Machine Learning

Dimensionality Reduction

Lecturer: Duc Dung Nguyen, PhD.
Contact: nddung@hcmut.edu.vn

Faculty of Computer Science and Engineering
Hochiminh city University of Technology

Contents

Matrix calculus

- A vector as a column matrix
- Dot product in matrix notation:

$$\mathbf{a}^T\mathbf{b}$$

- Vector projection of a on b:

$$\mathbf{a}_1 = r.\frac{\mathbf{b}}{||\mathbf{b}||}$$

$$r = \frac{\mathbf{a}.\mathbf{b}}{||\mathbf{b}||}$$

$\mathrm{a}.\mathrm{b}$ is a linear combination of a's dimensions.

- Matrix differentiation:

$$y = \Psi(x)$$

$$y_i = \Psi_i(x)$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

y is an $m \times 1$ matrix, x is an $1 \times n$ matrix

- **Proposition 1:**

$$\alpha = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T)$$

$\mathbf{x}$ is $n \times 1$, $\mathbf{A}$ is $n \times n$, $\mathbf{A}$ does not depend on $\mathbf{x}$

- **Proposition 1:**

$$\alpha = \mathbf{x}^T \mathbf{A} \mathbf{x}$$
$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T)$$

$\mathbf{x}$ is $n \times 1$, $\mathbf{A}$ is $n \times n$, $\mathbf{A}$ does not depend on $\mathbf{x}$

- **Proof**

$$\alpha = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j$$

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} a_{ik} x_i$$

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \mathbf{A}^T + \mathbf{x}^T \mathbf{A} = \mathbf{x}^T (\mathbf{A}^T + \mathbf{A})$$

- **Proposition 2: A** is symmetric

$$\alpha = \mathbf{x}^T \mathbf{A}\mathbf{x}$$
$$\frac{\partial \alpha}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{A}$$

$\mathbf{x}$ is $n \times 1$, $\mathbf{A}$ is $n \times n$, $\mathbf{A}$ does not depend on $\mathbf{x}$

- **Proposition 3: A** is symmetric

$$\alpha = x^T A x$$

$$\left(\frac{\partial \alpha}{\partial x}\right)^T = 2Ax$$

x is $n \times 1$ A is $n \times n$, A does not depend on x

- Eigenvalues and eigenvectors:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

$\mathbf{A}$ is $n \times n$ (linear transformation)
$\mathbf{v}$ is $n \times 1$
λ is an eigenvalue of $\mathbf{A}$'s
$\mathbf{v}$ is an eigenvector of $\mathbf{A}$'s

Example:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

$$\mathbf{A} = \begin{bmatrix} 19 & 20 & -16 \\ 20 & 13 & 4 \\ -16 & 4 & 31 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}, \quad \lambda = 27$$

- To find eigenvalues:

$$(\mathbf{A} - \lambda \mathbf{l})\mathbf{v} = 0$$

$$det(\mathbf{A} - \lambda \mathbf{l}) = 0$$

- To find eigenvalues:

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{v} = 0$$

$$det(\mathbf{A} - \lambda \mathbf{I}) = 0$$

- Example:

$$\mathbf{A} = \begin{bmatrix} .8 & .3 \\ .2 & .7 \end{bmatrix}$$

$$det(\mathbf{A} - \lambda \mathbf{I}) = det \begin{bmatrix} .8 - \lambda & .3 \\ .2 & .7 - \lambda \end{bmatrix}$$

$\lambda^2 - \frac{3}{2}\lambda + \frac{1}{2} = 0 \rightarrow \lambda_1 = 1$ and $\lambda_2 = \frac{1}{2}$

- To find eigenvetors:

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{v} = 0$$

- Example:

$$\mathbf{A} = \begin{bmatrix} .8 & .3 \\ .2 & .7 \end{bmatrix}$$

$\lambda_1 = 1$ and $\lambda_2 = 1/2$

$$(\mathbf{A} - \mathbf{I})\mathbf{v} = 0 \rightarrow \mathbf{v}_1 = \begin{bmatrix} .6 \\ .4 \end{bmatrix}$$

$$(\mathbf{A} - \frac{1}{2}\mathbf{I})\mathbf{v} = 0 \rightarrow \mathbf{v}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

- Proposition 4: $\mathbf{A}$ is a $n \times n$ symmetric matrix
  - All of its eigenvalues are real
  - There are n linearly independent eigenvectors for $\mathbf{A}$.

