

Title: Machine Learning Lecture

Speakers: Mohamed Hibat Allah

Collection: Machine Learning 2023/24

Date: April 22, 2024 - 3:15 PM

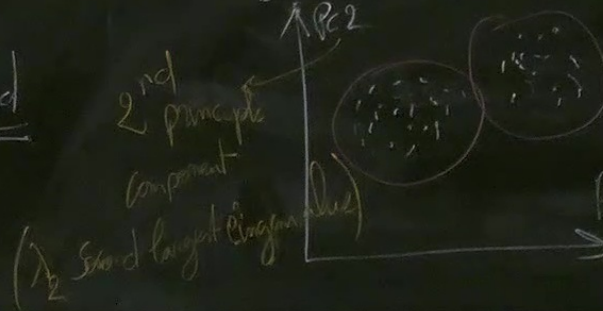
URL: <https://pirsa.org/24040057>

Lecture 9

Last time:

↳ Principle Component analysis (PCA)

↳ Linear method



1st principle component
(λ_1 largest eigenvalue of $X^T X$)

Remark

→ Data normalization

$$\vec{X} = (x_1, x_2, \dots, x_w)$$

Should have similar range (I) not, we need rescaling).

e.g. spin = ± 1 (same bounds)

Comparison
 λ_2 Second largest singular value \rightarrow R_1 (largest singular value)

t-SNE

Given two datapoint \vec{x}_i, \vec{x}_j

$$(j \neq i) \rightarrow P(j|i) = \frac{e^{-\|\vec{x}_i - \vec{x}_j\|^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\vec{x}_i - \vec{x}_k\|^2 / 2\sigma_i^2}}$$

$$P(i|i) = 0$$

\vec{x}_i

The " σ_i " are hyperparameters that are related "Perplexity" \rightarrow See HW2
 Hyperparameter

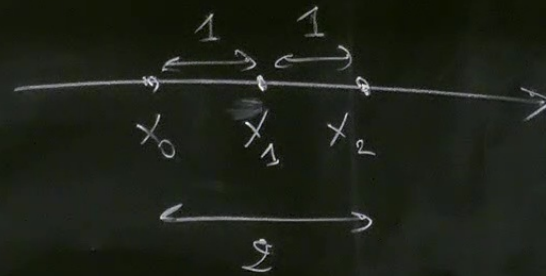
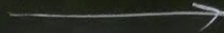
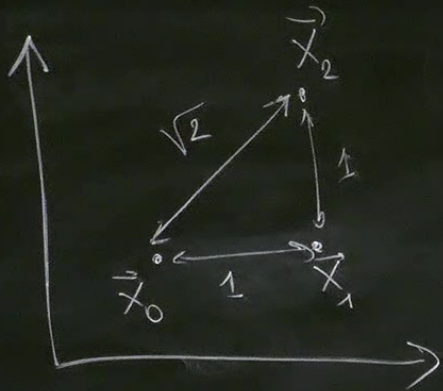
Goal of t-SNE find a lower dimensional representation $\vec{x}' \in \mathbb{R}^n$

