$$L(\mathbf{R}, \alpha) = |\mathbf{R}|^2 + |\mathbf{R_t}|^2 - 2\langle \mathbf{R}\mathbf{R_t} \rangle + \eta(|\mathbf{R}|^2\mathbf{x}^2 + \mathbf{y}^2 - 2\langle \mathbf{R}\mathbf{x}\mathbf{y} \rangle) + \alpha|\mathbf{R}|^2 - \alpha \tag{5}$$

Collect terms around $\mathbf{R}$s.

$$L(\mathbf{R}, \alpha) = (1 + \alpha + \eta\mathbf{x}^2)|\mathbf{R}|^2 - 2\langle \mathbf{R}\mathbf{R_t} \rangle - 2\eta\langle \mathbf{R}\mathbf{x}\mathbf{y} \rangle + (\eta\mathbf{y}^2 - \alpha + 1) \tag{6}$$

Now, to ease our burden when working with the coordinate form of the above expression, let us introduce an arbitrary orthonomal basis $\{\mathbf{e_1}, \ldots, \mathbf{e_n}\}$ and decompose $\mathbf{R} = a + \sum_{j,k} b_{jk}\mathbf{e_j}\mathbf{e_k}$. Likewise, let us define $\mathbf{x}\mathbf{y} = a_d + \sum_{j,k} b_{jk,d}\mathbf{e_j}\mathbf{e_k}$ for product of the labelled data. Here, a sum over j, k collects terms for $1 \leq j < k \leq n$ and a sum over i collects terms for $1 \leq i \leq n$.

$$\begin{aligned} L(\mathbf{R}, \alpha) = (1 + \alpha + \eta\mathbf{x}^2)(a^2 + \sum_{j,k} b_{jk}^2) \\ -2(aa_t + \sum_{j,k} b_{jk}b_{jk,t}) \\ -2\eta(aa_d + \sum_{j,k} b_{jk}b_{jk,d}) \\ +(\eta\mathbf{y}^2 - \alpha + 1) \end{aligned} \tag{7}$$

To carry out the extremization, find the differential with respect to $\mathbf{R}$ and set zero. We will begin with the scalar part of the new rotor.

$$\frac{\partial L}{\partial a} = 2a(1+\alpha) + \eta\mathbf{x^2}) - 2a_t - 2\eta a_d = 0 \tag{8}$$

Solving for a, observe that the following partial update is almost in the form of the full update discussed in the previous section.

$$a = \frac{a_t + \eta a_d}{1 + \alpha + \eta\mathbf{x}^2} \tag{9}$$

Repeat the same for each bivector component of the new rotor.

$$\frac{\partial L}{\partial b_{jk}} = 2b_{jk}(1+\alpha+\mathbf{x}^2) - 2b_{jk,t} - 2\eta b_{jk,d} = 0 \tag{10}$$

$$b_{jk} = \frac{b_{jk,t} + \eta b_{jk,d}}{1 + \alpha + \eta\mathbf{x}^2} \tag{11}$$

Observe that each of the components resulting from the update above is divided by a common expression that contains the unconstrained Lagrange multiplier α. Thus, we can always normalize the resulting spinor. Finally, if we reform the rotor $\mathbf{R}$ by summing all of the components together, we can recover the update in terms of the $\mathbf{R_t}, \mathbf{x}, \mathbf{y}$ as desired.

A intesting aside, that may someday lead to an alternative method of derivation, arises when we recall the plane-angle decomposition of the new rotor. Let $\exp(\mathbf{B_t})$ be the previous rotor and $\exp(\mathbf{B_d})$ be that rotor that explains the recent data. The rotor we have after the update can be decomposed similarly $\ln(\mathbf{R_{t+1}}) = \ln(\exp(\mathbf{B_t}) + \exp(\mathbf{B_d} + c(\eta))) - \ln(z)$. This appears to be a simple *soft-max* calculation where the final term ensures that the result has no scalar component. That is, that the result is always a pure bivector (the expected type for the generator of a rotor).

6 Extensions of Rotation Problem

We mentioned that a solution to the rotation problem could be very powerful. Here we will describe some methods for expanding the rotation problem to handle other problems without changing the algorithm at all. All extensions are made possible by transforming the application-level vectors before

and their processing by the rotation learning algorithm described above. Effectively, these map interesting subspaces at the application level to homogeneous subspaces in the motherspace. We appologise for not chasing down references to all of these techniques, however most are summarized or derived from [2].

The projective transformation (associated with the notion of homogeneous coordinates) can allow the rotation algorithm to map between arbitrary linear subspaces (those not passing through the origin). It is accomplished by adding one additional basis vector. Rotations in the transformed space can affect translations in the application's space. This technique is well known in machine learning and is usually embedded into the standard algorithms. We maintain it outside of our core algorithm for clarity.

Next, the conformal transformation can allow for transformations between oriented arbitrary generalized circular spaces (lines and circles, planes and spheres, hyperplanes and hyperspheres, etc.). It is accomplished by adding two additional basis vectors. The power of this tranformation comes from the fact that the operation to move things to and from the application space is quadratic.

Continuing, the conic transform can allow for transformations between the generalised conic surfaces centered at the origin (parabolas, ellipsoids, hyper-hyperbolae). It is not clear what a rotation on a space like this really means, but we will see this need not stop us from using this technique.

The final transformation we mention is the balanced transformation. This transformation is the most aggressive, doubling the dimensionality of the application space. Every linear function action on the original space can be represented by a by a transformation by a member of a subgroup of the spin group for the geometric algebra of the expanded space. This is promising for representing general linear functions in our problem setting, however it is not clear the limitations only using rotations are in under this scheme.

7 Conclusion

We have defined a new problem setting, a general approach with a solution template, and dug into an example problem and produced a very simple algorithm to solve it. The work in this area is far from over, however. We have evidence that the algorithm can always make progress in learning a rotation, but do not have a clear answer for how well it performs. Continued

practice with geometric calculus should expand our ability to prove things about this new algorithm. The idea geometric relative introduced early on needs to be developed more and its connection to known matrix algorithms explored. Even the case presented above motivated by the Euclidean distance might provide some insight into standard algorithms. In this same space of problems, the projection operation seems promising for developing a clean, simple, matrix free online PCA algorithm. Finally, a more general direction, the effects of composing several of the standard transformations discussed in the last section should be explored. Knowledge in this area can help us understand generalizations of many existing algorithms that work only on strictly linear subspaces.

References

[1] David Hestenes and Garret Sobczyk. *Clifford Algebra to Geometic Calculus: A Unified Language for Mathematics and Physics*. D. Reidel Publishing Company, 1984.

[2] Anthony Lasenby Chris Doran. *Geometic Algebra for Physicists*. Cambridge University Press, 2005.