- Proposition 5: $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ are linearly independent eigenvectors of $\mathbf{A}$, and $\lambda_1, \lambda_2, ..., \lambda_n$ are their corresponding eigenvalues

$$\mathbf{A} = \mathbf{PDP}^{-1}$$

where

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

Proof:

$$\mathbf{P} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n], \quad \mathbf{P}^{-1}\mathbf{P} = l, \quad \mathbf{P}^{-1}\mathbf{v}_i = \mathbf{e}_i$$

$$\begin{aligned}
\mathbf{P}^{-1}\mathbf{A}\mathbf{P} &= \mathbf{P}^{-1}\mathbf{A}[\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n] \\
&= \mathbf{P}^{-1}[\mathbf{A}\mathbf{v}_1, \mathbf{A}\mathbf{v}_2, ..., \mathbf{A}\mathbf{v}_n] \\
&= \mathbf{P}^{-1}[\lambda_1\mathbf{v}_1, \lambda_2\mathbf{v}_2, ..., \lambda_n\mathbf{v}_n] \\
&= [\lambda_1\mathbf{P}^{-1}\mathbf{v}_1\lambda_2\mathbf{P}^{-1}\mathbf{v}_2], ..., \lambda_n\mathbf{P}^{-1}\mathbf{v}_n] \\
&= [\lambda_1\mathbf{e}_1, \lambda_2\mathbf{e}_2, ..., \lambda_n\mathbf{e}_n] = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}
\end{aligned}$$

Example:

$$\mathbf{A} = \begin{bmatrix} 19 & 20 & -16 \\ 20 & 13 & 4 \\ -16 & 4 & 31 \end{bmatrix}$$

Example:

$$\mathbf{A} = \begin{bmatrix} 19 & 20 & -16 \\ 20 & 13 & 4 \\ -16 & 4 & 31 \end{bmatrix}$$

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} -2 \\ -1 \\ 2 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix}$$

$$\lambda_1 = 27, \quad \lambda_2 = 45, \quad \lambda_3 = -9$$

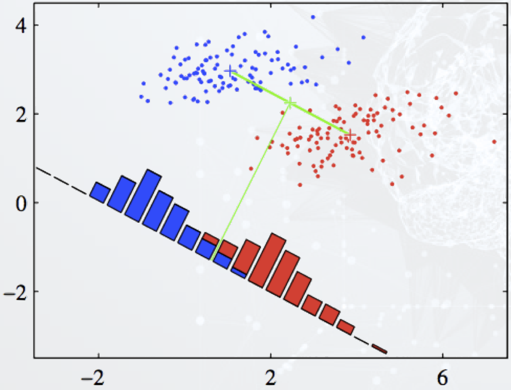
$$\mathbf{P} = \begin{bmatrix} 1 & -2 & 2 \\ 2 & -1 & -2 \\ 2 & 2 & 1 \end{bmatrix}$$

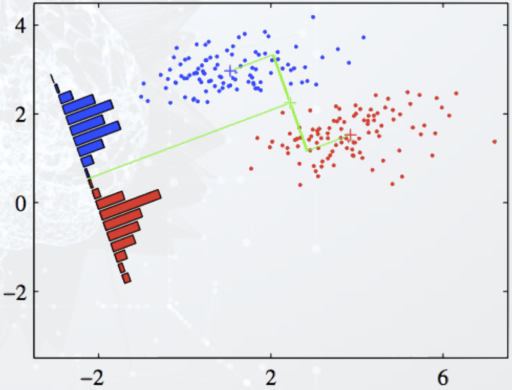
$$\mathbf{P}^{-1} = \frac{1}{9} \begin{bmatrix} 1 & 2 & 2 \\ -2 & -1 & 2 \\ 2 & -2 & 1 \end{bmatrix}$$

LDA

Case 1: **feature combinations** that are sufficient to classify samples.

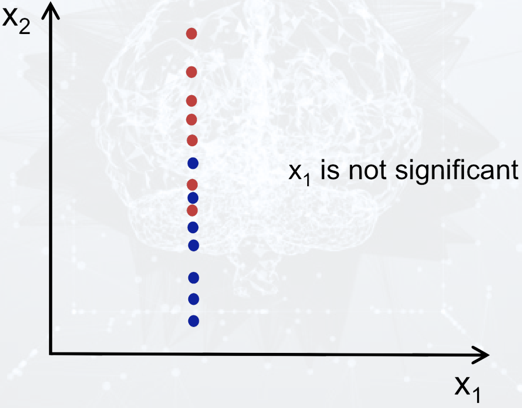
Example 1: linear combination of features.



