Quantum Expectation Maximization

and other quantum algorithms for learning representations

November 15, 2020

Cambridge University

Alessandro Luongo

In short

We propose a quantum algorithm for

Expectation-Maximization

that fits a

Gaussian Mixture Model

(using quantum linear algebra, QRAM, distance estimation, etc..)

for a matrix $X \in \mathbb{R}^{n \times d}$ in time:

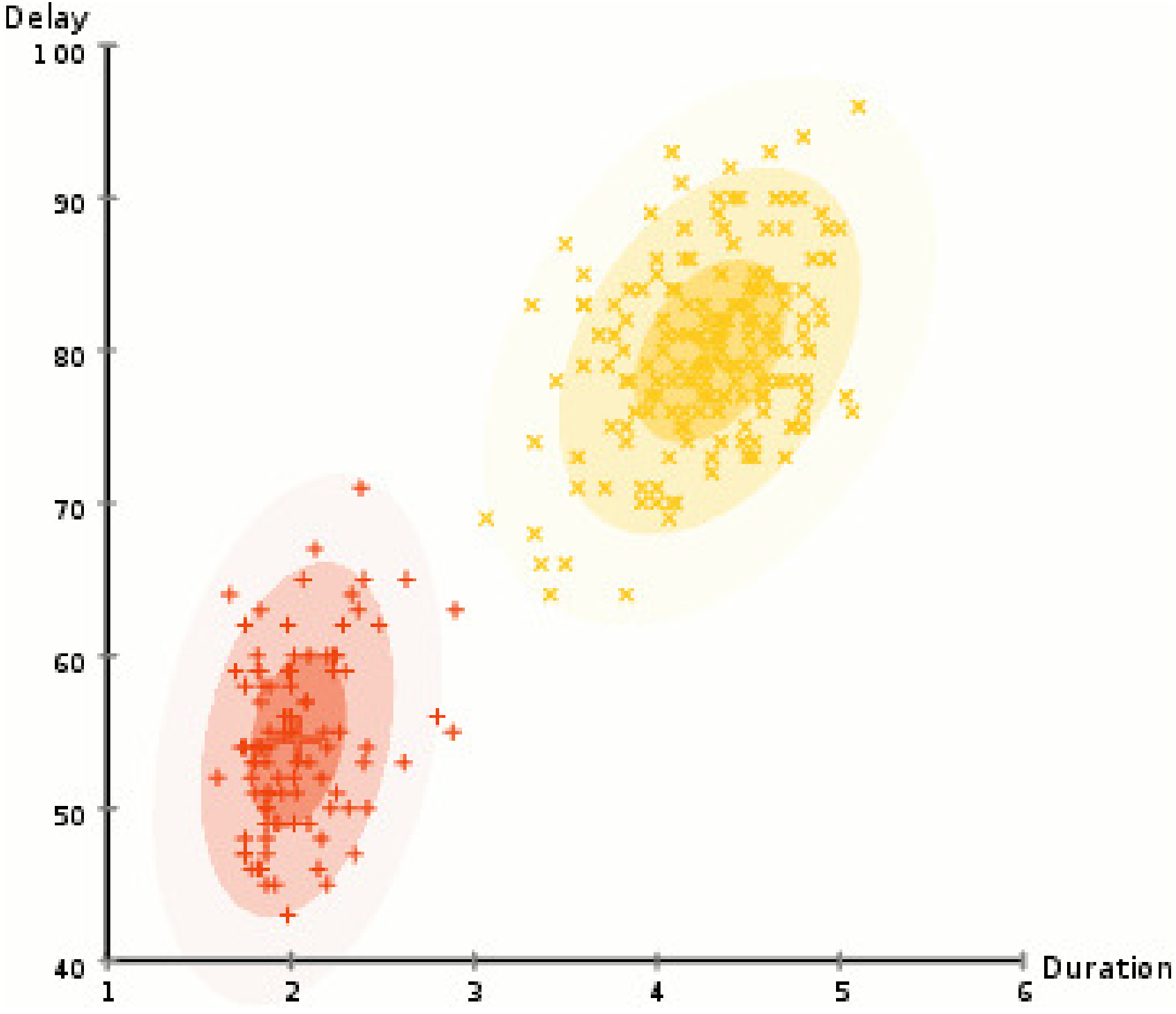
$$O\left(\frac{d^2 k^{4.5} \eta^3 \kappa^2(V) \kappa^2(\Sigma) \mu(\Sigma) \log n}{\delta_\mu^3}\right).$$

Particular case: **q-means**, a quantum algorithm for k-means:

$$\widetilde{O}\left(k^2 d \frac{\eta^{2.5}}{\delta^3} + k^{2.5} \frac{\eta^2}{\delta^3}\right)$$

Papers?

- **q-means**: *https://arxiv.org/abs/1812.03584*
  Iordanis Kerenidis, Jonas Landman, Anupam Prakash
  (Accepted at NeurIPS)
- **QEM**: *https://arxiv.org/pdf/1908.06657*
  Iordanis Kerenidis, Anupam Prakash
- (Also **QGMM** *https://arxiv.org/abs/1908.06655* Hideyuki Miyahara, Kazuyuki Aihara, Wolfgang Lechner)


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

Maximum Likelihood Estimation

Unsupervised data: $V \in \mathbb{R}^{n \times d}$.
Generative models: learn the best (i.e. with Maximum Likelihood) $p(v_i)$ and sample from it.