that preserves similarity distribution $p(i|j)$

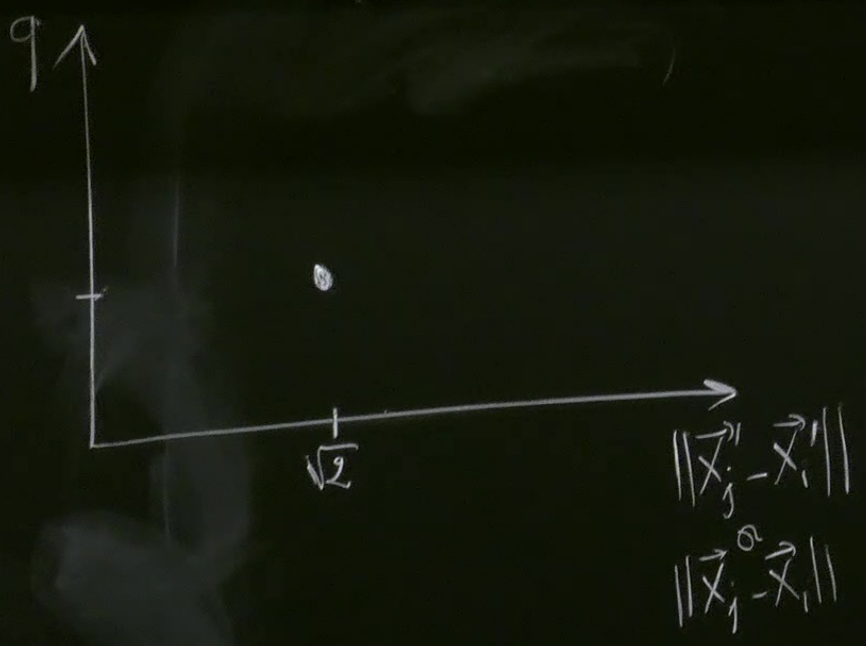
$$q(j|i) = \frac{g(\|\vec{x}'_i - \vec{x}'_j\|)}{\sum_{k \neq i} g(\|\vec{x}'_i - \vec{x}'_k\|)} \quad (i \neq j)$$

$q(i,j)$ defined in a similar way compared $p(i,j)$

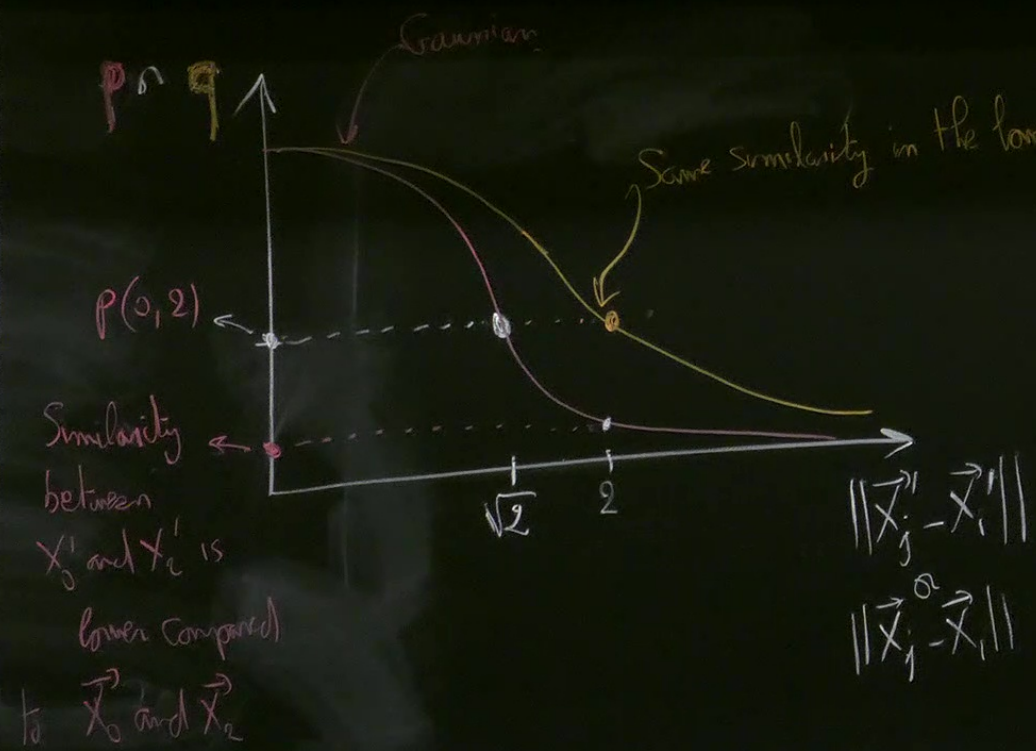
Hyperparameter



M M



M M



$$f(z) = \frac{1}{1+z^2}$$

(t-Student distribution)

Goal of t-SNE find a lower dimensional representation $\vec{x}' \in \mathbb{R}^n$
that preserves similarity distribution $p(i,j)$

$$q(j|i) = \frac{g(\|\vec{x}'_i - \vec{x}'_j\|)}{\sum_{k \neq i} g(\|\vec{x}'_i - \vec{x}'_k\|)} \quad (i \neq j)$$