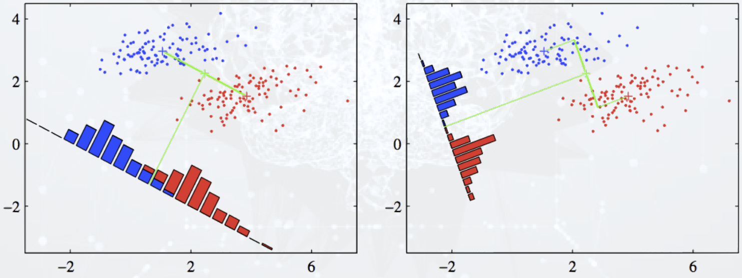


Case 2: a feature with **invariant values** over the samples is not useful for classification.

Example 2:



- To project a high dimensional vector to one dimension (linear combination of features so that:)
  - The **between-class distance** is maximized
  - The **within-class variance** is minimized



To optimize w in: $\mathbf{y} = \mathbf{w}^T \mathbf{x}$

- Two-class problem: N_1 points of class C_1 and N_2 points of class C_2

Linear Discriminant Analysis (LDA)

- Two-class problem: N_1 points of class C_1 and N_2 points of class C_2
- Mean vectors:

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n$$

- Two-class problem: N_1 points of class C_1 and N_2 points of class C_2
- Mean vectors:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n \in C_2} \mathbf{x}_n$$

- Mean of projected data:

$$m_k = \mathbf{w}^T \mathbf{m}_k$$

- Two-class problem: N_1 points of class C_1 and N_2 points of class C_2
- Mean vectors:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n \in C_2} \mathbf{x}_n$$

- Mean of projected data:

$$m_k = \mathbf{w}^T \mathbf{m}_k$$

- To be maximized:

$$(m_2 - m_1)^2$$

- Within-class variance:

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- To be minimized:

$$s_1^2 + s_2^2$$

- To be maximized:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- To be maximized:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Between-class covariance matrix:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

- To be maximized:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Between-class covariance matrix:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

- Within-class covariance matrix:

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

- To be maximized:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

- To be maximized:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_w\mathbf{w}}$$

- We have

$$\partial J(\mathbf{w})/\partial \mathbf{w} = 0$$

$$\rightarrow 2\mathbf{w}^T\mathbf{S}_W(\mathbf{w}^T\mathbf{S}_B\mathbf{w}) = 2\mathbf{w}^T\mathbf{S}_B(\mathbf{w}^T\mathbf{S}_W\mathbf{w})$$

$$\rightarrow (\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

$$\rightarrow (\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

- Solution:

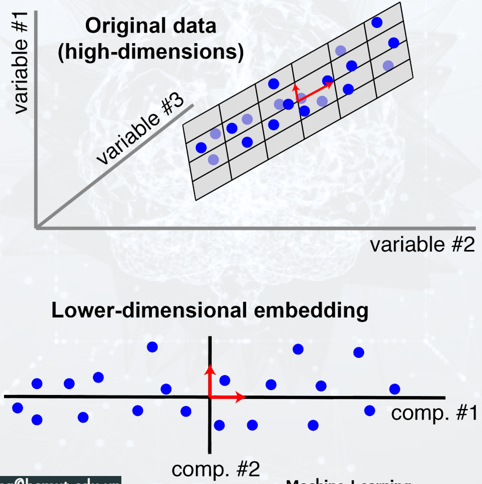
$$(\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  $(\mathbf{w}^T\mathbf{S}_B\mathbf{w})$ and $(\mathbf{w}^T\mathbf{S}_W\mathbf{w})$: scalar factors
  $\mathbf{S}_B\mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}$: in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$
  $\rightarrow \mathbf{w}$ is in the direction of $\mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

PCA

Principal Component Analysis (PCA)

To find the dimensions for which the projected data have the largest variance.