$$L(V; \gamma) = \prod_{i=1}^{n} p(v_i; \gamma)$$

$$\gamma^*_{MLE} = \arg\max_{\gamma} L(V; \gamma)$$

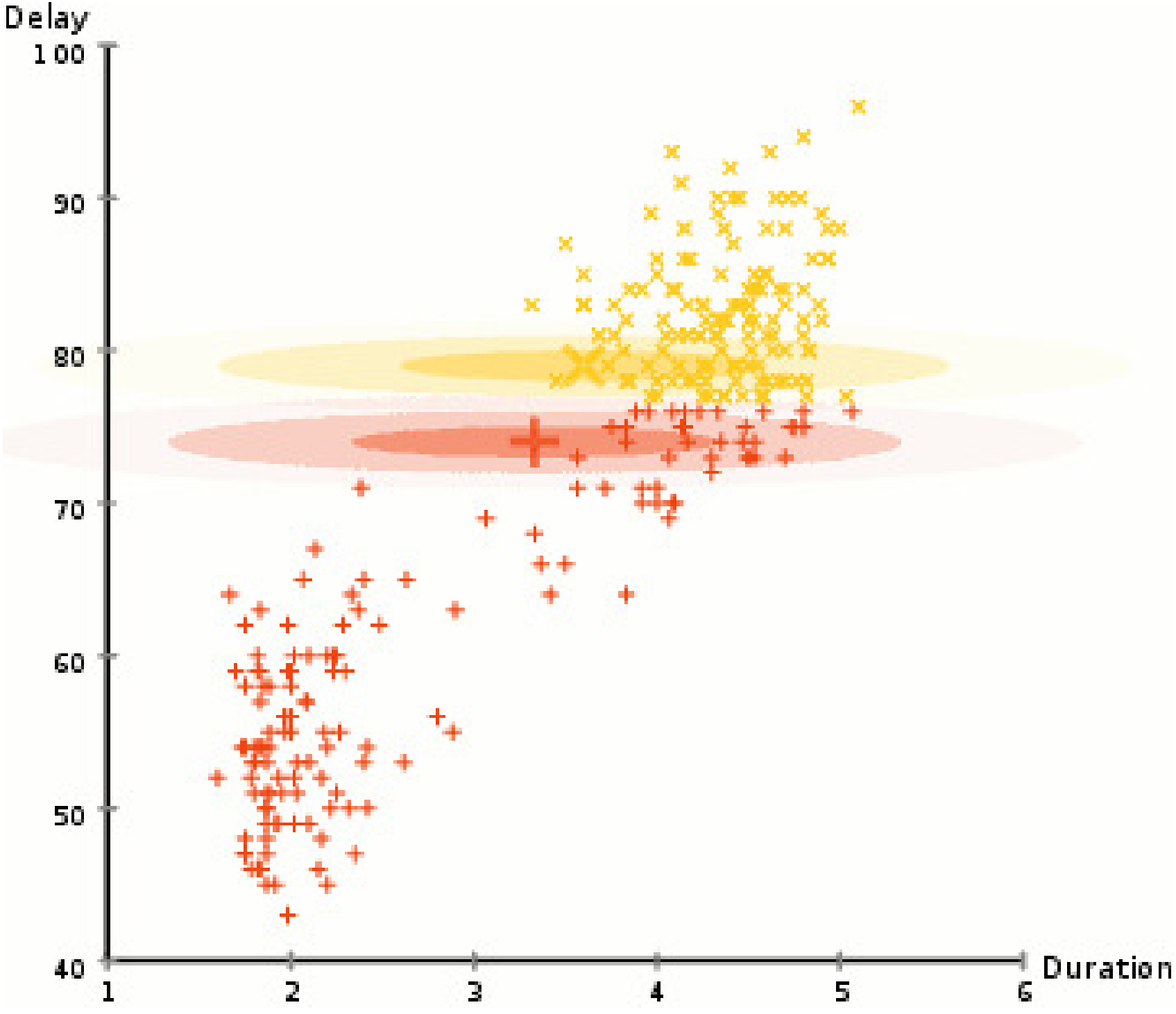
Maximum A Posteriori estimate

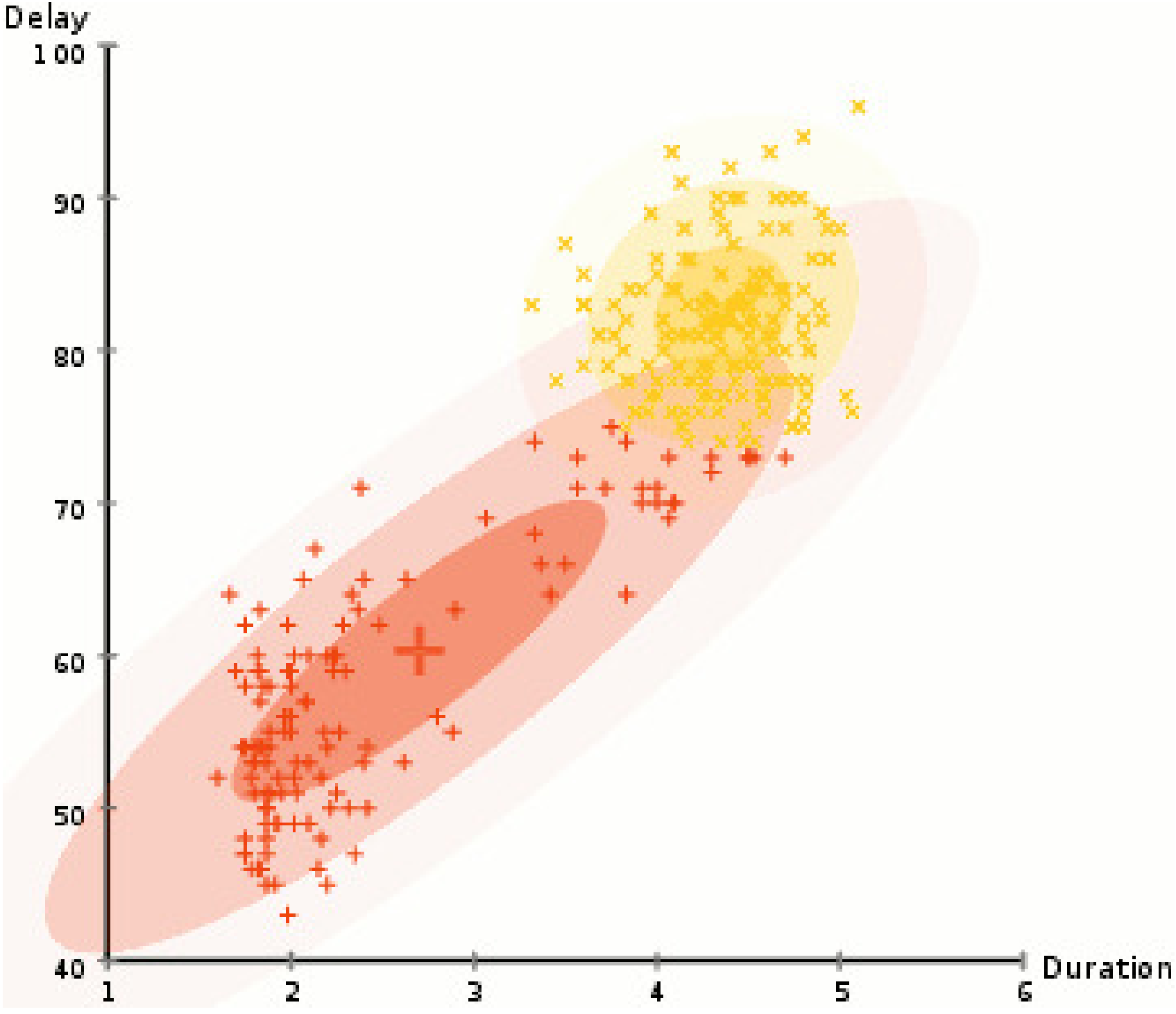
MAP uses a prior $p(\gamma)$ on the model:

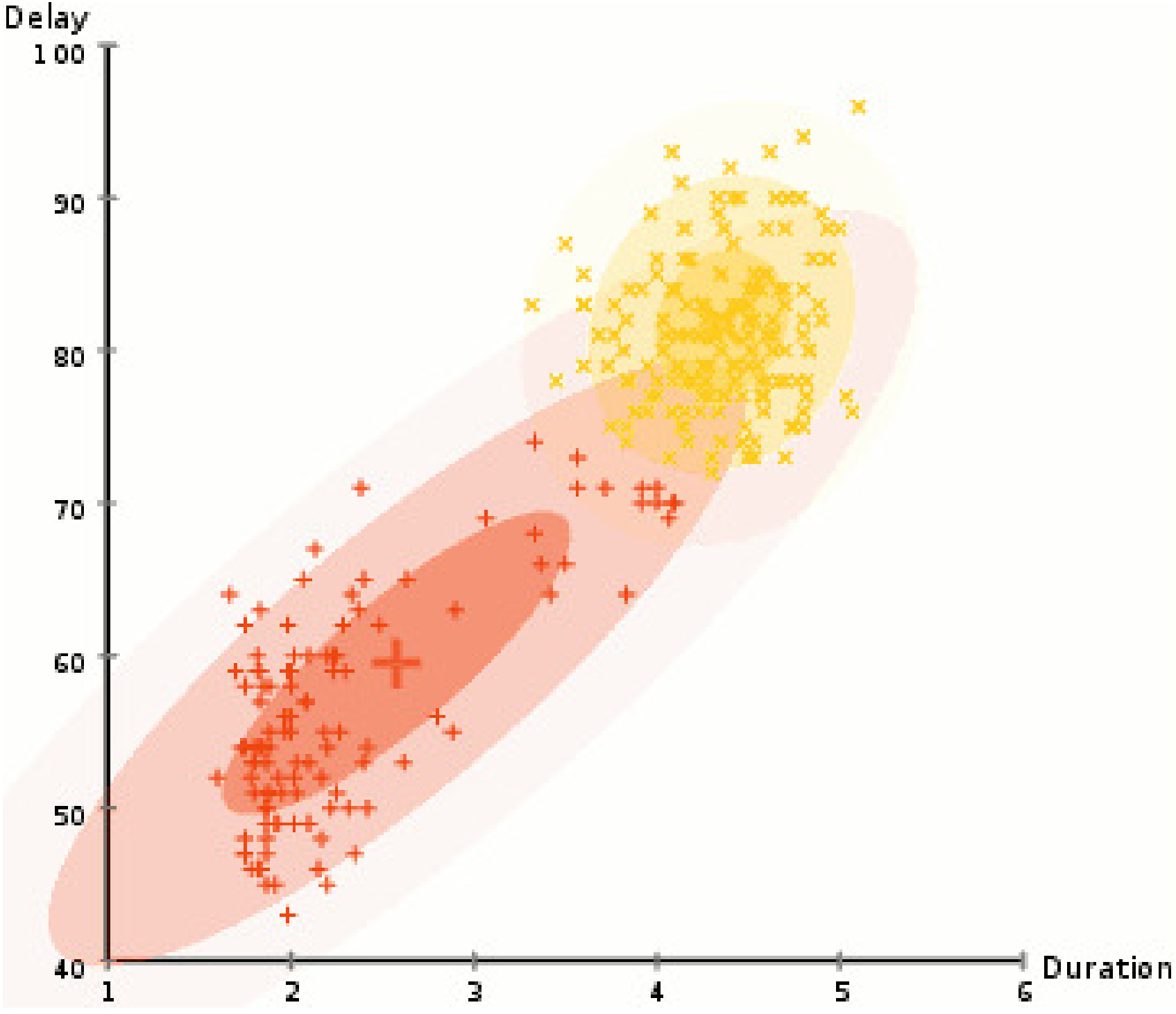
$$L(\gamma; V) = \prod_{i=1}^{n} p(v_i|\gamma)p(\gamma)$$

