Finding Principal Components

1. find eigenvalues by solving: $\det(\Sigma - \lambda I) = 0$

$$\det\begin{pmatrix} 2.0-\lambda & 0.8 \\ 0.8 & 0.6-\lambda \end{pmatrix} = (2-\lambda)(0.6-\lambda) - (0.8)(0.8) = \lambda^2 - 2.6\lambda + 0.56 = 0$$

$$\{\lambda_1, \lambda_2\} = \tfrac{1}{2}\left(2.6 \pm \sqrt{2.6^2 - 4 * 0.56}\right) = \{2.36, 0.23\}$$

2. find i^{th} eigenvector by solving: $\Sigma\, \mathbf{e}_i = \lambda_i\, \mathbf{e}_i$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_{1,1} \\ e_{1,2} \end{pmatrix} = 2.36\begin{pmatrix} e_{1,1} \\ e_{1,2} \end{pmatrix} \Rightarrow \begin{matrix} 2.0e_{1,1} + 0.8e_{1,2} = 2.36e_{1,1} \\ 0.8e_{1,1} + 0.6e_{1,2} = 2.36e_{1,2} \end{matrix} \Rightarrow e_{1,1} = 2.2e_{1,2}$$

$$\Downarrow$$

$$e_1 \sim \begin{bmatrix} 2.2 \\ 1 \end{bmatrix}$$

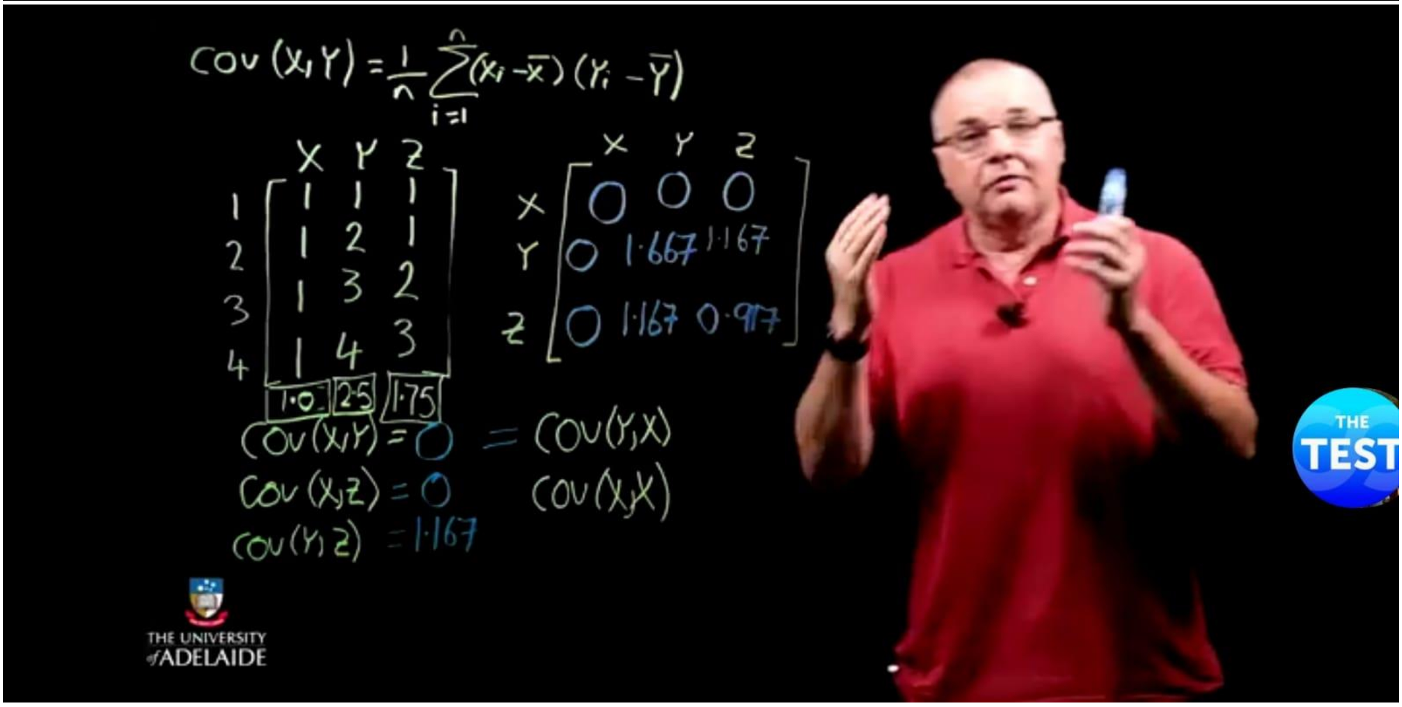
$$\Downarrow \text{ want: } \|e_1\| = 1$$

