Motivation

How can we project the given data so that the variance in the projected points is maximized?

Principal Component Analysis: Variance Probe

- **X**: \( m \)-dimensional random vector (vector random variable following a certain probability distribution).
- Assume \( E[X] = 0 \).
- Projection of a unit vector \( q \) ((\( qq^T \))\(^{1/2} \) = 1) onto \( X \):
  \[ A = X^T q = q^T X. \]
- We know \( E[A] = E[q^T X] = q^T E[X] = 0 \).
- The variance can also be calculated:
  \[ \sigma^2 = E[A^2] = E[(q^T X)(X^T q)] = q^T E[XX^T] q \]
  = \( q^T \text{covariance matrix} q \)
  = \( q^T R q \).

Principal Component Analysis: Variance Probe (cont’d)

- This is sort of a variance probe: \( \psi(q) = q^T R q \).
- Using different unit vectors \( q \) for the projection of the input data points will result in smaller or larger variance in the projected points.
- With this, we can ask which vector direction does the variance probe \( \psi(q) \) has external value?
- The solution to the question is obtained by finding unit vectors satisfying the following condition:
  \[ R q = \lambda q, \]
  where \( \lambda \) is a scaling factor. This is basically an eigenvalue problem.
PCA

- With an $m \times m$ covariance matrix $R$, we can get $m$ eigenvectors and $m$ eigenvalues:

$$Rq_j = \lambda_j q_j, j = 1, 2, ..., m$$

- We can sort the eigenvectors/eigenvalues according to the eigenvalues, so that

$$\lambda_1 > \lambda_2 > ... > \lambda_m.$$ 

and arrange the eigenvectors in a column-wise matrix

$$Q = [q_1, q_2, ..., q_m].$$

- Then we can write

$$RQ = Q\lambda$$

where $\lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_m)$. 

- $Q$ is orthogonal, so that $QQ^T = I$. That is, $Q^{-1} = Q^T$. 

PCA: Summary

- The eigenvectors of the covariance matrix $R$ of zero-mean random input vector $X$ define the principal directions $q_j$ along with the variance of the projected inputs have extremal values.

- The associated eigenvaluess define the extremal values of the variance probe.

PCA: Usage

- Project input $x$ to the principal directions:

$$a = Q^T x.$$ 

- We can also recover the input from the projected point $a$:

$$x = (Q^T)^{-1}a = Qa.$$ 

- Note that we don’t need all $m$ principal directions, depending on how much variance is captured in the first few eigenvalues: We can do dimensionality reduction.

PCA: Dimensionality Reduction

- Encoding: We can use the first $l$ eigenvectors to encode $x$.

$$[a_1, a_2, ..., a_l]^T = [q_1, q_2, ..., q_l]^T x.$$ 

- Note that we only need to calculate $l$ projections $a_1, a_2, ..., a_l$, where $l \leq m$. 

- Decoding: Once $[a_1, a_2, ..., a_l]^T$ is obtained, we want to reconstruct the full $[x_1, x_2, ..., x_l, ..., x_m]^T$.

$$x = Qa \approx [q_1, q_2, ..., q_l][a_1, a_2, ..., a_l]^T = \hat{x}.$$ 

Or, alternatively

$$\hat{x} = Q[a_1, a_2, ..., a_l, 0, 0, ..., 0]^T.$$ 

with $m - l$ zeros.
**PCA: Total Variance**

- The total variance of the components of the data vector is
  \[ \sum_{j=1}^{m} \sigma_j^2 = \sum_{j=1}^{m} \lambda_j. \]

- The truncated version with the first \( l \) components have variance
  \[ \sum_{j=1}^{l} \sigma_j^2 = \sum_{j=1}^{l} \lambda_j. \]

- The larger the variance in the truncated version, i.e., the smaller
  the variance in the remaining components, the more accurate the
  dimensionality reduction.

**PCA Example**

```matlab
inp=[randn(800,2)/9+0.5;randn(1000,2)/6+ones(1000,2)];
Q=[
    0.70285 -0.71134
    0.71134  0.70285
]
λ=[
    0.14425  0.00000
    0.00000  0.02161
]
```

**PCA’s Relation to Neural Networks: Hebbian-Based Maximum Eigenfilter**

- How does all the above relate to neural networks?