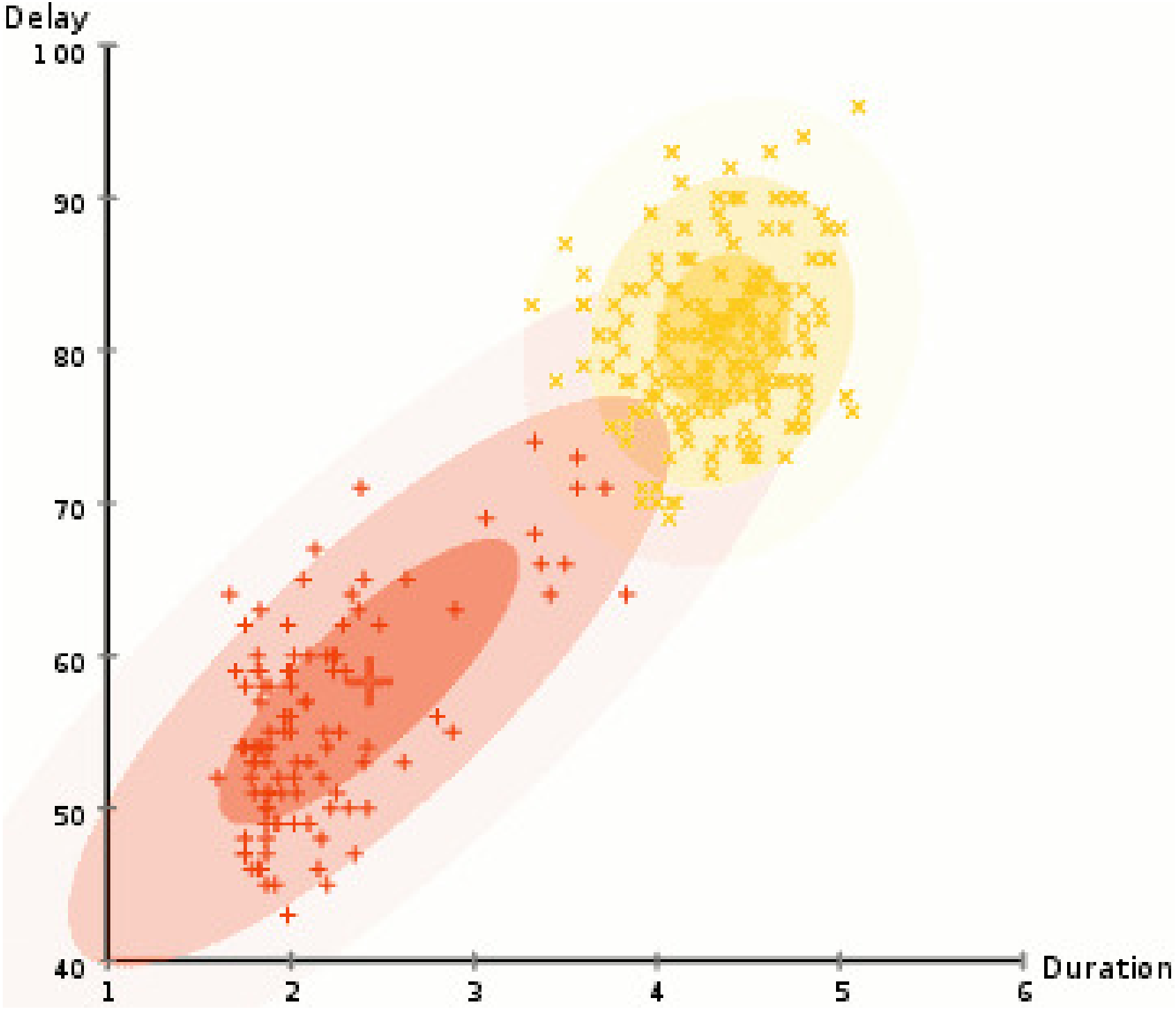
$$\gamma_{MAP} := \arg\max_{\gamma} L(\gamma; V)$$

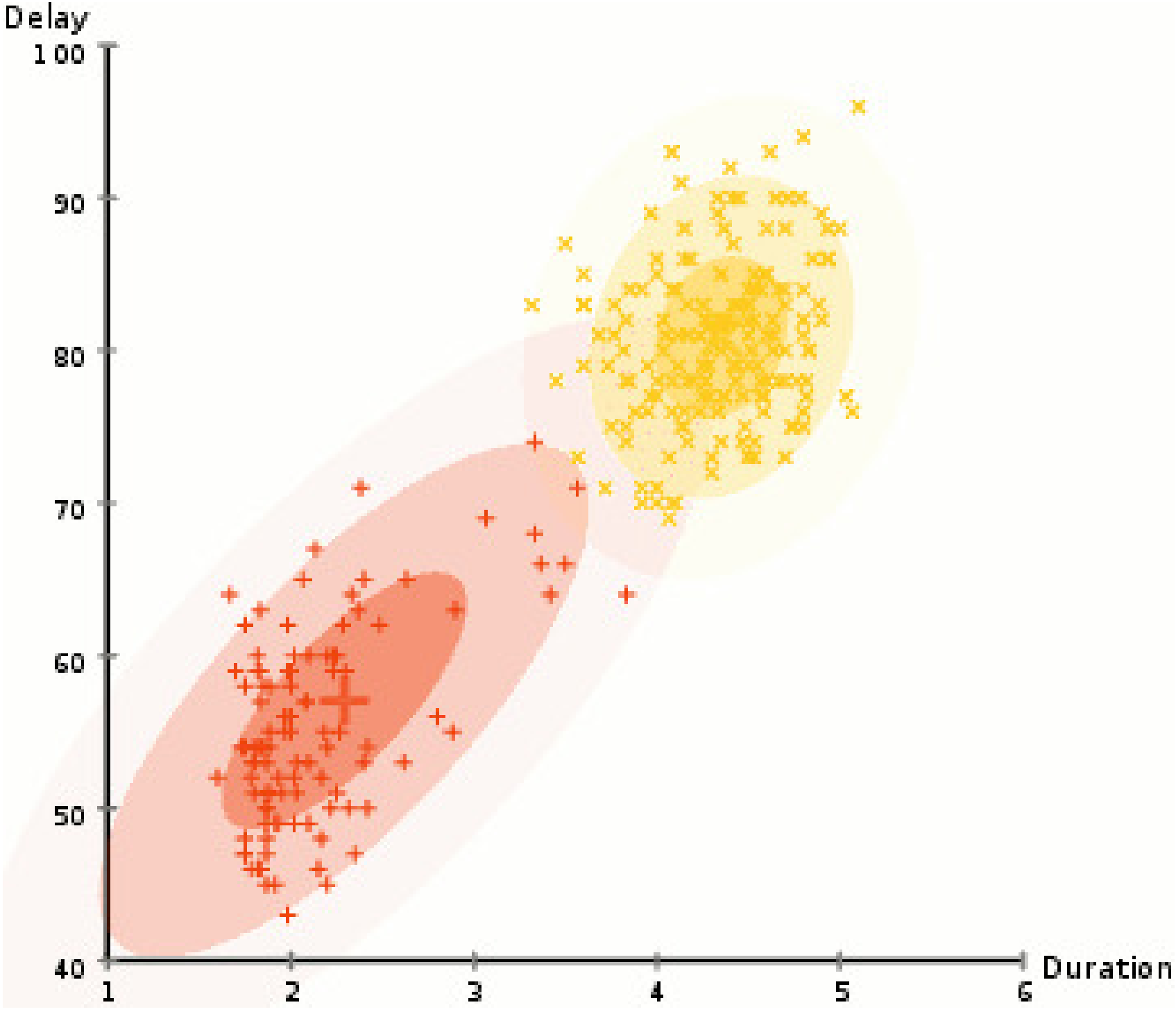
Intuition: Putting a prior (from domain experts) often avoid overfitting and wrong local minima (decrease number of iterations) ;)

