Appendix A Optimization

In Chap. 3, the representative model was constructed by simply averaging the models. The average is optimal if the models being averaged are sufficiently close in the space of models and the space is approximately Euclidean. However, because the space is very high dimensional there are no guarantees, especially due to noise, high intra-class variation, and discontinuities. Therefore, it may be useful to define an optimization problem.

Optimization is defined as "the science of determining the 'best' solutions to certain mathematically defined problems" [2]. In this chapter, the problem of constructing a representative model is posed as an optimization problem. Here, two approaches are suggested:

1. optimizing the squared error
2. optimizing the geodesic distance

Note that an optimal model, according to the objective function, does not necessarily mean that recognition performance will improves. The objective function must accurately represent the problem. In many applications, the objective function being optimized is not the "real" problem, however, the "real" problem may be ill-posed and much too difficult to solve. Therefore, in some cases, a simpler objective function is constructed so that the solution may be well defined.

A.1 Squared Error

A common optimization problem involves minimizing the squared error, by some defined error term. This is widely used in neural network applications [3]. The error in this application is considered the Euclidean distance between models. Directly, the optimization problem is stated as follows:

$$\underset{\varphi}{\text{minimize}} \sum_{i=1}^{n} \|\varphi - \varphi_i\|^2$$

where φ and φ_i are models.

This, of course, yields

$$\varphi = \frac{1}{n} \sum_{i=1} n\varphi_i$$

which is the average model.

However, if we redefine the model as a function of the input sketch. This indirect optimization problem becomes:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \sum_{i=1}^{n} \|\varphi(\boldsymbol{\alpha}) - \varphi(\boldsymbol{\alpha}_i)\|^2$$

It can be shown that the solution to this problem has the form

$$\sum_{i=1}^{n} (\varphi(\boldsymbol{\alpha}) - \varphi(\boldsymbol{\alpha}_i)) \frac{\partial \varphi}{\partial \boldsymbol{\alpha}} = 0.$$

Note the differences in these two optimization problems. The first seek to find the model φ that minimizes the squared error (in this case Euclidean distance) between each model, and as shown, the answer is the average model. The second problem seeks to find the sketch $\boldsymbol{\alpha}$ that minimizes the squared error between each model. Although, they are similar problems (and should result in a similar solution), the second problem relates the optimization back to the input sketch.

A.2 Geodesic

Similarly, if one considers to account for the nonlinearity of the "space of models" a geodesic distance may be defined, and then this distance may used in the optimization problem. Before we constructing this type of optimization problem, let us first define the term geodesic.

A parameterized curve γ is called a *geodesic* if the tangent vector field is parallel along γ [1]. Equivalently, if $D_\gamma \gamma' = 0$ (i.e. the acceleration vector is normal to the surface), where $D_\gamma \gamma'$ represents the covariant derivative of the vector field γ' with respect to γ.

Now, consider the following optimization problem:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \sum_{i=1}^{n} \delta\left(\varphi(\boldsymbol{\alpha}), \varphi(\boldsymbol{\alpha}_i)\right)^2$$

where $\delta(\,\cdot\,)$ represents the geodesic distance between two models.

The solution takes the following form:

$$\sum_{i=1}^{n} \frac{\partial \delta}{\partial \varphi} \frac{\partial \varphi}{\partial \boldsymbol{\alpha}} = 0.$$

Note that derivatives may have to be approximated in order to obtain a solution.

References

1. R. L. Bishop and S. I. Goldberg. *Tensor analysis on manifolds*. Dover Publications, 1980.
2. R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 1987.
3. S. Haykin. *Neural networks and learning machines*, chapter 4. Prentice Hall, 3rd edition, 2008.

Appendix B Subspace Approximations

There are many methods, including PCA, RPCA, and KPCA, that attempt to reduced the dimensionality of models or descriptors. The advantage reducing the number of dimensions is to alleviate the effects from the "curse of dimensionality" [1]. This also produces a subspace approximation in which many times the models become more representative of their class.

In this chapter, both PCA and KPCA are discussed specifically in the context of the sketch-based password system introduced in this dissertation.

