DME

final exam notes

intro lecture

- there's, like, a lot of data
- data mining ≈ data analysis ≈ data science
- data analysis as statistical inference

data generating process $\xrightarrow{\text{probability}}$ observed data → given a data generating process, what are the properties of the outcomes?

data generating process $\xleftarrow{\text{inference and data mining}}$ observed data → given the outcomes (data), what can we say about the process that generated them?

- data analysis process
  - get (raw) data (lecture 5)
    - understand sampling processes
    - any biases?
    - feedback loops between data analysis and collection?
  - exploratory data analysis (lectures 1-3)
    - become familiar with data
    - spot unexpected properties
    - anomalies, outliers, missing data?
  - prep data for further analysis
    - merge data sets, reformat
    - select/exclude data
    - provide clear rationale for selection/execution
  - build and fit model (lecture 4)
    - generalization is the goal
    - choice of evaluation metric
    - choice of hyperparameters
  - summarize, visualize results
  - deploy the product / communicate findings → start again } presentations, mini-project

- <u>first steps in exploratory data analysis</u>
  - distributions of single variables $\vec{m} = \underset{d\times n}{X} \frac{1}{n} 1_n = \begin{bmatrix} | & & | \\ x_1 & \ldots & x_n \\ | & & | \end{bmatrix} \frac{1}{n}$
    - numerical summaries

location:
- mean: $m = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\mathbb{E}(x) = \int x p(x)\,dx$
- median: robust; if some outlier exists, only changes median to value of neighboring point
- trimmed mean:
  $\frac{1}{n-2k}\sum_{i=k+1}^{n-k} x_{(i)} \Rightarrow k=0$ normal mean, $k \to \frac{n}{2}$ median

scale:
- variance: $v = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2 = \mathbb{E}(x^2) - \mathbb{E}(x)^2$
- median absolute deviation is more robust: $MAD = \text{median}(|x_i - \text{median}(x_i)|)$
- IQR (also robust): range of middle 50% of data: $IQR = x_{(\lceil 3/4n \rceil)} - x_{(\lceil n/4 \rceil)}$

shape (skew):
- $\text{skew}(x) = \mathbb{E}\left[\left(\frac{x-\mu}{\sigma}\right)^3\right] \Rightarrow$ $=0$ if symmetric around mean, >0 if long tail to the right
- Galton's measure of skewness: $\frac{(Q_3 - Q_2) - (Q_2 - Q_1)}{Q_3 - Q_1}$

- $\text{kurt}(x) = E\left[\left(\frac{x-\mu}{\sigma}\right)^4\right] \Rightarrow = 3$ for Gaussian

- excess kurtosis $(x) = \text{kurt}(x) - 3$

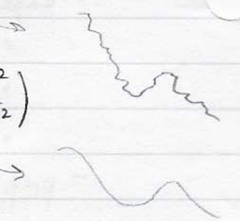
- robust kurtosis $(x) = \dfrac{\overbrace{(q_{7/8} - q_{5/8})}^{\text{spread around } 3/4} + \overbrace{(q_{3/8} - q_{1/8})}^{\text{spread around } 1/4}}{\underbrace{q_{3/4} - q_{1/4}}_{\substack{\text{spread around } 1/2 \\ (\times 2 \text{ range of two individual above})}}}$

- graphs
  - histogram
    - binning: $B_1 = [L, L+h),\ B_2 = [L+h, L+2h), \ldots,\ B_k = [L+(k-1)h, L+kh)$
  - kernel density estimate
    - fixes problems w/ histogram: doesn't depend on starting point, smooth
    - boxcar: counts points $\pm h/2$:

      $$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h}\Pi_h(x - x_i) \quad \text{where } \Pi_h(x) = \begin{cases} 1 & \text{if } x \in [-\frac{h}{2}, \frac{h}{2}] \\ 0 & \text{otherwise} \end{cases}$$