$$e_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$$

slope: 0.454

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_{2,1} \\ e_{2,2} \end{pmatrix} = 0.23\begin{pmatrix} e_{2,1} \\ e_{2,2} \end{pmatrix} \Rightarrow e_2 = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

PCA 5: finding eigenvalues and eigenvectors



PCA algorithm II (sample covariance matrix)

- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute covariance matrix Σ

$$\boxed{\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T} \quad \text{where} \quad \boxed{\overline{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i}$$

- **PCA** basis vectors = the eigenvectors of Σ

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

slide by Barnabás Póczos and Aarti Singh

Reminder: Eigenvector and Eigenval

$$\mathrm{A}x = \lambda x$$

A: Square matrix
λ: Eigenvector or characteristic vector
x: Eigenvalue or characteristic value

$$\textit{Show } x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \textit{ is an eigenvector for } A = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}$$

$$\textit{Solution}: Ax = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\textit{But for } \lambda = 0, \quad \lambda x = 0\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\textit{Thus, } x \textit{ is an eigenvector of } \mathrm{A}, \textit{and } \lambda = 0 \textit{ is an eigenvalue}.$$

[SVM]

[illegible] w, b

1) Draw decision boundary
2) Write equation for boundary: $-y+x-8=0$
3) Rewrite equation as $\vec{w}\cdot\vec{x}+b=0$
4) Scale equation $[1\ 1]\cdot\begin{bmatrix}x\\y\end{bmatrix}+(-8)=0$

⊖ Support vector [illegible]

$c([1\ 1])\begin{bmatrix}\cdot\\\cdot\end{bmatrix}+c(-8)=-1 \Rightarrow c=1/2$

$w=[1/2\ 1/2]\quad b=-4$

Calculate α ⇒ margin Width $= 2/\|w\|$

1) For all non support vector's $\alpha=0$ otherwise, the point lies on the gutter and $\alpha>0$
2) $\sum_{s.v.}\alpha_i y_i=0 \Leftrightarrow \sum_{\text{positive s.v.}}\alpha_p=\sum_{\text{negative s.v.}}\alpha_n$ s.v. = support vectors
3) $\sum_i \alpha_i y_i \vec{x}_i=\vec{w} \Leftrightarrow \vec{w}=\sum_{\text{positive s.v.}}\alpha_p x_p-\sum_{\text{negative s.v.}}\alpha_n x_n$

$K(u,v)=\varphi(u)^T\cdot\varphi(v)$ ⇒ No change

Polynomial → $K(u,v)=(u\cdot v+b)^n$

$K(u,v)=e^{-\frac{\|u-v\|^2}{2\sigma^2}}$ → Radial Basis Function

Decision Tree

...	Eye	Spec
	3	T
	3	T
	2	T
	2	T
	3	H
	2	T

$H(x)=-\sum_{i=1}^{k}P(x=x_i)\log_2 P(x=x_i)$ → "Entropy"

$IG(x)=H(x)-H(Y|X)$

$H(Y|X)=-\sum_{j=1}^{v}P(x=x_j)\sum_{i=1}^{k}P(Y=y_i|X=x_j)\log_2 P(Y=y_i|X=x_j)$

$H(species)=-\frac{2}{6}\log_2\frac{2}{6}-\frac{4}{6}\log_2\frac{4}{6}$

$H(Spec|Eye)=-\frac{4}{6}\left(\frac{1}{4}\log_2\frac{1}{4}+\frac{3}{4}\log_2\frac{3}{4}\right)-\frac{2}{6}\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)$

Eyes: 3 → H:1, T:3; 2 → H:1, T:1

En, highest IG → Root

Avoid overfit in D.T. ⇒ * Fixed depth or # leaves * Pruning * Early Stopping * use random forest