$q(i,j)$ defined in a similar way compared $p(i,j)$

$$q(i,j) \approx p(i,j)$$

To find a lower-dimensional representation $D = \{ \vec{x}' \}$: iteratively update
the points \vec{x}_i using gradient descent

Trainable parameter.

$$C(P||Q) = D_{KL}(P||Q) \\ = \sum_{i,j=1}^M P(i,j) \log \left(\frac{P(i,j)}{Q(i,j)} \right)$$

Kullback-Liebler (KL)
divergence

To find a lower-dimensional representation $D = \{ \vec{X}_i \}$: iteratively update
the points \vec{X}_i using gradient descent

Trainable parameter.

$$C(p||q) = D_{KL}(p||q) = \sum_{i,j=1}^M p(i,j) \log \left(\frac{p(i,j)}{q(i,j)} \right)$$

Kullback-Liebler (KL)
divergence

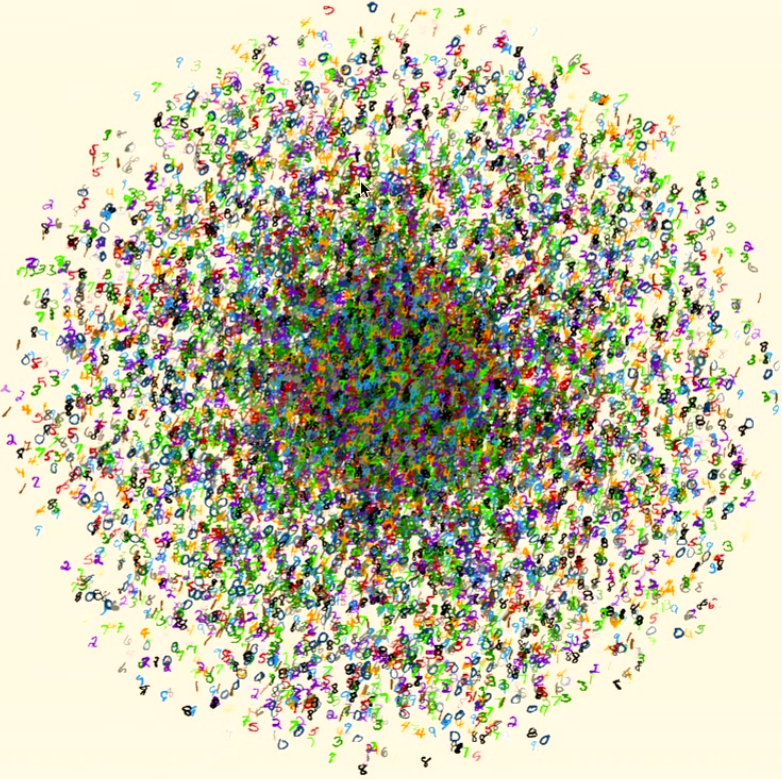
$$0 \log(b) = 0$$

Firefox File Edit View History Bookmarks Tools Window Help

tSNE for the Web x Lecture 9_MLE_principle_gaus: x +

