Overview of PCA

x, $\begin{bmatrix} \\ \\ \\ \end{bmatrix}_d \longmapsto z \begin{bmatrix} \\ \\ \\ \end{bmatrix}_k$, x "de-meaned"

$$\begin{cases} z_1 = \underbrace{\quad}_{1} \\ \vdots = \\ z_7 = 2.5x_1 - 2.9x_2 + 3.9x_3 + \dots 1.7x_d \\ \vdots \\ z_k = \end{cases} \quad , \quad k \ll d$$

$$z = Ax$$

$$\begin{bmatrix} z \end{bmatrix}_k = \begin{bmatrix} \overbrace{\quad A \quad}^{d} \end{bmatrix} \begin{bmatrix} x \end{bmatrix}_d$$

what about $z \longmapsto \hat{x}$? Approximately? Need $\underline{u}$, a basis.

$$\hat{x}^i = \bar{x} + \sum_{j=1}^{k} z_j^i \underline{u}_j$$

dot product

$$\hat{x}^i = \bar{x} + \sum_{j=1}^{k} z_j^i \underline{u}_j = \bar{x} + \sum_{j=1}^{k} \left[(x^i - \bar{x}) \cdot \underline{u}_j \right] \underline{u}_j$$

IF we know z, want $\hat{x}$.

So how do we get z's, u's from X?

Exact:

$$\underline{x}^i = \bar{\underline{x}} + \sum_{j=1}^{d} z_j^i \underline{u}_j$$

Approx:

$$\hat{\underline{x}}^i = \bar{\underline{x}} + \sum_{j=1}^{k} z_j^i \underline{u}_j \quad , \quad k \ll d$$

Error Analysis:

$$\text{Error}_k = \|X - \hat{X}\|_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{d} (x_j^i - \hat{x}_j^i)^2$$

$\|A\|_F$: Frobenius norm:

$$\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}$$

[bases $1 \ldots k$ are correctly reconstructed]

$$= \sum_{i=1}^{N} \sum_{j=1}^{d} (z_j^i \underline{u}_j)^2$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{d} \left[\underline{u}_j^T (\underline{x}^i - \bar{\underline{x}})\right]^2 \underbrace{\|\underline{u}_j\|_2^2}_{=1}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{d} \underline{u}_j^T (\underline{x}^i - \bar{\underline{x}})(\underline{x}^i - \bar{\underline{x}})^T \underline{u}_j$$

$$= \sum_{j=k+1}^{d} \underline{u}_j \underbrace{\left[\sum_{i=1}^{N} (\underline{x}^i - \bar{\underline{x}})(\underline{x}^i - \bar{\underline{x}})^T\right]}_{N\Sigma} \underline{u}_j$$

$$\text{Error}_k = N \sum_{j=k+1}^{d} \underline{u}_j^T \Sigma \underline{u}_j$$

Now, write Σ - symmetric, as eigendecomposition: $\Sigma = \sum_{j=1}^{d} \lambda_i \underline{v}_i \underline{v}_i^T$

$$\text{Error}_k = \sum_{j=k+1}^{d} \underline{u}_j^T \left[\sum_{i=1}^{d} \lambda_i \underline{v}_i \underline{v}_i^T\right] \underline{u}_j$$

$$= \sum_{j=k+1}^{d} \sum_{i=1}^{d} \lambda_i \underline{u}_j^T \underline{v}_i \underline{v}_i^T \underline{u}_j$$

← Pick $\underline{u}_j$'s to only pick up the smallest eigenvalues! $\Rightarrow \underline{u}_j = \underline{v}_j$ for $j = k+1 \ldots d$

$$= \sum_{j=k+1}^{d} \lambda_j$$

← minimized when these are the smallest eigenvalues.

[Formal proof examines Lagrangian w/ constants encoded]

Review: Approximating x^i by z^i, using PCA, minimizes the Frobenius norm of linear reconstruction error.