→ **Low Entropy**: Histogram has many lows and highs

* Values sampled from it are more predictable
* High from a varied (peaks & valleys) distributions

→ Standard D.T have no learning bias, training set error is zero! (no noise)

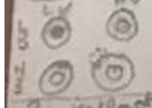
→ D.T. can represent any Boolean func. Linearly separation is %100 accuracy → Lin. classifier

Bias/Variance → Simple learner are good ↳ low variance, don't usually overfit

→ Simple learners are bad ↳ high bias, can't solve hard learning problems

LV HV

LB HB



Bagging (Bootstrap Aggregating)

$Var(Bagging(L(x,D)))=\frac{Var(L(x,D))}{N}$ → # examples; D → set containing N example

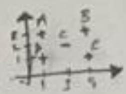
Boosting

1) Initialize weights $w_i=1/N$
2) Calculate error rate for each h, $\epsilon=\sum_{\text{wrong}}w_i$
3) Pick best h with smallest error rate
4) Calculate voting power of h: $\alpha=\frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$
5) Finished?
   * $H(x)=\text{Sign}(\sum\alpha_t h_t(x))$ is good enough?
   * Enough iteration (best h has $\epsilon=1/2$)
   * No good classifiers left

   Yes → H(x); ↓ NO
6) Update weight

$w_{new}\begin{cases}\frac{1}{2}\cdot\frac{1}{1-\epsilon}\cdot w_{old}, & \text{if right}\\ \frac{1}{2}\cdot\frac{1}{\epsilon}\cdot w_{old}, & \text{if wrong}\end{cases}$

		R_1	R_2	R_3
	w_A	1/5	1/8	1/12
	w_B	1/5	1/8	3/12
	w_C	1/5	1/2	4/12
	w_D	1/5	1/8	1/12
	w_E	1/5	1/8	3/12
[illegible]	ε	[illegible]	[illegible]	[illegible]
[illegible]	ε	[illegible]	[illegible]	[illegible]
[illegible]	ε	1/5	1/8	1/12
[illegible]	ε	[illegible]	[illegible]	[illegible]
[illegible]	ε	[illegible]	[illegible]	[illegible]
	h	[illegible]	[illegible]	[illegible]
	ε	[illegible]	[illegible]	[illegible]
	α	[illegible]	[illegible]	[illegible]



Bagging ⇒ is machine learning ensemble meta-algorithm designed to improve stability, reduce variance, accuracy

⇒ Reweight data points. ⇒ Weight is depend on classifier's accuracy

Boosting ⇒ is ML. ensemble meta-algo. for primarily only reduce bias ⇒ Resample data points ⇒ Weight of each classifier is the same

Random Forest ⇒ many is more likely to be accurate than that of one.

→ when we want to classify a new object, we take the vote of each D.T. and combine the outcome to make final decision _ majority vote wins

SVM ⇒ Why large margin? ⇒ max. robustness relative to uncertainty ⇒ symmetry breaking, independent of correctly classified instances

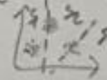
Clustering: Kmeans: 1) Initialize K cluster center randomly drawn without replacement

2) Repeat until converge. (until no change in [illegible])

↳ Assign each datapoint to cluster with the closest center

↳ Assign each cluster center to be the mean of its cluster's data point

$\mu_k \leftarrow |S_k|^{-1}\sum_{n\in S_k}x_n$



★ K-Means always terminate.

★ Good dissimilarity only useful for comparing clusterings

Spectral Clustering: Similarity Graphs

Model [illegible] local neighborhood relations between data points

– epsilon NN

$w_{ij}=\begin{cases}1 & |x_i-x_j|\le\epsilon\\0 & \text{otherwise}\end{cases}$ → Controls size of neighborhood

– Gaussian kernel similarity func.

$w_{ij}=e^{-\frac{|x_i-x_j|^2}{2\sigma^2}}$ → Controls size of neighborhood.

small σ → group only nearby points

min Cut: $cut(A,B)=\sum_{u\in A, v\in B}W(u,v)\quad A\cap B=\emptyset$

k-means vs. Spectral Clustering: Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundary.