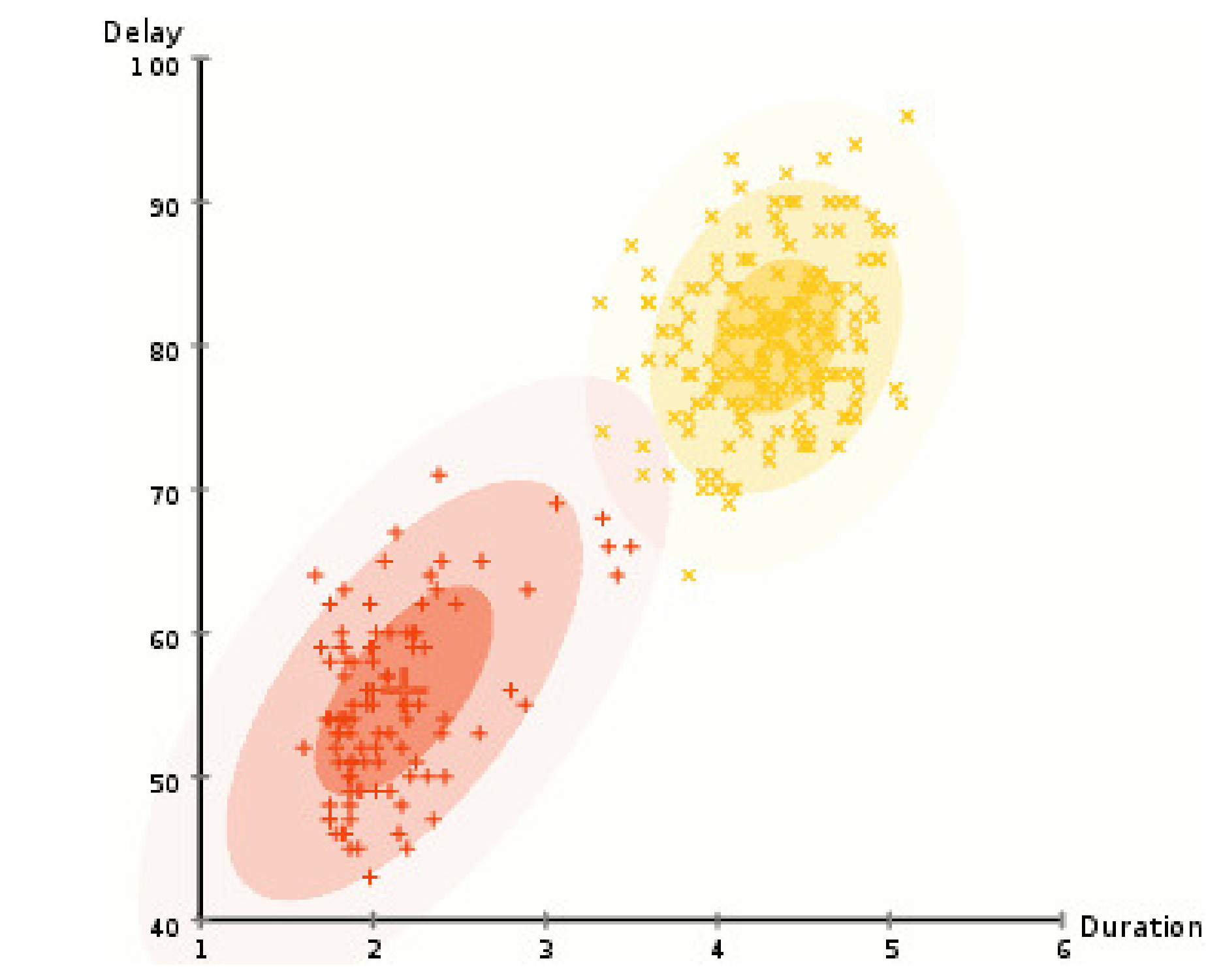

Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

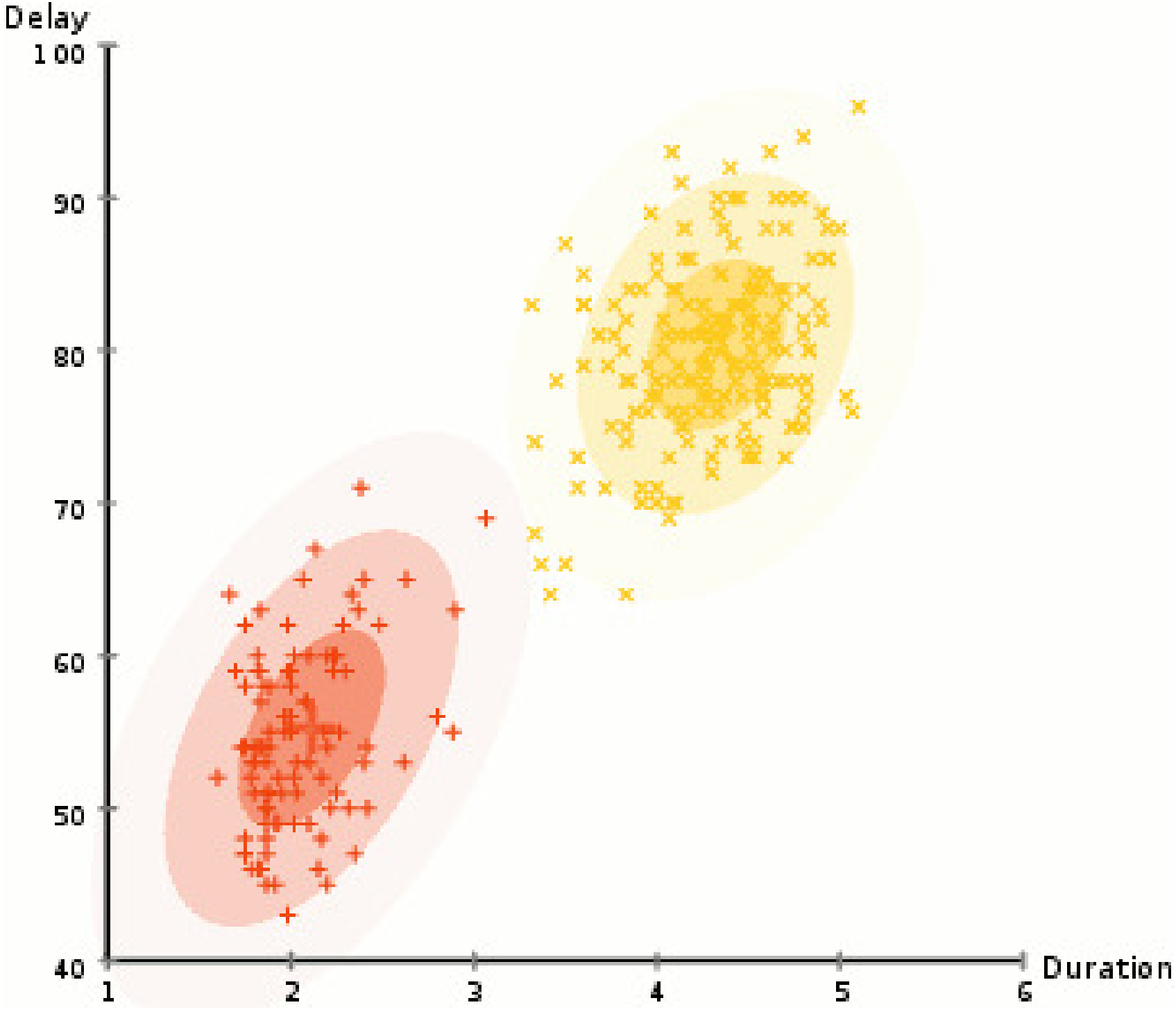

Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

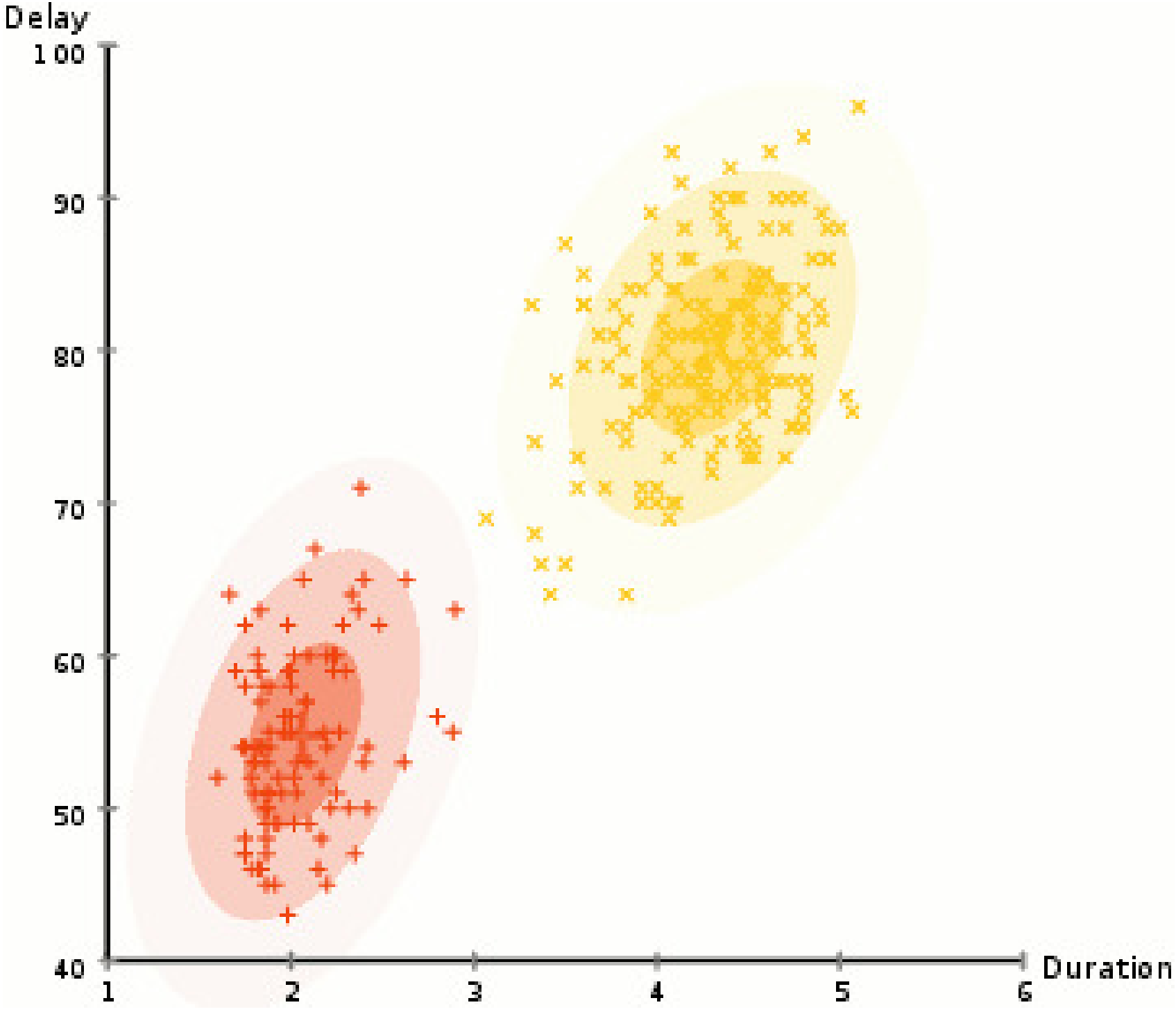

Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

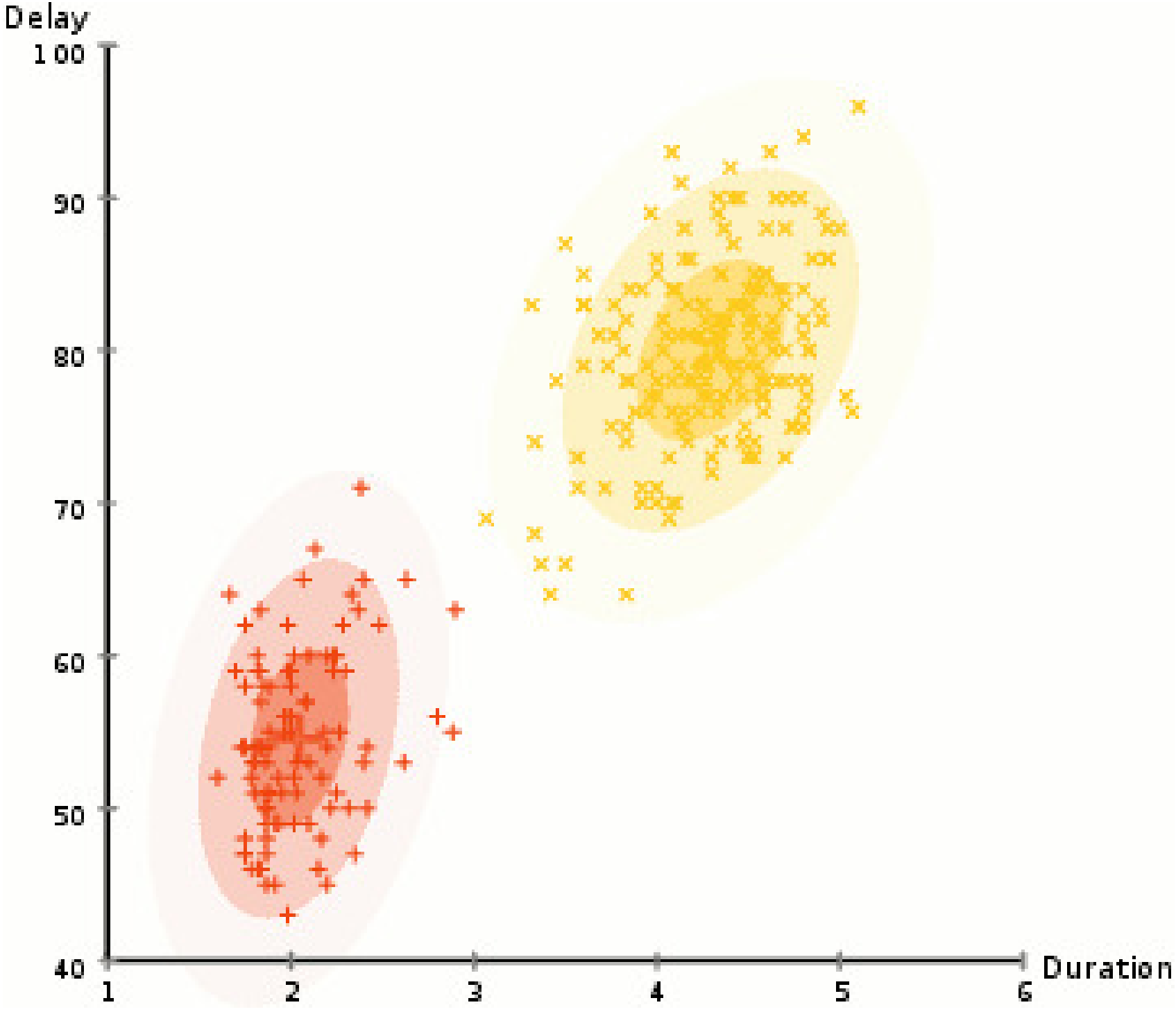

Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

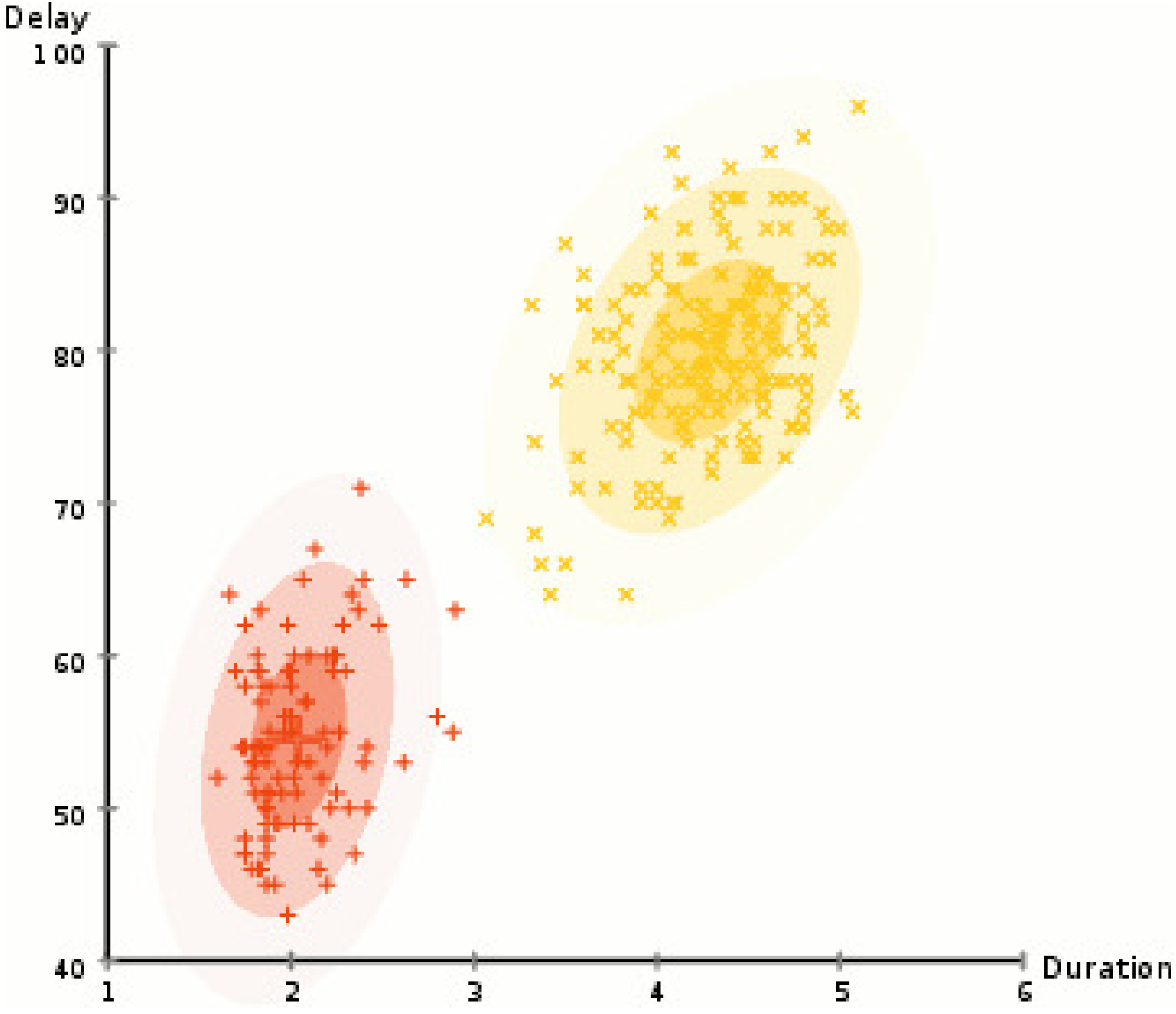

Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration


Delay
100
90
80
70
60
50
40
1
2
3
4
5
6
Duration

fitquantum.py

```
#!/bin/python
import quantum-sklearn #exist only if not measured
import microphone

gmm = quantum-sklearn.mixture.GaussianMixture(n_components=16,
                      max_iter=200, covariance_type='diag', n_init=3)

"perform an extensive recording of spoken voice"
audio, rate = read(microphone.listen())

"extract 20 dim mfcc features from an audio, performs CMS and combines"
" delta to make it 40 dim feature vector"
dataset_ = extract_features(audio, rate)

"scaling, normalization, etc.."
dataset = quantum-sklearn.preprocessing(dataset)

"HERE is where the quantum data analysis is done"
gmm.fit(features)

"store pre trained GMM model of my voice"
pickle.dump(open("scinawa.model", 'wb'), gmm)

print ("training finished")
```

example.py

example.py > ...

```python
#!/bin/python
import sklearn
import microphone

"""pre trained GMM model of my voice"""
scinawa_model = pickle.load(open("scinawa.model", 'rb'))

while 1:
    audio, rate = read(microphone.listen_sentence("sésame, ouvre-toi"))

    """extract 20 dim mfcc features from an audio, performs CMS and combines