    - Gaussian kernel:

      $$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \quad \text{where } K_h(x) = \frac{1}{\sqrt{2\pi h^2}}\left(-\frac{x^2}{2h^2}\right)$$



  - boxplot: median, Q_1/Q_3, and $Q_1 - 1.5\,IQR$, $Q_3 + 1.5\,IQR$

- joint distributions of two variables
  - numerical summaries
    - $\text{cov}(x,y) = E[(x-\mu_x)(y-\mu_y)] = E[xy] - \mu_x\mu_y$ — also, $C = E[(\underline{x}-\mu)(\underline{x}-\mu)^T]$ — $\text{cov}(ax+b, y) = a\,\text{cov}(x,y)$
    - correlation $\rho(x,y) = \dfrac{\text{cov}(x,y)}{\sqrt{V(x)V(y)}} = E\left[\left(\dfrac{x-\mu_x}{\sigma_x}\right)\left(\dfrac{y-\mu_y}{\sigma_y}\right)\right]$
    - multiplying by constant:

      $V(A\underline{x} + \underline{b}) = ACA^T$ where $C = V(\underline{x})$
    - correlation matrix K: use the linear transformation $D^{-1/2}(x-\mu)$ where D contains the diagonal elements of C, i.e., $D = \text{diag}(V(x_1), \ldots, V(x_d))$

      $K = D^{-1/2} C D^{-1/2}$
      - by construction, all diagonal elements are 1
    - nonlinear relationships:
      - correlation: $\rho(g(x), g(y)) = \dfrac{\text{cov}(g(x), g(y))}{\sqrt{V(g(x))V(g(y))}}$

- simple preprocessing
  - simple outlier detection: see if outside range $[Q_1 - k\,IQR,\ Q_3 + k\,IQR]$ where $k = 1.5$ usually
  - data standardization
    - centering: $X = \tilde{X} H_n$ where X is centered data, $\tilde{X}$ is uncentered data, $\underbrace{H_n}_{\text{"centering matrix"}} = I_n - \frac{1}{n}1_n 1_n^T$

      or: $X = ((\tilde{x}_1 - \underline{m}), \ldots, (\tilde{x}_n - \underline{m}))$

      $= \tilde{X} - (\underline{m}, \ldots, \underline{m})$

      $\underline{a}H_n = \underline{a} - \frac{1}{n}\underbrace{\underline{a}1_n}_{\text{sums values of } a}1_n^T = \underline{a} - \bar{a}1_n^T$

      $H_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} - \frac{1}{n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$

      and $\underline{m} = \tilde{X}\frac{1}{n}1_n$
  - scaling to unit variance: $\hat{C} = \frac{1}{n}\tilde{X}H_n\tilde{X}^T$

DME
final exam notes

principal component analysis

- assume for this lecture that data has been centered
- PCA by sequential variance maximization
  - first PCA direction
    - unit vector $\underline{w}_1$ for which the projected data $\underline{w}_1^T x_i$ is maximally variable

$$\max_{w_1} \ \underline{w}_1^T C \underline{w}_1 \qquad \text{do this because want to max variance } V(\underline{w}_1^T \underline{x})$$

$$\text{s.t. } \|\underline{w}_1\| = 1 \qquad \text{note: for later: } C = \frac{1}{n} XX^T$$

- can be solved in closed from from $C = U \Lambda U^T$

where U is an orthogonal matrix, Λ is a diagonal matrix with eigenvalues $\lambda_i \geq 0$ and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$

let $\underline{w}_1 = U\underline{a}$

then $\underline{w}_1^T C \underline{w}_1 = a^T U^T U \Lambda U^T U a = a^T \Lambda a = \sum_{i=1}^{d} a_i^2 \lambda_i$

and constraint becomes $\|w_1\|^2 = \underline{w}_1^T \underline{w}_1$

$$= a^T U^T U a = a^T a = \sum_{i=1}^{d} a_i^2 \stackrel{?}{=} 1$$

so formulation is now:

$$\max_{a_1, \dots, a_d} \sum_{i=1}^{d} a_i^2 \lambda_i$$

$$\text{s.t. } \sum_{i=1}^{d} a_i^2 = 1$$

to solve, just set $a_1 = 1$ and the remaining a_i to zero

$$\underline{w}_1 = U \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underline{u}_1$$