Hierarchical Clustering

Bottom-up ⇒ Agglomerative ⇒ Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

⇒ # dendrograms with n leafs $=(2n-3)!/[(2^{(n-2)})(n-2)!]$

Single Link (min) ⇒ Potentially long and skinny clusters

Complete L. (max) → Tight clusters

Average Link: → The most widely used measure → Robust against noise

Advantages	Disadv.
• Simple to implement	• May have imbalanced clusters
• Clusters have adaptive shapes	• Still have to choose # clusters or threshold
• Provides a hierarchy of clusters	• Need to use "ultrametric" to get a meaningful hierarchy

Good Clustering

Internal ⇒ Intra-class similarity high, inter-class is low

⇒ The measured quality of a clustering depends on both the obj. representation and the similarity measure used

External Criteria

→ Assesses a clustering with respect to ground truth in clusters

• Entropy of classes in clusters

• Purity → the ratio between the dominant class in the cluster and the size of cluster.

$purity(\Omega,C)=\frac{1}{N}\sum_k \max_j|w_k\cap C_j|$

$Entropy(CC_i)=\sum_{TC_j\in TC}-P(TC_j|CC_i)\log P(TC_j|CC_i)$; target, computed

Principal Component Analysis

Reminder: Eigenvector and Eigenvalue

Example 1: Find the eigenvalues of
$$A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{vmatrix} = (\lambda - 2)(\lambda + 5) + 12$$

$$= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

two eigenvalues: $-1, -2$

Note: The roots of the characteristic equation can be repeated. That is, $\lambda_1 = \lambda_2 = \ldots = \lambda_k$. If that happens, the eigenvalue is said to be of multiplicity k.

Example 2: Find the eigenvalues of
$$A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ 0 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 2 \end{vmatrix} = (\lambda - 2)^3 = 0$$

$\lambda = 2$ is an eigenvector of multiplicity 3.

PCA algorithm II (sample covariance matrix)

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = A(x - \mu)$$

where y is the new point, x is the old one, and the rows of A are the eigenvectors

SVM

Primal $\to \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left[y_i \left[\langle x_i, w \rangle + b \right] - 1 \right]$

Dual $\to \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$

$\sum_i \alpha_i y_i = 0 \qquad w = \sum_i \alpha_i y_i x_i$

$b = y_k - \sum_i \alpha_i y_i x_i^T x$ for $\alpha_k > 0$

Dual sol. provides the kernel solutions:
- never computes feature mapping, also most of α_i's would be zero.

Why Large-margins?
- max. robustness relative to uncertainity
- symmetry breaking, independent of correctly classified instances

Support vector $\to \alpha_i \geq 0$, non-support $\to \alpha_i = 0$

$y \leftarrow \text{sign}(\vec{w} \cdot \vec{x} + b)$

↓ dual solution

$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \langle x_i, x \rangle + b \right]$

Karush-Kuhn Tucker $\to \alpha_i > 0$

$y_i \left[\langle w, x_i \rangle + b \right] = 1$

optimization problem: $\frac{1}{2}\|w\|^2 + C \sum \xi_i$
- C is a regularization parameter:
  small C allows constraints to be easily ignored → large margin
- large C makes cons. hard to ignore → narrow margin, classifier makes few errors.
- $C = \infty$ enforces all cons.: hard margin
- kernel-trick: in the dual form. of the SVM features only appear as dot products which can be represented by kernels.
- SVM uses hinge-loss, gives zero weight non-support vectors.

$w = \sum_{p \in S} \alpha_p x_p - \sum_{n \in S} \alpha_n x_n$

Partitioning clust. → k-means, spectral clust.

Hierarchical clust. → bottom-up agglomerative

Decision Trees

$IG(X) = H(Y) - H(Y|X)$

$H(Y) = -\sum_{i}^{k} P(Y = y_i) \log_2 P(Y = y_i)$

$H(Y|X) = -\sum_{i}^{n} P_X(x_i) H_{Y|X = x_i}$

$= \sum_{i=1}^{n} P_X(x_i) \left(-\sum_{j=1}^{m} P_{Y|X}(y_j|x_i) \log P_{Y|X}(y_j|x_i) \right)$

Decision tree will overfit: in order to avoid
- ↳ fixed depth
- fixed number of leaves
- use random forest
- pruning
- early stopping

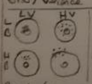
High entropy	Low entropy
Y is from a uniform like distribution, flat histogram, values sampled from it are less predictable	Y is from a varied (peaks and valleys) dist. Histogram has many lows and highs, values sampled from it are more predictable

- standart decision trees have no learning bias, training set error is always zero! (noise)

→ decision tree can represent any Boolean func.