    delta to make it 40 dim feature vector"""
    dataset = extract_features(audio, rate)

    """Compute the per-sample average log-likelihood of the given data X."""
    scores = scinawa_model.score(dataset)

    log_likelihood = scores.sum() # l(V) = \sum \log(p(v_i))

    if log_likelihood > 31.337:
        sesame.open = 1
    else:
        sesame.open = 0
        raise Exception("Unauthorized Access.")


```

Pre processing: $O(nd \log nd)$

1. Normalize features by a constant factor
2. Scale features to unit variance
3. Polynomial expansions to capture non-linearities
4. ...
5. Encode dataset as QRAM data structure (trees of angles)

Quantum Machine Learning Toolkit

1. QRAM
2. Quantum Linear Algebra
3. Distance estimation
4. Tomography

(others: amplification techniques, Hamiltonian simulations, etc..)

Loading data in QC

- Let $V \in \mathbb{R}^{n \times d}$,
- Let v_i row of V,w
- Let normalized unit vector $|v_i\rangle = \|v_i\|_2^{-1} v_i$.

$$\text{Then:} \quad U_{QRAM} : |i\rangle |0\rangle \mapsto \frac{1}{\|V\|_F} \sum_{i=0}^{n} \|v_i\| \, |i\rangle |v_i\rangle$$

Loading data in QC

- Let $V \in \mathbb{R}^{n \times d}$,
- Let v_i row of V,w
- Let normalized unit vector $|v_i\rangle = \|v_i\|_2^{-1} v_i$.

$$\text{Then:} \quad U_{QRAM} : |i\rangle |0\rangle \mapsto \frac{1}{\|V\|_F} \sum_{i=0}^{n} \|v_i\| \, |i\rangle |v_i\rangle$$

- Preparation time: $O(nd \log nd)$
- Size: $O(nd \log nd)$
- Execution time / depth: $O(\log nd)$

Quantum linear algebra (post HHL09)

- Encode $v \in \mathbb{R}^d$ (with $\|v\| = 1$) using $\lceil \log d \rceil$ qubits as

