In [1]:

```
## settings ↔
```

\usepackage*amssymb*

3.6. Generalization: Principal Axis as Optimization problem

Good descriptions often result from a minimization of an error function.

3.6.1 The Mean as Principal Point

The mean vector

$$\vec{\mu} = \langle \vec{x} \rangle_{\vec{x}} = \int_{\mathbb{R}^d} \vec{x} p(\vec{x}) d\vec{x}$$

(= $\frac{1}{N} \sum_{\alpha} \vec{x}^{\alpha}$ in the case of a discrete dataset)

minimizes the squared error $E(\vec{w})$:

$$\vec{\mu} = \arg\min_{\vec{w}} \underbrace{\langle \|\vec{x} - \vec{w}\|^2 \rangle_{p(x)}}_{E(\vec{w})}$$

Proof:

$$E(\vec{w}) = \langle (\vec{x} - \vec{w})^{\tau} (\vec{x} - \vec{w}) \rangle_{\vec{x}} = \langle \vec{x}^{\tau} \vec{x} \rangle - 2\vec{w}^{\tau} \langle \vec{x} \rangle + \vec{w}^{\tau} \vec{w}$$

A minimum has to fulfill the conditions for stationarity $\nabla_{\vec{w}} E = \vec{0}$, i.e. here

$$\nabla_{\vec{w}} E = -2\langle \vec{x} \rangle + 2\vec{w} = \vec{0} \Leftrightarrow \vec{w} = \langle \vec{x} \rangle \quad \text{q.e.d.}$$