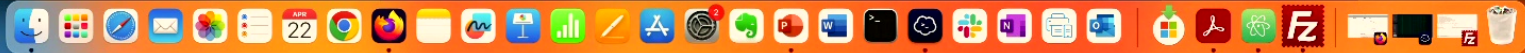
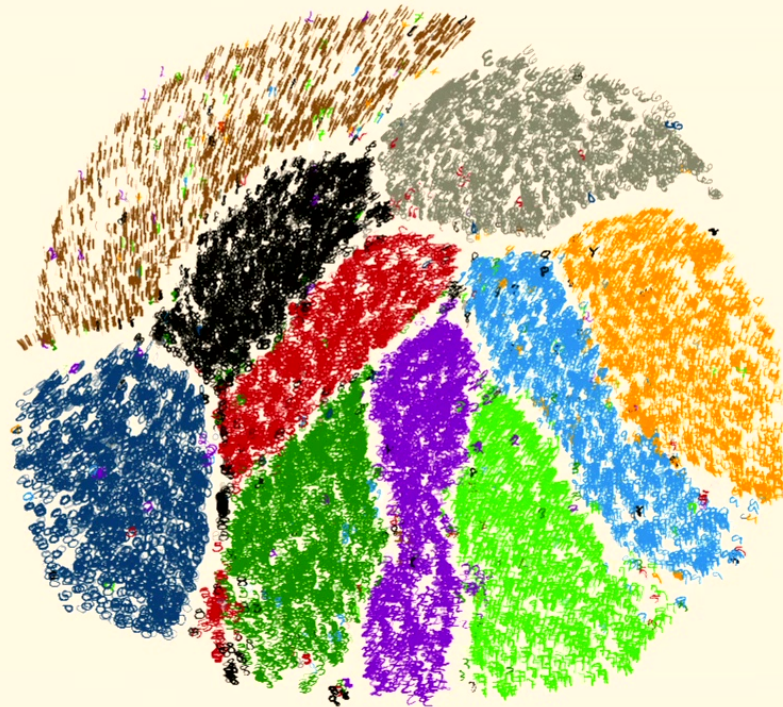
https://nicola17.github.io/tfjs-tsne-demo/

Iterate Transparency Texture Reset



The image shows a t-SNE visualization of handwritten digits. The digits are represented as points in a 2D space, colored by their class. The visualization shows a dense, circular cluster of points, with the digits scattered throughout. The background is a light yellow color. The visualization is centered on the page, with control elements above it. The control elements include a toggle for 'Iterate', a slider for 'Transparency' (set to 50%), and two more toggles for 'Texture' and 'Reset'.

Iterate Transparency Texture Reset




Firefox File Edit View History Bookmarks Tools Window Help

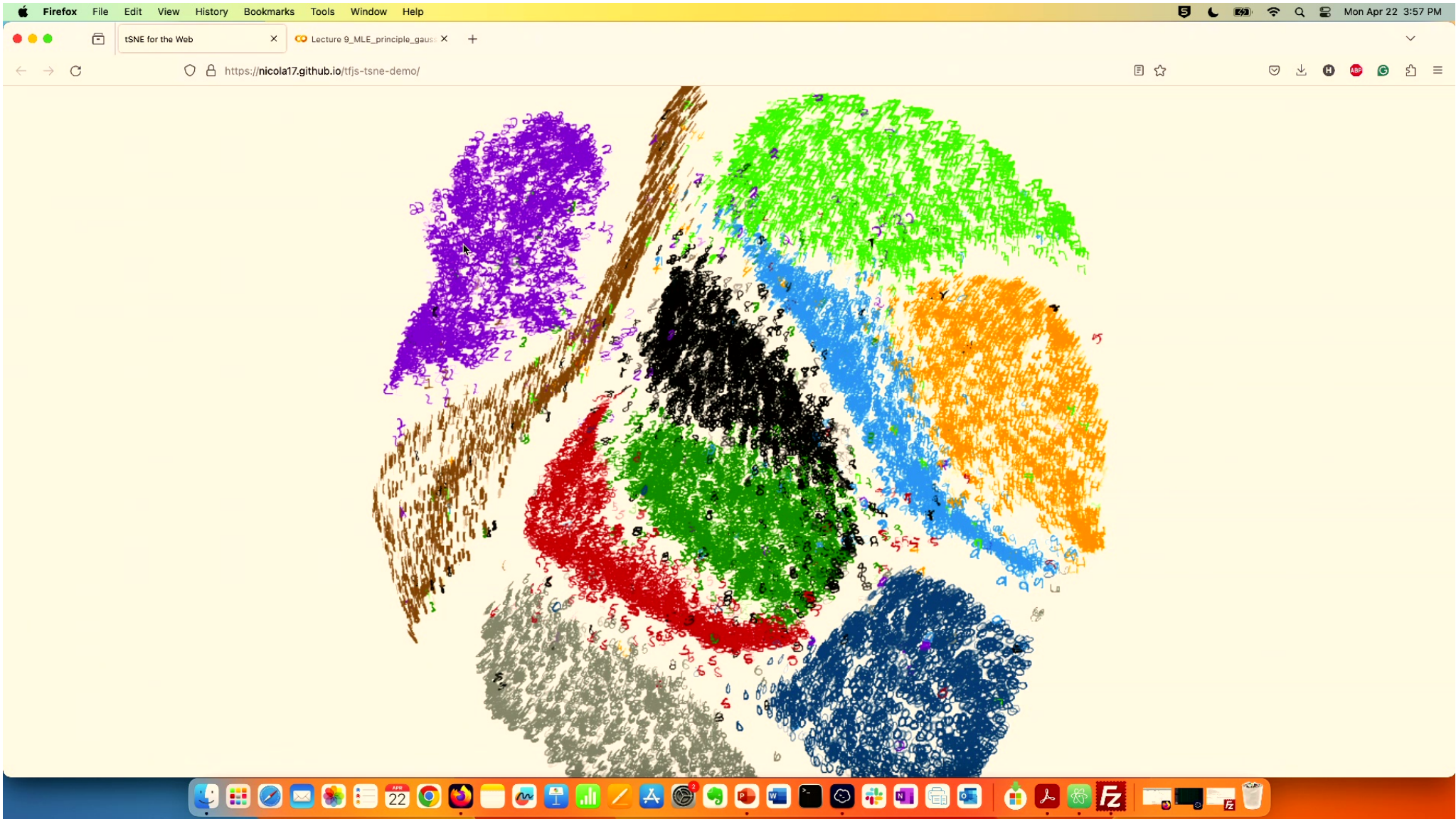
tSNE for the Web x Lecture 9_MLE_principle_gaus x +