$$|v\rangle = \sum_{i=1}^{d} v_i |i\rangle$$

- Encode symmetric matrix $A \in \mathbb{R}^{n \times n}$, $\|A\| = 1$ as a unitary $U \in \mathbb{C}^{2^\ell \times 2^\ell}$ acting on ℓ qubits

$$U = \begin{bmatrix} A/\mu & \cdot \\ \cdot & \cdot \end{bmatrix}, \text{ where } \mu \geq \|A\|$$

- Solving $Ax = b$ means preparing $|A^{-1}b\rangle$, in $\widetilde{O}\left(\mu\kappa(A)\right)$
- Norm $\left\|A^{-1}b\right\|$ is estimated with rel. error ϵ in $\widetilde{O}\left(\mu\kappa(A)/\epsilon\right)$
- Do tomography on $|A^{-1}b\rangle$ in $O\left(d/\delta^2\right)$ to recover δ-approx solution

Distance estimation

What: For $V \in \mathbb{R}^{n \times d}, C \in \mathbb{R}^{k \times d}$ in the QRAM

$$|i\rangle |j\rangle |0\rangle \mapsto |i\rangle |j\rangle |\overline{d(v_i, c_j)}\rangle$$

Trick: Wiebe, N., Kapoor, A., & Svore, K. arXiv:1401.2142.

Error: relative

Time:

$$O\left(\frac{T_{QRAM}\eta \log(1/\Delta)}{\epsilon}\right) = \widetilde{O}\left(\frac{\eta}{\epsilon}\right)$$

where $\eta = \max_{i,j}(\|v_i\|^2 + \|c_j\|^2)$

γ = Gaussian Mixture Models (GMM)

Hidden variables: $y_i \in [k]$ are labels of vectors v_i.

$$p(v_i, y_i) = p(y_i)p(v_i|y_i) = \underbrace{Mult(\theta)}_{\text{mixing weights}} \times \underbrace{\mathcal{N}(\mu_j, \Sigma_j)}_{\text{base probabilities}}$$

1. Multinomial distribution Mult(θ) (a dice) for $\theta \in \mathbb{R}^k$
2. Gaussian distribution $\mathcal{N}(\mu_j, \Sigma_j)$

Assumption on dataset! Dataset (v_i, y_i) is generated by:

1. Sampling a label $y_i \in [k]$ according to Mult(θ),
2. Sampling a vector v_i according to $\mathcal{N}(\mu_{y_i}, \Sigma_{y_i})$.