B.1 PCA

Given a set of observed feature vectors, $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_n\}$, such that

$$\sum_{i=1}^{n} \mathbf{v}_i = 0,$$

PCA diagonalizes the covariance matrix:

$$C = \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^T. \tag{B.1}$$

This decomposition is performed by solving the following eigenvalue problem

$$\lambda \mathbf{e} = C\mathbf{e}. \tag{B.2}$$

Let the set of eigenvectors $\left\{\mathbf{e}_1^*,\ \mathbf{e}_2^*,\ \ldots,\ \mathbf{e}_d^*\right\}$, which span the d-dimensional space, be the solution to the eigenvalue problem corresponding non-negative eigenvalues $\left\{\lambda_1^*,\ \lambda_2^*,\ \ldots,\ \lambda_d^*\right\}$ in descending order. Then, the principle components are consider to be the eigenvectors corresponding to the k largest eigenvalues, where $k < d$. These k principle components may be used to form a linear subspace, which is useful for reducing the dimensionality of the features $\mathbf{v}_i$ for $i = 1 \ldots n$.

B. S. Riggan et al., *Fundamentals of Sketch-Based Passwords,*
SpringerBriefs in Computer Science, DOI 10.1007/978-3-319-13629-5

Assuming the features $\mathbf{v}_i$ lie in d-dimensional space, each point may be projected into the linear subspace formed by the principle eigenvectors. That is,

$$\widetilde{\mathbf{v}}_i = \begin{bmatrix} \text{——} & \mathbf{e}_1^* & \text{——} \\ \text{——} & \mathbf{e}_2^* & \text{——} \\ & \vdots & \\ \text{——} & \mathbf{e}_k^* & \text{——} \end{bmatrix} \mathbf{v}_i$$

for $i = 1...n$.

In regard to this brief, a model may be constructed from these projected features (with reduced number of dimensions). Therefore, the model equation becomes the following:

$$m(\widetilde{\mathbf{v}}) = \frac{1}{z} \sum_{i=1}^{n} \exp\left(-\frac{1}{2} (\widetilde{\mathbf{v}} - \widetilde{\mathbf{v}}_i)^T \boldsymbol{\Sigma}^{-1} (\widetilde{\mathbf{v}} - \widetilde{\mathbf{v}}_i) \right). \tag{B.3}$$

Since dimensionality of the feature vectors effect the size of the model, reducing the dimensionality of the feature vectors makes computing the model more efficient. The danger in reducing the dimensionality too much is that the models from two distinct classes (i.e. different passwords) may be confused with one another.

B.2 KPCA

In the previous section, a subspace approximation of the model was constructed using PCA. Here, the subspace is constructed using nonlinear principle components, but the approach is very similar.

Consider a nonlinear mapping, $\Phi : \mathbb{R}^d \to \mathcal{G}$, where $\mathcal{G}$ denotes the feature space of arbitrary size. Similar to before, the observed features are assumed to be centered (this time in the feature space), meaning

$$\sum_{i=1}^{n} \Phi(\mathbf{v}_i) = 0.$$

Now, consider performing PCA (as discussed previously) in the feature space $\mathcal{G}$. Therefore, the covariance matrix

$$C' = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{v}_i) \Phi(\mathbf{v}_i)^T$$

is diagonalized by solving the following eigenvalue problem:

$$\lambda \mathbf{E} = C' \mathbf{E} \tag{B.4}$$

This time the eigenvectors, $\left\{\mathbf{E}_1^*,\ \mathbf{E}_2^*,\ \ldots,\ \mathbf{E}_d^*\right\}$, lie in $\mathcal{G}$ instead of $\mathbb{R}^d$. Therefore, when the eigenvectors are projected back into the original space, they are nonlinear. Using the k principal eigenvectors, a similar approach may be used to project the feature vectors into the nonlinear subspace constructed using KPCA and then construct a model.

There are, however, some considerations when using KPCA. One consideration is the fact that the mapping, Φ, is unknown. Therefore, the eigenvalue problem is solved using kernel methods; hence, "kernel" PCA. Another consideration is the centering the data in the input space does not guarantee that the projected data is also centered. Therefore, a slight modification must be made (see [2]).

References

1. R. Bellman. *Dynamic Programming*. Princton University Press, 1957.
2. B. Schölkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.