- first principal component score (not PC direction) is: $z_1 = \underline{w}_1^T \underline{x}$
  $= \underline{u}_1^T \underline{x}$ → also by $s_1^T v_1 \dots$
- variance of z_1 is λ_1
- all PC scores given by $\underline{w}_1^T X = \underline{u}_1^T X$

  - subsequent PC directions:

$$\max_{\underline{w}_2} \ \underline{w}_2^T C \underline{w}_2 \quad \Rightarrow \quad \max_{b_1, \dots, b_d} \sum_{i=1}^{d} b^2 \lambda_i \qquad \text{letting } \underline{w}_2 = U\underline{b}$$

$$\text{s.t. } \|\underline{w}_2\| = 1 \qquad \text{s.t. } \sum_{i=1}^{d} b_i^2 = 1$$

$\underline{w}_2^T \underline{w}_1 = 0$ → $\underline{w}_2^T \underline{x}$ needs to reveal something "new" about the data

$$\boxed{b_1 = 0} \quad \text{since} \quad \underline{w}_2^T \underline{w}_1 = \underline{b}^T U^T \underline{u}_1 = \underline{b}^T \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = b_1 = 0 \text{ since } \underline{w}_2^T \underline{w}_1 = 0$$

$$\Rightarrow \max_{b_2, \dots, b_d} \sum_{i=2}^{d} b^2 \lambda_i$$

$$\text{s.t. } \sum_{i=2}^{d} b_i^2 = 1$$

this is structurally similar to before; we get

$$\underline{w}_2 = U \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underline{u}_2$$

- Subsequent PC directions (cont'd)
  - so in general:

$$\max_{\underline{w}_m} \underline{w}_m^T C \underline{w}_m$$
$$\text{s.t. } \|w_m\| = 1$$
$$\underline{w}_m^T \underline{w}_i = 0 \text{ for } i = 1, \dots, m-1$$

  - total variance explained of the z_m, $m = 1, \dots, k$ is

$$\sum_{m=1}^{k} V(z_m) = \sum_{m=1}^{k} \lambda_m$$

- PCA by simultaneous variance maximization
  - Solves the problem:

$$\max_{w_1, \dots, w_k} \sum_{i=1}^{k} \underline{w}_i^T C \underline{w}_i$$
$$\text{s.t. } \|w_i\| = 1 \quad i = 1, \dots, k$$
$$\underline{w}_i^T w_j = 0 \quad i \neq j$$

It's interesting that the sequential (greedy) approach yields the same answer since greedy approaches usually don't

- PCA by maximization of approximation error
  - have a set of k orthonormal vectors $w_1, \dots, w_k$

$$P = \sum_{i=1}^{k} \underline{w}_i \underline{w}_i^T = W_k W_k^T \qquad \text{where } W_k = (\underline{w}_1, \dots, \underline{w}_k)$$

where P projects any vector onto its subspace

$$\hat{\underline{x}}_i = P \underline{x}_i \quad , \quad \hat{\underline{x}} = P \underline{x} = \overbrace{\sum_{i=1}^{k} \underline{w}_i \underline{w}_i^T}^{P} \underline{x}$$

ask: which subspace yields the smallest expected approximation error?

$$\min_{\underline{w}_1, \dots, \underline{w}_k} \mathbb{E} \| x - \overbrace{\sum^{k} \underline{w}_i \underline{w}_i^T}^{P} \underline{x} \|^2$$
$$\text{s.t. } \|\underline{w}_i\| = 1 \quad i = 1, \dots, k$$
$$\underline{w}_i^T \underline{w}_j = 0 \quad i \neq j$$

after some algebra:

$$\mathbb{E} \| \underline{x} - W_k W_k^T x \|^2 = \overbrace{\mathbb{E}(\underline{x}^T \underline{x})}^{\text{constant}} - \sum_{i=1}^{k} \underline{w}_i^T C \underline{w}_i$$