https://nicola17.github.io/tfjs-tsne-demo/

Iterate Transparency Texture Reset



The image shows a t-SNE visualization of data points, likely from a Gaussian mixture model, displayed on a web browser. The visualization is a complex, multi-modal distribution of points, colored in various colors (brown, orange, blue, green, purple, red, black, grey). The points are arranged in a butterfly-like shape. The browser interface includes a menu bar (Firefox, File, Edit, View, History, Bookmarks, Tools, Window, Help), a tab bar (tSNE for the Web, Lecture 9_MLE_principle_gaus), and a URL bar (https://nicola17.github.io/tfjs-tsne-demo/). Below the visualization, there are three control sliders: 'Iterate' (checked), 'Transparency' (set to 50%), and 'Texture' (unchecked) with a 'Reset' button (checked).



Hyperparameter

Generative modelling:

↳ Given datapoints $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_M$ coming from $P_{data}(\vec{x})$

Goal, find parameters $\vec{\lambda}$ of a model $P_{\lambda}(\vec{x}) \approx P_{data}(\vec{x})$

$$D_{KL}(P_{data} \parallel P_{\vec{\lambda}}) = \sum_{\vec{x}} P_{data}(\vec{x}) \log \left(\frac{P_{data}(\vec{x})}{P_{\lambda}(\vec{x})} \right)$$

$$= \sum_{\vec{x}} P_{\text{data}}(\vec{x}) \log(P_{\text{data}}(\vec{x}))$$

(-Entropy of data)
(const w.r.t λ)

$$- \sum_{\vec{x}} P_{\text{data}}(\vec{x}) \log(P_{\lambda}(\vec{x}))$$

$$\approx - \frac{1}{M} \sum_{i=1}^M \log(P_{\lambda}(\vec{x}_i))$$

Average Negative log-likelihood

$$\begin{aligned}
\vec{x}^* &= \text{Argmin} \left(D_{KL} (P_{\text{data}} \parallel P_{\vec{x}}) \right) \\
&= \text{Argmin} \left(-\frac{1}{M} \sum_{i=1}^M \log(P(\vec{x}_i)) \right) \\
&= \text{Argmax} \left(\log \left(\prod_{i=1}^M P(\vec{x}_i) \right) \right) \\
&= \text{Argmax} \left(\underbrace{\prod_{i=1}^M P(\vec{x}_i)}_{\text{Likelihood}} \right)
\end{aligned}$$