- A remarkable result by Oja (1982) shows that a single linear
  neuron with Hebbian synapse can evolve into a filter for the first
  principal component of the input distribution!

  - Activation:
    \[ y = \sum_{i=1}^{m} w_i x_i \]

  - Learning rule:
    \[ w_i(n + 1) = \frac{w_i(n) + \eta y(n)x_i(n)}{\left( \sum_{i=1}^{m} [w_i(n) + \eta y(n)x_i(n)]^2 \right)^{1/2}} \]

- Expanding the denominator as a power series, dropping the
  higher order terms, etc., we get

  \[ w_i(n + 1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] + O(\eta^2), \]

  with \( O(\eta^2) \) including the second- and higher-order effects of \( \eta \),
  which we can ignore for small \( \eta \).

- Based on that, we get

  \[ w_i(n + 1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] \]

  \[ = w_i(n) + \eta \left( \frac{y(n)x_i(n)}{\sqrt{\sum_{i=1}^{m} [w_i(n) + \eta y(n)x_i(n)]^2}} - \frac{y(n)^2w_i(n)}{\sqrt{\sum_{i=1}^{m} [w_i(n) + \eta y(n)x_i(n)]^2}} \right) \]

  \[ \text{Hebbian term} \quad \text{Stabilization term} \]
Matrix Formulation of the Algorithm

- Activation
  \( y(n) = x^T(n)w(n) = w^T(n)x(n) \)

- Learning
  \( w(n + 1) = w(n) + \eta(n)[x(n) - y(n)w(n)] \)

- Combining the above,
  \[ w(n + 1) = w(n) + \eta(n)[x(n)x^T(n)w(n) - w^T(n)x(n)x^T(n)w(n)w(n)] \]
  represents a nonlinear stochastic difference equation, which is hard to analyze.

Conditions for Stability

1. \( \eta(n) \) is a decreasing sequence of positive real numbers such that
   \[ \sum_{n=1}^{\infty} \eta(n) = \infty, \sum_{n=1}^{\infty} \eta^p(n) < \infty \text{ for } p > 1, \]
   \( \eta(n) \to 0 \text{ as } n \to \infty. \)

2. Sequence of parameter vectors \( w(\cdot) \) is bounded with probability 1.

3. The update function \( h(w, x) \) is continuously differentiable w.r.t. \( w \) and \( x \), and its derivatives are bounded in time.

4. The limit \( \bar{h}(w) = \lim_{n \to \infty} E[h(w, X)] \) exists for each \( w \), where \( X \) is a random vector.

5. There is a locally asymptotically stable solution to the ODE
   \[ \frac{d}{dt}w(t) = \bar{h}(w(t)). \]

6. Let \( q_1 \) denote the solution to the ODE above with a basin of attraction \( B(q) \). The parameter vector \( w(n) \) enters the compact subset \( \mathcal{A} \) of \( B(q) \) infinitely often with prob. 1.

Asymptotic Stability Theorem

- To ease the analysis, we rewrite the learning rule as
  \( w(n + 1) = w(n) + \eta(n)h(w(n), x(n)) \).

- The goal is to associate a deterministic ordinary differential equation (ODE) with the stochastic equation.

- Under certain reasonable conditions on \( \eta, h(\cdot, \cdot), \) and \( w \), we get the asymptotic stability theorem stating that
  \[ \lim_{n \to \infty} w(n) = q_1 \]
  infinitely often with probability 1.

Stability Analysis of Maximum Eigenfilter

Set it up to satisfy the conditions of the asymptotic stability theorem:

- Set the learning rate to be \( \eta(n) = 1/n. \)

- Set \( h(\cdot, \cdot) \) to
  \[ h(w, x) = x(n)y(n) - y^2w(n) = x(n)x^T(n)w(n) - [w^T(n)x(n)x^T(n)w(n)]w(n) \]

- Taking expectaion over all \( x \),
  \[ \bar{h} = \lim_{n \to \infty} E[x(n)x^T(n)w(n) - (w^T(n)x(n)x^T(n)w(n))w(n)] \]
  \[ = Rw(\infty) - [w^T(\infty)Rw(\infty)]w(\infty) \]