→ so minimizing approx error is equivalent to $\max \sum_{i=1}^{k} \underline{w}_i C \underline{w}_i$

(one minus)

- computing fraction of variance explained:

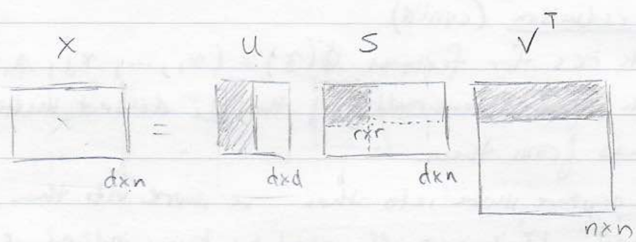
$$\frac{\mathbb{E} \| \underline{x} - U_k U_k^T \underline{x} \|^2}{\mathbb{E}(\underline{x}^T \underline{x})} = 1 - \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} = 1 - [\text{fraction of variance explained}]$$

DME
final exam notes

<u>principal component analysis</u> (cont'd)

- SVD:

$$X = USV^T$$



so now it's an approximation

$$\hat{X} = \sum_{i=1}^{r} s_i \underline{u}_i \underline{v}_i^T$$

$r \le \min(d, n)$ is rank of X

$$= \sum \underline{u}_i \underbrace{\underline{v}_i s_i}_{\text{PC scores } (\underline{z}_i)}$$

$\underline{u}_i \rightarrow$ PC directions (same $\underline{u}_i$ as before)

$$= \sum_{i=1}^{k} \underline{u}_i \underline{z}_i^T$$

- if the Gram matrix is defined as $G = X^T X$, we can find its eigenvalue decomposition as

$$G = (USV^T)^T (USV) = \dots = V \Sigma V^T \quad \text{where } \Sigma = SS \rightarrow \text{eigenvalues } s_i^2$$

so we have s_i's and V; this is what we need to compute the PC scores

we can therefore compute the PC scores of X from its Gram matrix of centered data, without first computing the PC directions

$$Z = \sqrt{\Sigma_k} V_k^T = S_k V_k^T$$

also, $Z = U_k^T X$ but we don't have U_k if we only have the Gram matrix

<u>dimensionality reduction</u>

- can compute G from uncentered $\tilde{G}$:

$$G = X^T X = H_n^T \tilde{X}^T \tilde{X} H_n = H_n \tilde{G} H_n$$

$H_n \rightarrow$ removes mean value of each column from the column / " " " row " " row

(need to use fact that H_n is symmetric)

So just need to "double center" first

- can compute G from Δ, where Δ has elements $\delta_{ij}^2 = \|\underline{x}_i - \underline{x}_j\| = \|x_i\|^2 + \|x_j\|^2 - 2x_i^T x_j$

since $\|x_i\|^2$ is a constant along row i
and $\|x_j\|^2$ is a constant along column j,

we can double-center and multiply by $-\frac{1}{2}$ (to recover $x_i^T x_j$):

$$G = -\frac{1}{2} H \Delta H$$

dimensionality reduction (cont'd)

- idea: compute PCs for features $\phi(\underline{x}) = (x_1, \dots, x_d, x_1 x_2, \dots, x_1 x_d, \dots, x_d x_d)^T$

kernel PCA

  - the much higher dimensionality of the $\underline{\phi}_i$ doesn't matter as long as we only compute k principal components from them.
  - PCs computed from $\underline{\phi}_i$ must capture more info than PCs captured from $\underline{x}_i$
  - kernel trick: don't actually need to know individual ϕ_i, only the inner products $\underline{\phi}_i^T \underline{\phi}_j = \phi(\underline{x}_i)^T \phi(\underline{x}_j)$