$$\begin{aligned}
 \vec{x}^* &= \text{Argmin} \left(D_{KL} (P_{\text{data}} \parallel P_{\vec{x}}) \right) \\
 &= \text{Argmin} \left(-\frac{1}{M} \sum_{i=1}^M \log(P(\vec{x}_i)) \right) \\
 &= \text{Argmax} \left(\log \left(\prod_{i=1}^M P(\vec{x}_i) \right) \right) \\
 &= \text{Argmax} \left(\underbrace{\prod_{i=1}^M P(\vec{x}_i)}_{\text{Likelihood}} \right)
 \end{aligned}$$

Maximum Likelihood estimation (MLE) principle.

$$\begin{aligned}
 \vec{x}^* &= \text{Argmin} (D_{KL}(P_{\text{data}} \parallel P_{\vec{x}})) \\
 &= \text{Argmin} \left(-\frac{1}{M} \sum_{i=1}^M \log(P_{\vec{x}}(\vec{x}_i)) \right) \\
 &= \text{Argmax} \left(\log \left(\prod_{i=1}^M P_{\vec{x}}(\vec{x}_i) \right) \right) \\
 &= \text{Argmax} \left(\underbrace{\prod_{i=1}^M P_{\vec{x}}(\vec{x}_i)}_{\text{Likelihood}} \right)
 \end{aligned}$$

Maximum Likelihood estimation (MLE) principle.

$$\vec{x}_{\text{ML}} = \text{argmax}_{\vec{x}} \left(\prod_{i=1}^M P_{\vec{x}}(\vec{x}_i) \right)$$

$$= \sum_{\vec{x}} P_{\text{data}}(\vec{x}) \log(P_{\text{data}}(\vec{x}))$$

(- Entropy of data)
(const w.r.t \vec{x})

$$- \sum_{\vec{x}} P_{\text{data}}(\vec{x}) \log(P_{\lambda}(\vec{x}))$$

$$\approx - \frac{1}{M} \sum_{i=1}^M \log(P_{\lambda}(\vec{x}_i))$$

Average Negative log-likelihood

$$\begin{aligned}
 \vec{\lambda}^* &= \text{Argmin} (D_{KL} (P_{\text{data}} \parallel P_{\vec{\lambda}})) \\
 &= \text{Argmin} \left(-\frac{1}{M} \sum_{i=1}^M \log(P_{\vec{\lambda}}(\vec{x}_i)) \right) \\
 &= \text{Argmax} \left(\log \left(\prod_{i=1}^M P_{\vec{\lambda}}(\vec{x}_i) \right) \right) \\
 &= \text{Argmax} \left(\underbrace{\prod_{i=1}^M P_{\vec{\lambda}}(\vec{x}_i)}_{\text{Likelihood}} \right)
 \end{aligned}$$

Maximum Likelihood estimation (MLE) principle.

$$\vec{\lambda}_{MLE} = \text{argmax}_{\vec{\lambda}} \left(\prod_{i=1}^M P_{\vec{\lambda}}(\vec{x}_i) \right)$$

$P(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_M)$

Firefox File Edit View History Bookmarks Tools Window Help

tsNE for the Web x Lecture 9_MLE_principle_gaus: x

https://colab.research.google.com/drive/1zSD9KbVZy68--b3XS4A1W822AmVk2fpU?authuser=1#scrollTo=h8h8-l7TgF1W

Lecture 9_MLE_principle_gaussian_distribution.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