- Substituting \( \bar{h} \) into the ODE,
  \[ \frac{d}{dt}w(t) = \bar{h}(w(t)) = Rw(t) - [w^T(t)Rw(t)]w(t). \]
Stability Analysis of Maximum Eigenfilter

- Expanding $w(t)$ with the eigenvectors of $R$,

$$ w(t) = \sum_{k=1}^{m} \theta_k(t)q_k, $$

and using basic definitions

$$ Rq_k = \lambda_k q, \quad q_k^T Rq_k = \lambda_k $$

we get (see next slide for derivation)

$$ \sum_{k=1}^{m} \frac{d\theta_k(t)}{dt} q_k = \sum_{k=1}^{m} \lambda_k \theta_k(t)q_k - \left[ \sum_{l=1}^{m} \lambda_l \theta_l^2(t) \right] \sum_{k=1}^{m} \theta_k(t)q_k. $$

Next, we show $Rw(t) = \sum_{k=1}^{m} \lambda_k \theta_k(t)q_k$, using $Rq_k = \lambda_k q$.

$$ Rw(t) = R \sum_{k=1}^{m} \theta_k(t)q_k $$

$$ = \sum_{k=1}^{m} \theta_k(t)Rq_k $$

$$ = \sum_{k=1}^{m} \lambda_k \theta_k(t)q_k $$

Stability Analysis of Maximum Eigenfilter (cont’d)

Equating the RHS's of the following

$$ \frac{dw(t)}{dt} = \frac{d}{dt} \left( \sum_{k=1}^{m} \theta_k(t)q_k \right), $$

we get

$$ \sum_{k=1}^{m} \frac{d\theta_k(t)}{dt} q_k = \sum_{k=1}^{m} \lambda_k \theta_k(t)q_k - \left[ \sum_{l=1}^{m} \lambda_l \theta_l^2(t) \right] \sum_{k=1}^{m} \theta_k(t)q_k. $$

Stability Analysis of Maximum Eigenfilter (cont’d)

Next, we show

$$ [w^T(t)Rw(t)]w(t) = [ \sum_{l=1}^{m} \lambda_l \theta_l^2(t) ] \sum_{k=1}^{m} \theta_k(t)q_k. $$

$[w^T(t)Rw(t)]w(t) = [w^T(t)Rw(t)] \sum_{k=1}^{m} \theta_k(t)q_k$

$$ = \left[ \sum_{l=1}^{m} \theta_l(t)q_l^T \right] \left[ \sum_{k=1}^{m} \theta_k(t)q_k \right] $$

$$ = \sum_{l=1}^{m} \theta_l(t)q_l^T \left[ \sum_{k=1}^{m} \theta_k(t)q_k \right] $$

$$ = \sum_{l=1}^{m} \theta_l(t)q_l^T R \left[ \sum_{k=1}^{m} \theta_k(t)q_k \right] $$

$$ = \sum_{l=1}^{m} \theta_l(t)q_l^T \left[ \sum_{k=1}^{m} \theta_k(t)q_k \right] $$

$$ = \sum_{l=1}^{m} \theta_l(t) \theta_k(t)q_l^T \lambda_k q_k $$

$$ = \sum_{l=1}^{m} \theta_l(t) \theta_k(t)q_l^T \lambda_k q_k $$

$$ = \sum_{l=1}^{m} \theta_l(t) \theta_k(t) \lambda_l \sum_{k=1}^{m} \theta_k(t)q_k $$

$$ = \sum_{l=1}^{m} \theta_l(t) \theta_k(t) \lambda_l \sum_{k=1}^{m} \theta_k(t)q_k $$

$$ = \sum_{l=1}^{m} \theta_l^2(t) \lambda_l \sum_{k=1}^{m} \theta_k(t)q_k $$

$$ = \sum_{l=1}^{m} \theta_l^2(t) \lambda_l \sum_{k=1}^{m} \theta_k(t)q_k $$

{ Inner sum disappears since $q_l^T q_k = 0$ for $l \neq k$ and $= 1$ for $l = k$ }
Stability Analysis of Maximum Eigenfilter (cont’d)