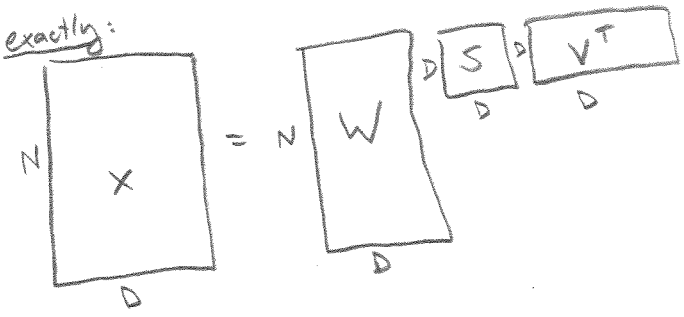
Note: The problem of $\min \|X - \hat{X}\|_F$, where $\hat{X} = WH$, $w_{ij} \geq 0$, $h_{ij} \geq 0$ $\forall_{ij}$ is Non-negative Matrix Factorization (NMF).

Essentially PCA with non-negativity constraints.

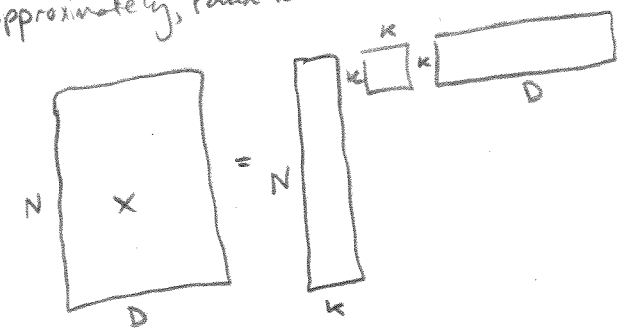
SVD for PCA:

$X = WSV^T$

exactly:



Approximately, rank k:



$$x_j^i = \sum_{\ell=1}^{k} \lambda_\ell w_{i\ell} (V^T)_{\ell j}$$

$$\underline{x}^i = \sum_{\ell=1}^{k} \lambda_\ell w_{i\ell} \underbrace{(\underline{V}^T)_\ell}_{\text{bases}} = \underbrace{\lambda_1 w_1^i}_{\text{coords}} (\underline{V}^T)_1 + \ldots + \lambda_k w_k^i (\underline{V}^T)_k$$

Why is SVD better than eigendecomposition of X^TX?

- X sparse can give X^TX dense. Sparse solvers much faster!
- Forming X^TX can cause loss of numerical precision.
- Really fast SVD solvers exist, optimized for returning just top k.

PCA is a dimensionality reduction technique, SVD is a mathematical technique for fast PCA.

Classification with Naive Bayes

Previously: logistic regression: DISCRIMINATIVE

Now: Bayes optimal + Naive Bayes: GENERATIVE.

Suppose you knew $P(Y|X)$, how to classify?

$h: X \to Y \qquad X = \{0,1\}^D, \; Y = \{0,1\}$

$$h_{opt}(X) = \underset{y}{\operatorname{argmax}} \; P(Y|X)$$

But how many parameters are there in $P(Y|X)$? A lot! You'll see...

Now: Bayes Rule:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

$P(X|Y)$ = Prob of data | class

$P(Y)$ = Prob of class Y

$P(X)$ = Prob of data

$$P(Y|X) \propto P(X|Y) \cdot P(Y)$$

Prior $P(Y)$: on K classes: $\{0, 1, \ldots, K\}$

$$\left.\begin{aligned} P_r(Y=1) &= \pi_1 \\ P_r(Y=2) &= \pi_2 \\ &\vdots \\ P_r(Y=k) &= \pi_k \end{aligned}\right\} \quad \sum_{i=1}^{K} \pi_i = 1$$

Likelihood $P(X|Y)$: on $\{0,1\}^D$:

You can view this as:

$$P_r(X=x|Y=y) = P_r(X_1=x_1|Y=y) \cdot P_r(X_2=x_2|Y=y, X_1=x_1) \cdot P_r(X_3=x_3|Y=y, X_1=x_1, X_2=x_2) \cdot \cdots P_r(X_d=x_d|Y=y, X_1=x_1, \ldots, X_{d-1}=x_{d-1})$$