# True parameters
mu_true = 1
sigma_true = 1

# Generate random data
np.random.seed(0)
data = np.random.normal(mu_true, sigma_true, 100)

[ ] initial_guess = [0.5, 0.1] #Initial ansatz for mu and sigma

sns.histplot(data, kde=False, stat="density", label='Histogram')
x = np.linspace(-2, 4, 300)
plt.plot(x, norm.pdf(x, mu_true, sigma_true), 'r-', lw=2, label='True mu, sigma')
plt.plot(x, norm.pdf(x, initial_guess[0], initial_guess[1]), 'b--', lw=2, label='Initial guess')
plt.legend()
plt.show()
```

Connected to Python 3 Google Compute Engine backend

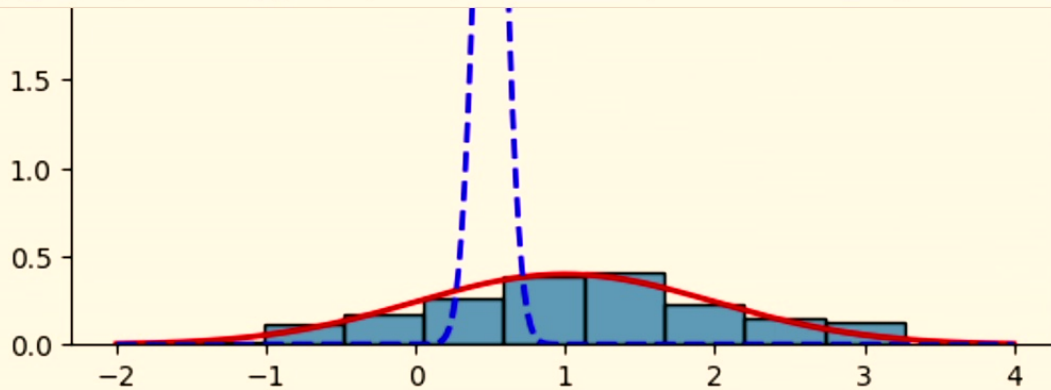
```
Firefox File Edit View History Bookmarks Tools Window Help
Lecture 9_MLE_principle_gaus: X
https://colab.research.google.com/drive/1zSD9KbVZy68--b3XS4A1W822AmVk2fpU?authuser=1#scrollTo=g0R53GZyek33
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

[ ] # True parameters
mu_true = 1
sigma_true = 1

# Generate random data
np.random.seed(0)
data = np.random.normal(mu_true, sigma_true, 100)

▶ initial_guess = [0.5, 0.1] #Initial ansatz for mu and sigma

[ ] sns.histplot(data, kde=False, stat="density", label='Histogram')
x = np.linspace(-2, 4, 300)
plt.plot(x, norm.pdf(x, mu_true, sigma_true), 'r-', lw=2, label='True mu, sigma')
plt.plot(x, norm.pdf(x, initial_guess[0], initial_guess[1]), 'b--', lw=2, label='Initial guess')
plt.legend()
plt.show()
```

```
from scipy.optimize import minimize

def neg_log_likelihood(params):
    mu, sigma = params
    return -np.sum(np.log(norm.pdf(data, mu, sigma)))

result = minimize(neg_log_likelihood, initial_guess, bounds=[(-1, 1), (0.1, 2)])
mle_mu, mle_sigma = result.x

[ ] sns.histplot(data, kde=False, stat="density", label='Histogram')
x = np.linspace(-2, 4, 300)
plt.plot(x, norm.pdf(x, mu_true, sigma_true), 'r-', lw=2, label='True mu, sigma')
plt.plot(x, norm.pdf(x, mle_mu, mle_sigma), 'b--', lw=2, label='MLE mu, sigma')
```

Firefox File Edit View History Bookmarks Tools Window Help

tsne for the web x Lecture 9_MLE_principle_gaus: x

https://colab.research.google.com/drive/1zSD9KbVZy68--b3XS4A1W822AmVk2fpU?authuser=1#scrollTo=VWgVPeEsgyE0

```
+ Code + Text  
plt.plot(x, norm.pdf(x, mle_mu, mle_sigma), 'b--', lw=2, label='MLE mu, sigma')  
plt.legend()  
plt.show()
```

Bin Range	Density
-1.5 to -1.0	0.02
-1.0 to -0.5	0.11
-0.5 to 0.0	0.17
0.0 to 0.5	0.26
0.5 to 1.0	0.39
1.0 to 1.5	0.41
1.5 to 2.0	0.23
2.0 to 2.5	0.15
2.5 to 3.0	0.13