• Factoring out $q_k$, we get

$$\frac{d\theta_k(t)}{dt} = \lambda_k \theta_k(t) - \left[ \sum_{l=1}^{m} \lambda_l \theta_l^2(t) \right] \theta_k(t).$$

• We can analyze the above in two cases (details in following slides):

  - Case I: $k \neq 1$
    In this case, $\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)} \to 0$ as $t \to \infty$, by using
    \[
    \frac{d\theta_k(t)}{dt} \quad \text{above to derive} \quad \frac{d\alpha_k(t)}{dt} = - (\lambda_1 - \lambda_k) \alpha_k(t). \\
    \]
    positive!

  - Case II: $k = 1$
    In this case, $\theta_1(t) \to \pm 1$ as $t \to \infty$, from
    \[
    \frac{d\theta_1(t)}{dt} = \lambda_1 \theta_1(t) \left[ 1 - \theta_1^2(t) \right]. \\
    \]

Stability Analysis of Maximum Eigenfilter (cont’d)

Case I (in detail): $k \neq 1$

• Given

$$\frac{d\theta_k(t)}{dt} = \lambda_k \theta_k(t) - \left[ \sum_{l=1}^{m} \lambda_l \theta_l^2(t) \right] \theta_k(t). \quad (1)$$

• Define $\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)}$.

• Derive

$$\frac{d\alpha_k(t)}{dt} = \frac{1}{\theta_1(t)} \frac{d\theta_1(t)}{dt} - \frac{\theta_k(t)}{\theta_1^2(t)} \frac{d\theta_1(t)}{dt} \quad (2)$$

• Plug in (1) above into (2). (Both $d\theta_k(t)/dt$ and $d\theta_1(t)/dt$.)

• Finally, we get: $\frac{d\alpha_k(t)}{dt} = - (\lambda_1 - \lambda_k) \alpha_k(t)$, so $\alpha_k(t) \to 0$ as $t \to \infty$.  

22

Stability Analysis of Maximum Eigenfilter (cont’d)

Case II: $k = 1$

\[
\frac{d\theta_1(t)}{dt} = \lambda_1 \theta_1(t) \left[ 1 - \theta_1^2(t) \right]. \\
\]

Using results from Case I ($\alpha_l \to 0$ for $l \neq 1$ and $t \to \infty$), $\theta_1(t) \to \pm 1$ as $t \to \infty$, from

$$\frac{d\theta_1(t)}{dt} = \lambda_1 \theta_1(t) \left[ 1 - \theta_1^2(t) \right].$$

23

Stability Analysis of Maximum Eigenfilter (cont’d)

Recalling the original expansion

$$w(t) = \sum_{k=1}^{m} \theta_k(t) q_k,$$

we can conclude that

$$w(t) \to q_1, \quad \text{as} \quad t \to \infty.$$

where $q_1$ is the normalized eigenvector associated with the largest eigenvalue $\lambda_1$ of the covariance matrix $R$.

• Other conditions of stability can also be shown to hold (see the textbook).
Summary of Hebbian-Based Maximum Eigenfilter

Hebbian-based linear neuron converges with probability 1 to a fixed point, which is characterized as follows:

- Variance of output approaches the largest eigenvalue of the covariance matrix \( \mathbf{R} \) \((y(n)\) is the output):
  \[
  \lim_{n \to \infty} \sigma^2(n) = \lim_{n \to \infty} E[Y^2(n)] = \lambda_1
  \]

- Synaptic weight vector approaches the associated eigenvector
  \[
  \lim_{n \to \infty} \mathbf{w}(n) = \mathbf{q}_1
  \]
  with
  \[
  \lim_{n \to \infty} \|\mathbf{w}(n)\| = 1.
  \]

Generalized Hebbian Algorithm for full PCA

- Sanger (1989) showed how to construct a feedforward network to learn all the eigenvectors of \( \mathbf{R} \).

- Activation
  \[
  y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n), j = 1, 2, \ldots, l
  \]

- Learning
  \[
  \Delta w_{ji}(n) = \eta \left[ y_j(n)x_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n)y_k(n) \right],
  \]
  \[
  i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, l.
  \]