Model: $\gamma = (\theta, \mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k)$

γ = Gaussian Mixture Models (GMM)

Definition (**Fitting a GMM**)

Fitting a GMM means finding a model that maximize likelihood: $\gamma^* = argmax_\gamma L(V;\gamma)$, that is:

$$\gamma^* = argmax_\gamma \left[L(V;\gamma) = \prod_{i=1}^{n} p(v_i;\gamma) = \prod_{i=1}^{n} \sum_{j=0}^{k} \theta_j N(v_i|\mu_j,\Sigma_j) \right]$$

- Alas, there is not a closed form solution :(... so
- to fit, we use an iterative algorithm called Expectation-Maximization (EM)
- Errors of quantum models:
  - $\left\| \theta - \overline{\theta} \right\| < \delta_\theta$ mixing weights
  - $\left\| \mu_j - \overline{\mu_j} \right\| < \delta_\mu$ base distribution
  - $\left\| \Sigma_j - \overline{\Sigma_j} \right\| < \delta_\mu \sqrt{\eta}$ base distribution

k-means: $O(tndk)$

Find initial centroids μ_j^0.

Repeat until centroids are steady: $|\mu_j^t - \mu_j^{t+1}| \leq \tau$

- **Expectation:**
  - Compute distance for point $v_i, \forall i \in [n]$, and centroid $\mu_j^t, \forall j \in [k]$

$$d(v_i, \mu_j^t)$$

  - Assign points to closer cluster:

$$l(v_i) = \arg\min_{c \in [k]} d(v_i, \mu_j^t)$$

- **Maximization:**
  Update centroid

$$\mu_j^{t+1} = \frac{1}{|C_j|} \sum_{i \in C_j} v_i$$

- t=t+1

GMM: $O(tnd^{2.3}k)$

1: **repeat**

2: **Expectation**

$$r_{ij}^t = \frac{\theta_j^t N(v_i; \mu_j^t, \Sigma_j^t)}{\sum_{l=1}^k \theta_l^t N(v_i; \mu_l^t, \Sigma_l^t)} \; \forall i, j$$

3: **Maximization**

Update the parameters of the model as:

$$\theta_j^{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^n r_{ij}^t$$

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t}$$

$$\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t (v_i - \mu_j^{t+1})(v_i - \mu_j^{t+1})^T}{\sum_{i=1}^n r_{ij}^t}$$

4: t=t+1

5: **until** $|\ell(\gamma^{t-1}; V) - \ell(\gamma^t; V)| < \tau$

Quantum Expectation

We want:

$$r_{ij} = p(v_i|y = j)$$

$$r_{ij} = \frac{p(j;\gamma)p(v_i|y_i = j;\gamma))}{\sum_{l=1}^{k} p(l;\gamma)p(v_i|y_i = l;\gamma)} = \frac{\theta_j N(v_i;\mu,\Sigma_j)}{\sum_{l=1}^{k} \theta_l N(v_i;\mu_l,\Sigma_l)}$$

i.e.

$$|i\rangle\,|j\rangle \mapsto |i\rangle\,|j\rangle\,|r_{ij}\rangle$$

Trick: base probability (i.e. Gaussians) are in exponential family...
r_{ij} is a softmax function, which is Lipschitz-continuous! (many interesting distribution are in exponential family)
Error: additive
Time:

$$O(\frac{k^{1.5}\eta\kappa(\Sigma)}{\delta_\mu})$$

GMM

1: **repeat**

2: **Expectation**

$$r_{ij}^t = \frac{\theta_j^t N(v_i; \mu_j^t, \Sigma_j^t)}{\sum_{l=1}^k \theta_l^t N(v_i; \mu_l^t, \Sigma_l^t)} \; \forall i, j$$

3: **Maximization**

Update the parameters of the model as:

$$\theta_j^{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^{n} r_{ij}^t$$

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t}$$

$$\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t (v_i - \mu_j^{t+1})(v_i - \mu_j^{t+1})^T}{\sum_{i=1}^n r_{ij}^t}$$

4: t=t+1

5: **until** $|\ell(\gamma^{t-1}; V) - \ell(\gamma^t; V)| < \tau$

Quantum Maximization θ^{t+1}

We want:

$$\theta_j^{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^{n} r_{ij}^t$$

Trick: Use Expectation Step and postselection to build:

$$|\sqrt{R}\rangle := \sum_{j=1}^{k} \sqrt{\overline{\theta_j}^{t+1}} \, |\sqrt{R_j}\rangle \, |j\rangle \, . \tag{1}$$

Error:

$$\left\| \overline{\theta}^t - \theta^t \right\| < \delta_\theta$$

Runtime:

$$T_\theta = O\left(k^{3.5} \eta^{1.5} \frac{\kappa^2(\Sigma)\mu(\Sigma)}{\delta_\theta^2} \right)$$

GMM

1: **repeat**

2: **Expectation**

$$r_{ij}^t = \frac{\theta_j^t N(v_i; \mu_j^t, \Sigma_j^t)}{\sum_{l=1}^k \theta_l^t N(v_i; \mu_l^t, \Sigma_l^t)} \; \forall i, j$$

3: **Maximization**

Update the parameters of the model as:

$$\theta_j^{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^n r_{ij}^t$$

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t}$$

$$\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t (v_i - \mu_j^{t+1})(v_i - \mu_j^{t+1})^T}{\sum_{i=1}^n r_{ij}^t}$$

4: t=t+1

5: **until** $|\ell(\gamma^{t-1}; V) - \ell(\gamma^t; V)| < \tau$

Quantum Maximization μ_j^{t+1}

What we want:

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^{n} r_{ij}^t v_i}{\sum_{i=1}^{n} r_{ij}^t}$$

Trick

Let $R_j^t \in \mathbb{R}^n$ be the vector of responsibilities of the points for the Gaussian j at time t, i.e. $(R_j^t)_i = r_{ij}^t$. Then:

$$\mu_j^{t+1} = \frac{V^T R_j^{t+1}}{n\theta_j}$$

.

Error:

$$\left\| \overline{\mu_j}^t - \mu_j^t \right\| < \delta_\mu$$

Time:

$$T_\mu = \widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V))}{\delta_\mu^3} \right)$$

Quantum Maximization Σ^{t+1}

We want:

$$\Sigma_j^{t+1}$$

Trick:

- $|\text{vec}[x_i x_i^T]\rangle = |x_i\rangle\,|x_i\rangle$
- $\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^{n} r_{ij} v_i v_i^T}{n\theta_j} - \mu_j^{t+1}(\mu_j^{t+1})^T$

Error:

$$\left\|\Sigma_j - \overline{\Sigma_j}\right\| < \delta_\mu \sqrt{\eta}$$

Runtime:

$$T_\Sigma := \widetilde{O}\Big(\frac{k d^2 \eta \kappa^2(V)(\mu(V'))}{\delta_\mu^3}\Big)$$

Quantum Likelihood estimation

We want:

$$\mathbb{E}[p(v_i; \gamma^t)]$$

Trick:

- From $p(x|y)$ and $p(y) \Rightarrow$ we know $p(x)$.
- $L(\gamma^t; V) = n\mathbb{E}[p(v_i; \gamma^t)]$

Error: absolute

Runtime:

$$T_\ell = \widetilde{O}\left(k^{1.5}\eta^{1.5}\frac{\kappa^2(\Sigma)\mu(\Sigma)}{\epsilon^2}\right).$$

Theorem: QEM

Theorem (Quantum Expectation-Maximization)

Let $V \in \mathbb{R}^{n \times d}$ stored in QRAM, and $\delta_\theta, \delta_\mu > 0$.
Then, Quantum Expectation-Maximization fits a (MAP or MLE) Gaussian Mixture Model in time:

$$T_{QEM} = \widetilde{O}\left(\frac{d^2 k^{4.5} \eta^3 \kappa^2(V) \kappa^2(\Sigma) \mu(\Sigma)}{\delta_\mu^3}\right),$$

Theorem: QEM

Theorem (Quantum Expectation-Maximization)

Let $V \in \mathbb{R}^{n \times d}$ stored in QRAM, and $\delta_\theta, \delta_\mu > 0$.
Then, Quantum Expectation-Maximization fits a (MAP or MLE) Gaussian Mixture Model in time:

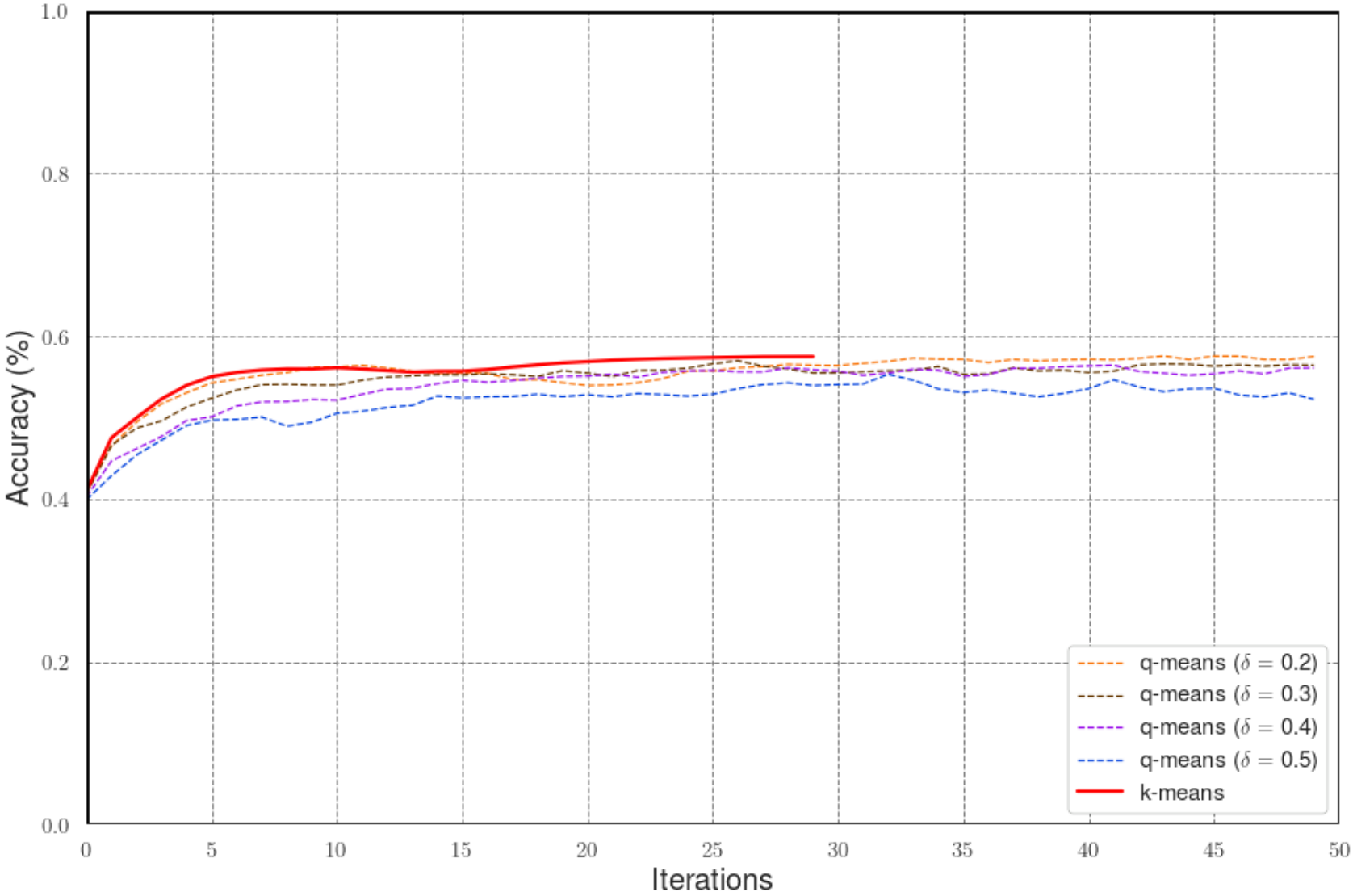
$$T_{QEM} = \widetilde{O}\left(\frac{d^2 k^{4.5} \eta^3 \kappa^2(V) \kappa^2(\Sigma) \mu(\Sigma)}{\delta_\mu^3}\right),$$

Proof.

$$T_{QEM} = O(T_\theta + T_\mu + T_\Sigma + T_\ell).$$

□

q-means on MNIST




Biometric authentication to your Quantum Lab.

Speaker recognition

		$\|\Sigma\|_2$	$\|logdet(\Sigma)\|$	$\kappa^*(\Sigma)$	$\mu(\Sigma)$	$\mu(V)$	$\kappa(V)$
MAP	avg	0.244	58.56	4.21	3.82	2.14	23.82
	max	2.45	70.08	50	4.35	2.79	40.38
ML	avg	1.31	14.56	15.57	2.54	2.14	23.82
	max	3.44	92,3	50	3.67	2.79	40.38

- Classical ML accuracy: 169/170
- Quantum ML accuracy: 167/170
- Max element of Σ_j^{-1} set to 5 via $\kappa = \frac{1}{\lambda_\tau}$

De-quantizations

- Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning -*Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, Chunhao Wang* [1910.06151]
- Quantum-Inspired Classical Algorithms for Singular Value Transformation - *Dhawal Jethwani, François Le Gall, Sanjay K. Singh* [1910.05699]

... but also...

- Arrazola, Juan Miguel, et al. "Quantum-inspired algorithms in practice." arXiv preprint [1905.10415] .

Dequantized recommandation system' runtimes:

$$\widetilde{O}\left(\frac{\|A\|_F^{24}}{\epsilon^{12}\sigma^{24}}\right)$$

Conclusions

- Dataset of 10^{12} to get a speedup ... can be done better?.
- We made well-clusterability assumptions, but we have runtime guarantees on non well clusterable datasets!
- We have also quantum initialization strategies!
- QEM works out-of-the-box for all base distributions in exponential family!
- TODO:
  - Extend QEM to other algorithms (QIBM) etc..
  - A new faster-than-classical QRAM based quantum algo for computing determinant
  - Hidden Markov Models, other Mixture Models, Factor analysis, Message passing algorithm in bayesian networks, and many others.


QUESTIONS
ANSWERED
HERE
EVEN THE
SILLY ONES

(sketch proof)

- Use QRAM to build:

$$\frac{\|v_i\|}{\sqrt{Z_{ij}}} |i\rangle |j\rangle |0\rangle |v_i\rangle + \frac{\|c_j\|}{\sqrt{Z_{ij}}} |i\rangle |j\rangle |1\rangle |c_j\rangle$$

- Hadamard on 3rd qubit. Note that

$$p(1)_{ij} = \frac{1}{2Z_{ij}}(\|v_i\|^2 + \|c_j\|^2 - 2\|v_i\|\|c_j\| \langle v_i\rangle c_j) = \frac{d(v_i, c_j)^2}{2Z_{ij}}$$

- Perform amplitude estimation on L copies.
- Use Median Lemma (Wiebe et. al.)
- Invert circuit to remove garbage (and multiply by $2Z_{ij}$).

Exponential Family

$$p(v|\nu) := h(v)\exp\{o(\nu)^T T(v) - A(\nu)\}$$

where:

- $\nu \in \mathbb{R}^p$ is called the *canonical or natural* parameter of the family,
- $o(\nu)$ is a function of ν (which often is just the identity function),
- $T(v)$ is the vector of sufficient statistics: a function that holds all the information the data v holds with respect to the unknown parameters,
- $A(\nu)$ is the cumulant generating function, or log-partition function, which acts as a normalization factor,
- $h(v) > 0$ is the *base measure* which is a non-informative prior and de-facto is scaling constant.

Slow Feature Analysis

Input: $t \in [t_0, t_1]$, vectors of d different coordinates:

$$x(t) \in \mathbb{R}^d$$

.

Output: a multi-valued function g, i.e. a set of K functions

$$g : \mathbb{R}^d \mapsto \mathbb{R}^K$$

$$g(x(t)) = \left[(g_1(x(t)), g_2(x(t)), ..., g_K(x(t))\right]^T$$

where

$$y(t) = \left[y_1(t), y_2(t), ..., y_K(t)\right]^T = g(x(t))$$

such that these vectors **represent the slowest possible signals**.
Formally: $\forall j \in \{1...J\}$ the delta value:

$$\Delta_j = \Delta(y_j) = \langle \dot{y}_j^2 \rangle_t \text{is minimal.}$$

Additional constraints

- $\langle y_j \rangle_t = 0$ (average $= 0$) The average over time of each component of the signal should be zero.
- $\langle y_j^2 \rangle_t = 1$ (variance $= 1$) The variance over time of each component of the signal should be 1.
- $\forall j' < j : \langle y_{j'} y_j \rangle_t = 0$ (**signals are decorrelated**) . This also introduces an order, such that the first signal is the slowest, the second signal is the second slowest and so on.

Generalized Eigenvalue Problem

$$AW = BW\Lambda$$

- $B = X^T X$
- $A = \dot{X}^T \dot{X}$.

https://projectivesimulation.org