|  |  |
| --- | --- |
| **SVMs:** Maximum margin classifier, decision boundary: . Idea is to create a margin on either side of the decision boundary to better separate the classes. **Support Vectors** are the points directly on the margin and inside the boundaries but correctly classified and points that are incorrectly classified. Linear SVMs as empirical risk minimization. Use Lagrange multipliers to reframe to called the LOSS = Hinge Loss = .  **Strength + Weaknesses of Linear SVMs:** (1) Does not depend on Gaussianity or any other probability distribution. (2) SVMs tend to more robust then logistic regression such as to outliers as the outlliers are only affected by a linear amount whereas in CE the loss would shoot way up and the outliers would have exponential loss. (3) , SVMs are still computable and have a unique and global solution (b/c regularization).  **Non-Linear (Kernel) SVMs:** Recall: kernels and RKHS to take infinite dimensional non-linear problems and turn them into problems depending on dimensions (result of Mercer’s Theorem).  **. Kernel Trick**: Derive optimization problem as a function of inner products in , replace all linear inner products with with .  **SVM Dual Problem** (equivalent to other forms listed before): subject to , , . Param , Hyperparameter . The kernel SVM is just taking and replacing with , everytime we see an inner product replace it with the kernel of the inner product.  **Interpret**: are sparse and are support vectors. Linear: . Kernel: Predictions , predictions are only a function of the support vectors.  **Kernel:** RBF kernels are more flexible then polynomial kernels.  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  **Unsupervised Learning:** Only have data without labels/outcomes. Goal: find some structure in the data that is likely to generalize to future data. PATTERNS via dimension reduction techniques, GROUPS via clustering techniques, ASSOCIATION via graphs/networks, ANAMOLIES, …  **Dimension Reduction**: Have features and want to reduce these/ engineer new features that retain major patterns in the data. **Q.** Why use this? **A**. (1) To visualize the data o.w. w/o dimension reduction we have scatter plots? (2) Want maybe because there is lots of redundancy or irrelevant features (denoising). (3) Pattern discovery. Note that this is useful for both supervised AND unsupervised learning; However, pattern discovery leans towards unsupervised learning.  **Principal Component Analysis (PCA)**: Note that PCA is the optimal linear reduction technique!! where gives principal component scores (e.g. embedding of chapters) and gives the direction,Idea is to retain high variance patterns via linear projections – linear combinations of features that maximize variance. Orthogonal projection of points onto hyperplane such that the variance is maximized. These high variance patterns retain the most information in lower dimensions! **Q**. Is the PC line the same as LS line? **A**. No, the OLS takes the vertical projection, not orthogonal projection. Set up PCA mathematically.  **Optimize**: Line/hyperplane , (weights on each feature) and if is orthogonal, , then is a linear projection and becomes the **first PC** line/plane. Now we want to find the weights that maximizes the variance of this plane. subject to . Note: where is the variance-covariance matrix; we don’t actually know the population so instead we will use an estimate of (assuming ) then (MLE of ). Therefore, the optimization problem is not subject to (to find the first PC) and subsequent PC must be orthogonal. **All PC**: subject to and and . These constraints are to ensure orthogonality.  **Solution**: Eigen decomposition. Recall, . PC solution is given by Eigen Decomp of . Alternatively given by Singular Value Decomp of = .  **SVD**: subject to , , , . Review SVD, , are orthonormal UNSCALED PC, are orthonormal PC DIRECTIONS, diagonal matrix s.t. PROP OF THE VARIANCE OF EACH PC. How to interpret SVD solution is the first major observation pattern and is the first major feature pattern. subsequent patterns are uncorrelated and note that is the first singular value and the variance explained by the first pattern is .  **PVE** (Proportion of Variance Explained): .  **CVE** (Cumulative VE - screeplot): by the first PCs.  **Properties of PCA**: (1) The PC problem is **NON-CONVEX**, yet there exists a global unique closed form solution solved by SVD!!!!!!!!!!!!!!!!!!!!!!!!!!!! (2) are unique up to a sign change. (3) The solution is ordered and nested (important for interpretation), i.e., solution is the same regardless of number of PC selected, dimensions.  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  **:** PCA is the best linear dimension reduction technique, subject to yields the SVD solution! PCA is very good for interpretation. : To center and scale or not? Application dependent! 2-models: **(Application 1)** - Cov model max . This model implies we should center the features! If we scale the features, then we maximize variance patterns in the correlation matrix instead of variance-covariance matrix. If the scales are completely different such as income and proportion of race or something, then scaling could make sense! **(Application 2)** – Low-Rank Mean model. where . We don’t want to center or scale here, our goal is to find the mean! **Conclusion**: If we don’t assume multivariate Gaussianity then we might not want to center or scale our data. It is okay to try with and without centering and choose which has better patterns but don’t scale unless the scales of features differ significantly.  **PCA Extensions**: Regularized PCA functional PCA, PCA for smooth factors, spatial + timeseries. Sparse PCA penalty. Supervised PCA (such as LDA), etc.  **Other Linear Dimensional Reduction Techniques**: (Although PCA is optimal linear DR, sometimes we may not only want to find the highest variance patterns). We may want to interpret different types of patterns beyond PCA.  **Non-Negative Matrix Factorization (NMF):** subject to and and so this is exactly PCA except instead of orthogonality constraints we are adding non-negativity constraints! Why do we want to do this? NMF finds patterns that go in one direction, i.e., PCA finds orthogonal directions that maximize variance, but components can mix positive and negative features, making interpretation harder. NMF finds additive, non-negative directions (parts-based patterns), making it easier to interpret components as intuitive groups or traits. These are called **Archetypal patterns**; not optimal mathematically but very interpretable. **Theory**: NMF yields local solution (you can get different solutions from different starting points) and is not nested/ordered – meaning choosing is super important (change k yields diff solns). This basically yields a set of patterns that seem significant but cannot determine which is most important/significant.  **Non-Linear Dimension Reduction (Manifold Learning)**: Why do we want nonlinear DR? The most interesting patterns don’t necessarily have to lie on a hyperplane! Project our data onto a manifold – some geometric object in -dimensional space. Then we can visualize the data ( observations) as projected onto the manifold by unfolding this manifold into a hyperplane! Manifold learning seeks a lower dimensional representation , , that represents the patterns/distances amongst observations.  **Limitations:** In Manifold learning, (1) we only get observation patterns and we cannot get/interpret feature patterns unlike Linear DR! (2) We never learn the manifold.  **Manifold Map**: This is the visualization of the manifold as a hyperplane. These are very good for interpretation. They are invariant to rotation or translations meaning the position does not matter because we can unfold or flip the manifold before viz as hyperplane – we only care about the relative distances between observations in manifold map.  NOTE: There are two types of manifold learning (1) Global globally represent distances or dissimilarity in lower-dimensional space; can interpret distances between two points as true distance, i.e., if two points are far we can interpret them as far! Local only locally represent distances or dissimilarities in lower-dimensional space; tries to preserve nearest neighbors so within the neighborhood, this cluster of points are close but the distance between clusters cannot be interpreted as either close or far!  **Kernel PCA (GLOBAL):** : distance or dissimilarities, (rbf, poly, etc). Kernel PCA is the eigen decomposition of (global soln) non-linearize distances and apply PCA.  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  **Spectral Methods:**  Spectral methods use the eigenvectors of similarity or distance matrices to embed data in lower dimensions, capturing global structure (e.g., in PCA, spectral clustering, or Laplacian eigenmaps). **Steps**: (1) Compute distance or dissimilarity matrix between observations. (2) Do PCA on and the first columns of are the . Distances Classical MDS (Multi-Dimensional Scaling) , e.g., are centered distances. How is Classical MDS similar to PCA?  **Theorem:** Classical MDS is exactly equivalent to PCA for Euclidean distances. Note, you can also use other distances beyond Euclidean distances such as , Hamming, …  **Spectral methods:** PCA, Kernel PCA, Classical MDS, Spectral Embedding, Laplacian Eigenmaps, Spectral Clustering. **Not spectral methods:** t-SNE, UMAP, NMF  **Spectral Embedding**: as a graph-based dissimilarity matrix (turn our observations into an graph, if two observations are close draw edge, threshold = epsilon graph, nearest neighbor graphs = connect k closest neighbors). is the adjacency matrix (similarity, so almost want inverse of this) and = edge between observations and . Use Normalized Laplacian (like inverse of – acts like dissimilarity for graphs) where is diagonal matrix of degrees for each node.  **Global (Non-Spectral) Manifold Methods:** Metric MDS, seeks to optimize a loss function that keeps distances in X (original data) and Z (dimension reduced data) close! Therefore, (Euclidean distance in lower dimensional space & clearly tries to preserve global distances). CONVEX global solution but we do NOT get a nested and ordered solution.  **Local (Neighbor Embedding) Methods:** Find a lower dimensional space () such that close neighbors remain close, i.e., not all distance are preserved just within cluster distances are preserved. Only interpret locally!  **t-Stochastic Neighbor Embedding (t-SNE)** , **Uniform Manifold Approximation Projection (UMAP)**  **Steps**: (1) Calculate the original space embedding probabilities (normalized dissimilarities), e.g. for t-SNE and UMAP uses generalized distances and calculates distances only over nearest neighbors. (2) Define lower dimensional space embedding probabilities where t-SNE Cauchy and UMAP generalized Cauchy. Far away points get pushed further away!! (3) Learn by fitting some loss function between original and lower dimensional probabilities. t-SNE uses KL-divergence and UMAP uses Cross Entropy Loss .  **Pro/Con:** Very useful for visualizing local patterns and finding clusters. Although they yield a local solution and are very sensitive to hyperparameter selection.  **Application:** Interpret UMAP with caution and JOINTLY USE THIS METHOD WITH A GLOBAL METHOD (like PCA)!!!!!  **Clustering**: Want to find groups of observations (or features or both) that are similar.  **Flat vs Nested Clusters:** A nested cluster is that smaller groups are preserved within the larger group, i.e., if we changed the hyperparameter for number of clusters it would just merge groups to create bigger clusters. Flat is just non-nested; note that flat clustering is very dependent on the hyperparameter (the number of clusters)!  **Notation**: is the number of clusters (fixed) and : where gives the cluster assignment for obs and is the distance btw observation and . **Goal/Optimization**: Min the within cluster dist . Group similar obs. This is NON-CONVEX (np-hard and computationally infeasible for medium sized data). Therefore, we need to find local approximations for clustering!  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  **Kmeans Clustering:**  Gives hard clustering labels; assumes K. Goal is to approximately solve in a fast computational manner. Use Euclidean distance (Kmeans only uses Euclidean distance). Simplify this optimization by using the mean of cluster into (still NP-hard). Now we apply the **Kmeans trick**! Auxiliary variable, mean of cluster (EQN \*)  **Kmeans Algorithm** (iterative algo): Pseudo Code: some initialization of . While not converged: (a) minimize (EQN \*) over given and (b) minimize (EQN \*) over given , end if converged.  **Mathematically:** **(b)** (sample mean of the kth cluster). **(a)** is harder then (b), but still simple, it is simply applying Nearest Centroid Classifier (assign points that optimize the assignment - minimize Euclidean distance - to their nearest mean ). Geometric solution, e.g., for 3 classes you draw a triangle with vertices at the means and edges connecting the vertices and take a decision boundary orthogonal to the midpoint of each edge.  **Summary**: Kmeans algorithm iterates between taking a sample mean and Nearest Centroid Classifier very simple and fast; good for big data.  **Properties:** (1) Attain local solution (convergence guaranteed when means stabilize), can’t certify a global solution. (2) Linear cluster boundary! (3) Very dependent on initialization, therefore we can use Kmeans++ initialization spreads out initial centroid. Worst initializations happen when all the centroids are close together and the clusters don’t separate out well. (4) Kmeans will perform best when NCC is optimal or equivalently Naïve Bayes Classifier under certain conditions, i.e., Kmeans performs well under Gaussian + spherical covariance (Kmeans finds clusters that are spheres/uncorrelated features) and tends to find balanced clusters (of equal sizes). Dependent on (not nested); changing will often dramatically change the solution.  **Pro/Cons:** Advantages are quick and intuitive. Cons are the restrictive assumptions and sensitivity to ; choosing via validation is super important!  **Curse**: When there exists the curse of dimensionality makes calculating distances very difficult therefore Kmeans will perform subpar if directly applied – we must first apply a dimension reduction such as PCA (to spread out the data into clusters and sphere the data by decorrelating)! Therefore, in high dimensional settings or with lots of correlated features PCA and Kmeans combination is very strong for clustering; assumption check: viz in 2d - PC1, PC2.  **Mixture Models**: Soft cluster labels using a generative clustering distribution (assumes underlying distribution of the data given cluster ).  **Gaussian Mixture Models:** This is a soft cluster extension of Kmeans! Assume the data comes from a mixture of different distributions. Assume is the cluster label for the observation, mixture probability (). GMM assumes that (assume here that we have a spherical covariance). Estimation via MLE!! where . Hence, **NOT-CONVEX** – multiplied by two parameters that we are trying to optimize. Trick – it is easy to compute the MLE if is known, just standard computation. So let’s assume that is a latent or hidden variable auxiliary variable  **Expectation-Maximization (EM) Algorithm** (Iterative algo): (1) E-step infers the hidden parameters given the parameters, (2) M-step estimate the parameters given the inferred hidden variables .  **E-step**: , hence we get which is called the responsibilities (same for all mixture models not just GMM) that cluster takes for observation which is just the soft cluster assignment for observation .  **M-step:** Is simply the MLE of the Gaussians: , (weighted sample mean), (weighted sample variance).  ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- | **insert:**  insert |