  $\Rightarrow$ then we can just compute the PC scores from the Gram matrix of $\underline{\Phi}$:

$$(\tilde{G})_{ij} = \underline{\phi}_i^T \underline{\phi}_j = \phi(\underline{x}_i)^T \phi(\underline{x}_j) = k(\underline{x}_i, \underline{x}_j)$$

  - example kernels:
    - polynomial: $k(\underline{x}, \underline{x}') = (\underline{x}^T \underline{x}')^a$
    - Gaussian: $k(\underline{x}, \underline{x}') = \exp\left(-\frac{\|\underline{x} - \underline{x}'\|^2}{2\sigma^2}\right)$
  - to compute PCs, proceed exactly as before:

$$G = H_n \tilde{G} H_n \quad \Rightarrow \quad Z = \sqrt{\Sigma_k}\, V_k^T$$

- multidimensional scaling (MDS)
  - umbrella term for several methods that operate on dissimilarities δ_{ij} (one simple example is Euclidian distance)
  - metric MDS: numerical values of the dissimilarities are assumed to carry info

  (contrast this w/ nonmetric MDS, where only the rank-order of the dissimilarities matter)

$$\min_{z_1, \dots, z_n} \sum_{i<j} w_{ij} \big(\underbrace{\|\underline{z}_i - \underline{z}_j\|}_{\text{"stress": essentially a cost function}} - \delta_{ij} \big)^2$$

  i.e., distance between points we create ≈ distance according to distance metric

    - goal is to find n points $z \in \mathbb{R}^k$ that solve this,
    - $\|\underline{z}_i - \underline{z}_j\|$ is the Euclidian distance between $\underline{z}_i$ and $\underline{z}_j$
    - k usually set to 2 for visualization
    - usually solved by gradient descent
    - weights $w_{ij} \geq 0$ are specified by the user
  - nonmetric MDS: only relation between the δ_{ij} are assumed to matter

  optimization modified to:

$$\min_{z_1, \dots, z_n, f} \sum_{i<j} w_{ij} \left(\|\underline{z}_i - \underline{z}_j\| - f(\delta_{ij}) \right)^2$$

  where f is a monotonic function that converts the dissimilarities into distances

    - typically solved by iterating between optimization w.r.t. the $\underline{z}_i$ and w.r.t. f, which can be done by regression
  - classical MDS: assumes dissimilarities δ_{ij} are (squared) Euclidian distances
    - like before,
      1. compute hypothetical Gram matrix G' : $G' = -\frac{1}{2} H_n \Delta H_n$
      2. compute top k eigenvectors $\underline{v}_k \in \mathbb{R}^n$ of G and form matrices Σ_k and V_k
      3. compute PC scores as $Z = \sqrt{\Sigma_k}\, V_k^T$
    - nice in that it produces nested solutions
    - problem: Δ is symmetric but not necessarily PSD
      - solution: choose k small enough that all eigenvalues contained in Σ_k are positive

DME
final exam notes

dimensionality reduction (cont'd)

- isomap
  - uses geodesic distance, which is when dissimilarity is measured by the shortest distance between two points only when allowed to travel on the data manifold from one neighboring datapoint to the next
  - with disconnected parts of the graph, the isomap is often applied to each component separately

neighborhood of a datapoint can be taken to be m-nearest neighbors or all points w/in a Euclidian distance