View through many, many parameters:

$$Pr(\underline{X}=\underline{x} \mid Y=y) = \theta_1^{y} \times \theta_2^{y,1} \times \theta_3^{y,1,2} \times \cdots \times \theta_d^{y,1,\ldots,d-1}$$

For $\theta_2^{y,1}$ there are 2^2 parameters: $\theta_2^{y=0,x_1=0}$, $\theta_2^{y=1,x_1=0}$, $\theta_2^{y=0,x_1=1}$, $\theta_2^{y=1,x_1=1}$

In general, $O(2^d)$ parameters. Eek!

Introduce: <u>Conditional Independence Assumption</u>:

$$Pr(X_2=x_2 \mid Y=y, X_1=x_1) = Pr(X_2=x_2 \mid Y=y), \quad X_1 \text{ doesn't matter.}$$

Thus:

$$Pr(X=x \mid Y=y) = \prod_{i=1}^{d} Pr(X_i=x_i \mid Y=y), \quad \text{much simpler than above!}$$

<u>Naive Bayes Classifier</u>:

$$\hat{y} = h(x) = \arg\max_y Pr(Y=y) \prod_{i=1}^{d} Pr(X_i=x_i \mid Y=y)$$

MLEs for Naive Bayes:

$\Pr(Y=c) = \pi_c$: Categorical($\underline{\pi}$)

$\Pr(X_j = x_j \mid Y=c) = \theta_{jc}$: Categorical($\underline{\theta}_c$)

$$\Pr(X, Y \mid \underline{\pi}, \theta) = \prod_{i=1}^{N} \Pr(Y_i \mid \underline{\pi}) \times \prod_{j=1}^{d} \Pr(x_j^i \mid \theta_j)$$

$$= \prod_{i=1}^{N} \left[\prod_{c=1}^{K} \pi_c^{\mathbb{1}[Y_i=c]} \prod_{j=1}^{d} \Pr(x_j^i \mid \theta_{jc})^{\mathbb{1}[Y_i=c]} \right]$$

$$\log \Pr(X, Y \mid \underline{\pi}, \theta) = \sum_{c=1}^{k} N_c \log \pi_c + \sum_{j=1}^{d} \sum_{c=1}^{k} N_{jc} \log \theta_{jc}$$

where: N_c = count of datapoints labelled c

N_{jc} = count of datapoints w/ j labelled c.

N = count of datapoints

Constrained optimization: $\ell(\underline{\pi}, \theta, \lambda, \underline{\nu}) \log \Pr(X, Y \mid \underline{\pi}, \theta) + \lambda\left(1 - \sum_{i=1}^{k} \pi_i\right) + \sum_{c=1}^{k} \nu_c \left(1 - \sum_{j=1}^{d} \theta_{jc}\right)$

Here $\lambda, \nu_1, \ldots, \nu_c$ are Lagrangian multipliers.

$$\frac{\partial \ell}{\partial \pi_c} = 0 \Rightarrow \frac{N_c}{\pi_c} - \lambda = 0 \Rightarrow \pi_c = \frac{N_c}{\lambda}.$$

since $\sum_{i=1}^{K} \pi_i = 1$, $\sum_{c=1}^{C} \frac{N_c}{\lambda} = 1 \Rightarrow$

$$\lambda = \sum_{c=1}^{C} N_c = N$$

This, $\boxed{\hat{\pi}_c^{MLE} = \frac{N_c}{N}}$

Similarly:

$$\frac{\partial \ell}{\partial \theta_{jc}} = 0 \Rightarrow \frac{N_{jc}}{\theta_{jc}} - \nu_c = 0 \; ; \; \nu_c = N_c \text{ as above} \Rightarrow \hat{\theta}_{jc}^{MLE} = \frac{N_{jc}}{N_c}$$

Smoothing

$$\pi_c = \frac{N_c + \alpha_c}{N + \sum_{c=1}^{k} \alpha_c}$$

Add base counts to prevent ignorant classification.

Equiv to Dirichlet Prior on Categorical Distribution Dir(α).