- Consider N points of unlabeled data $\{\mathbf{x}_n\}$

- Consider N points of unlabeled data $\{\mathbf{x}_n\}$
- Mean vector:

$$\mathbf{m} = \frac{1}{N} \sum_{n=1..N} \mathbf{x}_n$$

Principal Component Analysis (PCA)

- Consider N points of unlabeled data $\{\mathbf{x}_n\}$
- Mean vector:

$$\mathbf{m} = \frac{1}{N} \sum_{n=1..N} \mathbf{x}_n$$

- Variance of projected data on dimension $\mathbf{u}_1$:

$$\frac{1}{N} \sum_{n=1..N} (\mathbf{u}_1^T.\mathbf{x}_n - \mathbf{u}_1^T.\mathbf{m}) = \mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1$$

where $\mathbf{S}$ is the data covariance matrix:

$$\mathbf{S} = \frac{1}{N} \sum_{n=1..N} (\mathbf{x}_n - \mathbf{m}).(\mathbf{x}_n - \mathbf{m})^T$$

Principal Component Analysis (PCA)

- To be maximized:

$$\frac{1}{N}\sum_{n=1..N}(\mathbf{u}_1^T.\mathbf{x}_n - \mathbf{u}_1^T.\mathbf{m}) = \mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1$$

  with the constraint on the unit vector $\mathbf{u}_1$:

$$\mathbf{u}_1^T.\mathbf{u}_1 = 1$$

- To be maximized:

$$\frac{1}{N}\sum_{n=1..N}(\mathbf{u}_1^T.\mathbf{x}_n - \mathbf{u}_1^T.\mathbf{m}) = \mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1$$

  with the constraint on the unit vector $\mathbf{u}_1$:

$$\mathbf{u}_1^T.\mathbf{u}_1 = 1$$

- Lagrange function to be maximized:

$$\mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1 + \lambda_1.(1 - \mathbf{u}_1^T.\mathbf{u}_1)$$

Principal Component Analysis (PCA)

- To be maximized:

$$\frac{1}{N}\sum_{n=1..N}(\mathbf{u}_1^T.\mathbf{x}_n - \mathbf{u}_1^T.\mathbf{m}) = \mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1$$

  with the constraint on the unit vector $\mathbf{u}_1$:

$$\mathbf{u}_1^T.\mathbf{u}_1 = 1$$

- Lagrange function to be maximized:

$$\mathbf{u}_1^T.\mathbf{S}.\mathbf{u}_1 + \lambda_1.(1 - \mathbf{u}_1^T.\mathbf{u}_1)$$

- Solution:

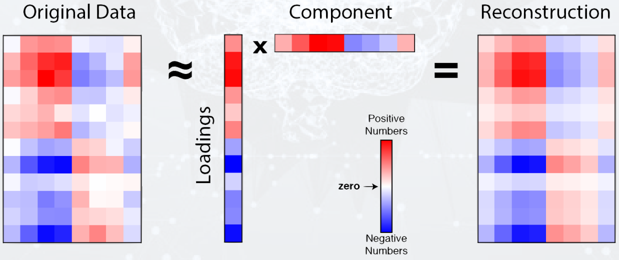
$$\mathbf{S}.\mathbf{u}_1 = \lambda_1.\mathbf{u}_1$$

- Soluution is an eigenvalue and eigenvector:

$$\mathbf{S}.\mathbf{u}_1 = \lambda_1.\mathbf{u}_1$$

- PCA:
  - Compute the data covariance matrix (square and symmetric) in the original space of **D** dimensions.
  - Find **D** eigenvalues and eigenvectors of the covariance matrix.
  - Select the largest $M < D$ eigenvalues and the corresponding eigenvectors to be the new space.

$$\mathbf{S} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_D] \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_D]^{-1}$$



Feature selection & text classification

Feature selection for text classification

This is important because of:

- High dimensionality of text features.
- Existence of irrelevant/noisy features (not only redundant, but also with negative effects).

Representation of a document:

- Bag of words.
- Sequence of words (strings).
- Plus grammatical and semantic elements.
- Probabilistic distribution on topics (topic modeling).

Feature selection for text classification

The most basic and common feature filtering:

- Removal of **stop-words** (the common words such as articles, conjunctions, prepositions,...)
- **Stemming**: different forms (e.g. singular, plural, difference tenses, ...) of the same word are consolidated into a single word.

Feature selection for text classification

- The basic idea of word feature selection: retain only those words that discriminate document classes.
- How to measure the **discriminative power** of a word?

GINI Index

- Consider a word w and assume k class-labels.
- Let $p_n(w)$ be the fraction of those documents containing w that belong to class n:

$$p_n(w) = \text{Prob(a document } \in \text{class } n | \text{it contains } w)$$

$$\sum_{n=1..k} p_n(w) = 1$$

GINI Index

- Consider a word w and assume k class-labels.
- Let $p_n(w)$ be the fraction of those documents containing w that belong to class n:

$$p_n(w) = \text{Prob(a document } \in \text{class } n | \text{it contains } w)$$

$$\sum_{n=1..k} p_n(w) = 1$$

- Gini-index of w:

$$G(w) = \sum_{n=1..k} p_n(w)^2 \in [1/k, 1]$$

the higher, the greater discriminative power of w.