performance evaluation in predictive modeling

- prediction and training loss
  - $J(h) = \mathbb{E}_{\hat{y},y}[\mathcal{L}(\hat{y},y)] = \mathbb{E}_{\underline{x},y}[\mathcal{L}(h(\underline{x}),y)]$
  - optimize: $\min_h J(h)$
  - three difficulties:
    1. can't compute expectation over $(\underline{x},y)$ analytically
    2. $\mathcal{L}$ may be hard to evaluate
    3. minimizing $\mathcal{L}$ w.r.t. a function is generally difficult
  - training loss
    - approx training loss w/ sample average
      $$J(h) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(\underline{x}_i), y_i)$$
    - samples called D^{train}
    - indicate model family and hyperparams: $h(\underline{x}) \equiv h_{\underline{\lambda}}(\underline{x};\underline{\theta})$ where $\underline{\lambda}$ is a vector of hyperparams indicating model family and some tuning params associated with it; $\underline{\theta}$ is usually weights
    - work with training loss function
      $$J_{\underline{\lambda}}(\underline{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h_{\underline{\lambda}}(\underline{x}_i;\underline{\theta}), y_i)$$

- generalization performance
  - algorithm that turns training data D^{train} into a prediction function $\hat{h}$ by A:
    $$\hat{h} = A(D^{train})$$
    A is more general than $\hat{h}$

    so:
    $$\bar{J}(A) = \mathbb{E}_{D^{train}}[J(\hat{h})] = \mathbb{E}_{D^{train}}[J(A(D^{train}))]$$
  - overfitting: a model has been overfitted to the training data if reducing its complexity reduces the (expected) prediction loss.
  - underfitting: when increasing flexibility $\Rightarrow$ decreased (expected) prediction loss
  - variability of prediction loss increases with the flexibility of the model
  - bias is the difference between training loss and prediction loss as $n \to \infty$

- estimating generalization performance
  - hold-out approach: $\hat{h} = A(D^{train})$
    validation error: $\hat{J}(\hat{h};\tilde{D}) = \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} \mathcal{L}(\hat{h}(\tilde{x}_i), \tilde{y}_i)$ where $\tilde{D}$ is validation set (and tildes denote validation-related variables)

    useful to stratify

performance evaluation in predictive modeling (cont'd)

- estimating generalization performance (cont'd)
  - cross validation:
    - split data into K subsets ("folds") $D_1, \dots, D_K$, perhaps w/ stratification

do for $k=1,\dots,K$
$$\begin{cases} D_k^{train} = \bigcup_{i \neq k} D_i, \quad D_k^{val} = D_k \\ \hat{h}_k = A(D_k^{train}) \\ \hat{J}_k = \hat{J}(\hat{h}_k ; D_k^{val}) \end{cases}$$

$$CV = \frac{1}{K} \sum_{i=1}^{K} \hat{J}_k$$ $\Rightarrow$ estimate of prediction performance $\bar{J}(A)$ as opposed to $\hat{J}$ from before

- can estimate variance of cv-score: $V(CV) \approx \frac{1}{K} V(\hat{J})$

$$V(\hat{J}) \approx \frac{1}{K} \sum_{k=1}^{K} (\hat{J}_k - CV)^2$$

- hyperparameter selection and performance evaluation
  - two main approaches:
    1. hold-out to select hyperparams, hold-out for performance eval
    2. CV " " " " (again) " "
  - Approach for "two-times hold-out"
    1. Split off test data D^{test}
    2. Of remaining, split in D^{train} and D^{val}
    3. for all λ:
       $\hat{h}_\lambda = A_\lambda (D^{train})$
    4. choose best λ as $\hat{\lambda} = \operatorname{argmin}_\lambda PL(\lambda)$ where $PL(\lambda) = \hat{J}(\hat{h}_\lambda ; D^{val})$
    5. re-estimate model w/ best λ on all non-test data:
       $\hat{h} = A_{\hat{\lambda}} (D^{train} \cup D^{val})$
    6. evaluate on test data:
       $\hat{J} = \hat{J}(\hat{h} ; D^{test})$
    7. [optional] re-estimate $\hat{h}$ using all data for final prediction function $\hat{h}$
       $\hat{h}(x) = A_{\hat{\lambda}}(D)$

    note: CV-score is typically an optimistic estimate of prediction loss b/c hyperparams are chosen to minimize it
  - approach for cross-validation and hold-out: similar to above