Linearly sep. → 100% accuracy → lin. classifier
↳ Boolean func. → decision tree

Bias/Variance Tradeoff



- simple learners are good
  - NB, LR, low variance, don't usually overfit
- simple lea. are bad
  - high bias, can't solve hard prob.

Bagging (Bootstrap Aggregating)

$$\text{Var}(\text{Bagging}(L(x, D))) = \frac{\text{Var}(L(x, D))}{N}$$

Random Forests: ensemble method designed for decision tree classifier

two sources of randomness: Bagging and Random input vectors"

⇒ Bagging method: each tree is grown using a bootstrap sample of training data

⇒ Random vector method: At each node, best split is chosen from a random sample of m attributes instead of all attributes

Bagging reduces variance by avg./bagging has little effect on bias. Can we average and reduce bias? Yes: Boosting!

Boosting

we weak learner to create strong learner

$H(x) = \text{sign}\left(\sum_i^t \alpha_i h_i(\vec{x}) \right)$

Minimize the error:

$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$

$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$

$$w = \begin{cases} \frac{1}{2} \cdot \frac{1}{\varepsilon} \cdot w_{old} \to \text{wrong} \\ \frac{1}{2} \cdot \frac{1}{1-\varepsilon} \cdot w_{old} \to \text{correct} \end{cases}$$

Boosting vs. Bagging
- resample data points, reweights data points.
- weight of each classifiers is the same, dependent on class. accuracy
- only variance reduction, both bias and variance reduced

Clustering:

k-means: always terminate,

is the clustering any good?
- Global dissimilarity only useful for comparing clustering.

Spectural clustering

$w_{ij} = \left\{ e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}} \right.$ → control size of neighborhood

small - σ: nearby points

min. cut: $cut(A, B) = \sum_{u \in A, v \in B} W(u, v)$, with $A \cap B = \emptyset$

k-means vs spect. clust

Applying k-means to Laplacian eigenvectors allows us to find cluster with non-convex boundary.

agglomerative

hierarchical: still have to choose # of clust.

$n \text{ leafs} = (2n-3)! / [(2(n-2)).(n-2)!]$

single link: potentially long and skinny clusters
complete link: tight clusters
avg. link: robust against noise
+ provides a hierar. of clusters, clusters have adaptive shapes
- may have unbalanced clusters

Good clustering

Internal criterion	external criteria
intra-class ↗ inter-class ↘ the measured quality of clust. depends obj. repr. and the similarity measure used	· purity entropy of classes in clusters ↓ $\frac{1}{N} \cdot \sum_k \max_j \lvert w_k \cap c_j \rvert$

PCA

PCA: orthogonal projection of the data onto a lower-dim. linear space that...
- maximizes variance of projected data
- minimizes mean squared distance between
  - data-point, - projections

PCA #1: points in the direction of the largest variance, each subsequent principal component is orthogonal to previous one

PCA Algorithm I: Given centered data $x_1 \ldots x_m$ compute the principal vectors

$w_1 = \underset{\|w\|=1}{\text{argmax}} \frac{1}{m} \sum_{i=1}^{m} \{ (w^T x_i)^2 \}$ 1^{st} PCA

x' PCA reconstruction: $w_1 w_1^T x_i$

PCA basis vectors = the eigenvectors of Σ,

Larger eigenvalue → more important eigenvectors

PCA Alg. II: Given data $x_1, \ldots x_m$ compute covariance matrix Σ,

$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x_i - \bar{x})(x - \bar{x})^T$

eigenvalues = $|\lambda I - A|$ 'nın kökleri

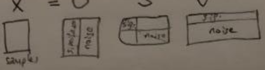
PCA alg. III: (SVD of the data matrix)

Singular Value Decomp. of the centered data matrix X.

$X = [x_1, \ldots x_m] \in \mathbb{R}^{N \times M}$, M: number of ins. N: dim

$X_{features \times samples} = USV^T$

$X = U \quad S \quad V^T$



columns of U: the principal vectors

Matrix S: diagonal, shows importance of each eigenvector

columns of V^T: the coefficient for reconstructing the samples