Criticism: $p_n(w)$ may be biased when the global class distribution of documents is usually not uniform.

Gini index

- Let P_n be the fraction of documents that belong to class n.
- Define the normalized probability $q_n(w)$:

$$q_n(w) = \frac{p_n(w)/P_n}{\sum_{m=1..k} p_m(w)/P_m}$$

Gini index

- Let P_n be the fraction of documents that belong to class n.
- Define the normalized probability $q_n(w)$:

$$q_n(w) = \frac{p_n(w)/P_n}{\sum_{m=1..k} p_m(w)/P_m}$$

- New gini-index of w:

$$G(w) = \sum_{n=1..k} q_n(w)^2 \in [1/k, 1]$$

- Prior **inhomogeneity** of a set of documents:

$$E = -\sum_{n=1..k} P_n.\log(P_n)$$

Information gain

- Prior **inhomogeneity** of a set of documents:

$$E = -\sum_{n=1..k} P_n.\log(P_n)$$

- Let $F(w)$ be the fraction of documents that contains w
- Inhomogeneity after using w:

$$E(w) = -F(w).\sum_{n=1..k} p_n(w).\log(p_n(w))$$

$$-(1 - F(w)).\sum_{n=1..k} (1 - p_n(w)).\log(1 - p_n(w))$$

Information gain

- Prior **inhomogeneity** of a set of documents:

$$E = -\sum_{n=1..k} P_n.\log(P_n)$$

- Let $F(w)$ be the fraction of documents that contains w
- Inhomogeneity after using w:

$$E(w) = -F(w).\sum_{n=1..k} p_n(w).\log(p_n(w))$$

$$-(1-F(w)).\sum_{n=1..k} (1-p_n(w)).\log(1-p_n(w))$$

- Information gain

$$I(w) = E - E(w)$$

Mutual information

- How a word w and a class n are correlated?

Mutual information

- How a word w and a class n are correlated?
- Expected co-occurrence of w and class n:

$$P_n.F(w)$$

- True co-occurrence of w and class n:

$$p_n(w).F(w)$$

Mutual information

- How a word w and a class n are correlated?
- Expected co-occurrence of w and class n:

$$P_n.F(w)$$

- True co-occurrence of w and class n:

$$p_n(w).F(w)$$

- Mutual information of w and class n:

$$M_n(w) = \log(\frac{p_n(w).F(w)}{P_n.F(w)}) = \log(\frac{p_n(2)}{P_n})$$

- Mutual information of w and class n:

$$M_n(w) = \log(\frac{p_n(w).F(w)}{P_n.F(w)}) = \log(\frac{p_n(2)}{P_n})$$

- $M_n(w) = 0$: w is not relevant to class n.
- $M_n(w) > 0$: w is positively correlated to class n.
- $M_n(w) < 0$: w is negatively correlated to class n.

Mutual information

- Mutual information of w and class n:

$$M_n(w) = \log(\frac{p_n(w).F(w)}{P_n.F(w)}) = \log(\frac{p_n(2)}{P_n})$$

- Average and maximum values of $M_n(w)$:

$$M_{avg}(w) = \sum_{n=1..k} P - n.M - n(w)$$

$$M_{max}(w) = max_n \{M_n(w)\}$$

t-SNE

What is t-SNE

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: an unsupervised, non-linear technique.

What is t-SNE

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: an unsupervised, non-linear technique.
- Primarily used for **data exploration** and visualizing high-dimensional data.

What is t-SNE

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: an unsupervised, non-linear technique.
- Primarily used for **data exploration** and visualizing high-dimensional data.
- t-SNE gives an intuition of *how data is arranged in high dimensional space*.

- Calculating the **probability of similarity of points** in *high-dimensional space* and calculating the probability of similarity of points in the corresponding *low-dimensional space*.

[1] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

- Calculating the **probability of similarity of points** in *high-dimensional space* and calculating the probability of similarity of points in the corresponding *low-dimensional space*.
- Minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.

[1] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

- Calculating the **probability of similarity of points** in *high-dimensional space* and calculating the probability of similarity of points in the corresponding *low-dimensional space*.
- Minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.
- Minimizes the sum of Kullback-Leibler divergence of overall data points using a gradient descent method.[1]

[1] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

What is t-SNE

- PCA vs. t-SNE?