In [5]:

```
# principal point example: ↔
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

3.6.2. Principal Axes

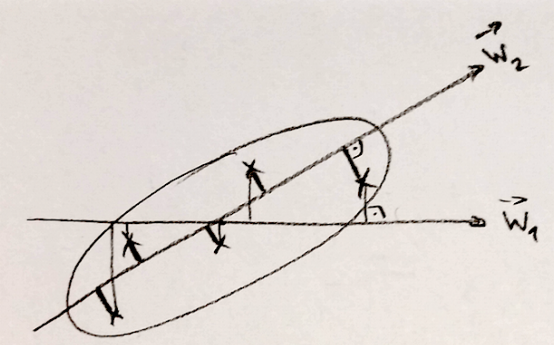
Given:

- centered data (with mean vector $\langle \vec{x} \rangle = \vec{0}$)

Wanted:

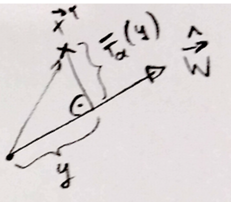
- Axis that minimizes the squared distance to the data

$$E(\vec{w}) = \frac{1}{N} \sum_{\alpha=1}^{N} \min_{y} \left[\|\vec{x}^{\alpha} - \vec{w} \cdot y\|^2 \right] = \frac{1}{N} \sum_{\alpha=1}^{N} \min_{y} F_{\alpha}(y)$$



with

$$F_{\alpha}(y) = \|\vec{x}^{\alpha} - \vec{w} \cdot y\|^2 = (\vec{x}^{\alpha} - \vec{w} \cdot y)^2$$



Step S1: Minimize

$$\begin{aligned} F_{\alpha}(y) &= (\vec{x}^{\alpha} - \vec{w}y)^{\tau}(\vec{x}^{\alpha} - \vec{w}y) \\ &= \vec{x}^{\alpha\tau}\vec{x}^{\alpha} - 2\vec{x}^{\alpha\tau}\vec{w}y + \vec{w}^{\tau}\vec{w}y^2 \end{aligned}$$

Necessary condition is

$$\frac{\partial F_{\alpha}}{\partial y} = 0 \Leftrightarrow 2(\vec{w}^{\tau}\vec{w}y - \vec{x}^{\alpha\tau}\vec{w}) = 0$$

.

$$\Rightarrow \; y^{\alpha} = \frac{\vec{w}^{\tau}\vec{x}^{\alpha}}{\|\vec{w}\|^2}$$

is a latent variable, resp. the projection index for the data point $\vec{x}^{\alpha}$.

Inserting y^{α} into E gives

$$\begin{aligned} E(\vec{w}) = \frac{1}{N} \sum_{\alpha} F_{\alpha}(y^{\alpha}) &= \frac{1}{N} \sum_{\alpha} \vec{x}^{\alpha\tau}\vec{x}^{\alpha} - \frac{2}{N} \sum_{\alpha} \vec{w}^{\tau}\vec{x}^{\alpha} \frac{\vec{w}^{\tau}\vec{x}^{\alpha}}{\vec{w}^{\tau}\vec{w}} + \frac{\vec{w}^{\tau}\vec{w}}{N} \sum_{\alpha} \frac{\vec{w}^{\tau}\vec{x}^{\alpha}\vec{w}^{\tau}\vec{x}^{\alpha}}{(\vec{w}^{\tau}\vec{w})^2} \\ &= \frac{1}{N} \sum_{\alpha} \vec{x}^{\alpha\tau}\vec{x}^{\alpha} - \frac{1}{N}\vec{w}^{\tau} \sum_{\alpha} \frac{\vec{x}^{\alpha}\vec{x}^{\alpha\tau}\vec{w}}{\|\vec{w}\|^2} \end{aligned}$$

Step S2: Minimize E according to $\vec{w}$

(the first term does not depend on $\vec{w}$, so only the second remains)

$$\begin{aligned}\arg\min_{\vec{w}} E(\vec{w}) &= \arg\max_{\vec{w}} \left(\frac{\vec{w}^T}{\|\vec{w}\|} \left(\frac{1}{N} \sum_{\alpha} \vec{x}^{\alpha} \vec{x}^{\alpha\tau} \right) \frac{\vec{w}}{\|\vec{w}\|} \right) \\ &= \arg\max_{\hat{w}} \left(\hat{w}^{\tau} \mathbf{C} \hat{w} \right)\end{aligned}$$

The axis in direction of the first eigenvector $\hat{u}_1$ belonging to the largest eigenvalue λ_1 of the dataset covariance matrix $\mathbf{C}$ minimizes the mean-square error (MSE).

Remarks:

- Generalization on q-dimensional linear manifolds results in principal (hyper-)plains which are spanned by the first q eigenvectors of $\mathbf{C}$ corresponding to the first q eigenvalues in descending order.
- While in 2D (features x, y) linear regression minimizes the y-distance as depending variable over x, here x and y are treated symmetrically, the distance **perpendicular** to the subspace is minimized.

3.6.3. Principal Curve and Principal Surface

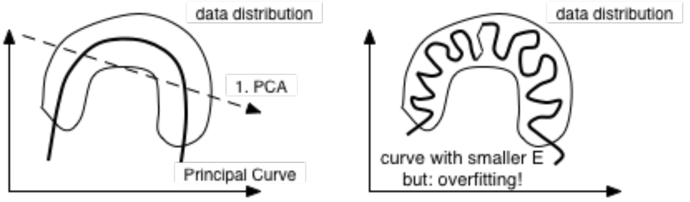
Generalization to nonlinear manifolds are considerably more difficult: depending on the dimention q of the latent variable the result is a principal curve ($q = 1$), principal surface ($q = 2$), etc.

First we look at the special case $q = 1$.

Wanted:

Curve $\vec{f} : \mathbb{R} \rightarrow \mathbb{R}^d, \vec{f}(y)$, which minimizes the Mean-Square Error (MSE)

$$E = \langle \min_y \|\vec{x} - \vec{f}(y)\|^2 \rangle$$



- Problem 1: Parameterization of $\vec{f}(y)$
  - polygonal line (i.e. concatenated linear segments)
  - Polynomial component functions
- Problem 2: limitation of flexibility to avoid trivial interpolations
  - Regularization terms to reward smoothness, e.g.

$$E_G = E + \int \left| \frac{d^2 f}{d\lambda^2} \right|^2 d\lambda$$

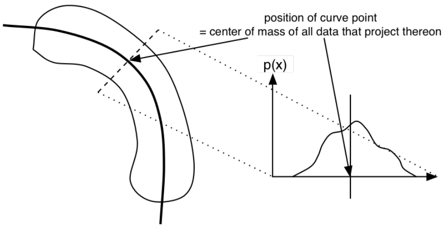
  - length constraints
- Problem 3: Form of the optimzation landscape: Minimization necessary by all parameters
  - Nonlinear optimization problem
  - → risk of getting stuck in local optima.

Principal Curve by Hastie and Stuezle (1989)

Principal Curve was introduced as a curve which fulfills the so called self-consistency condition

$$\vec{f}(y) = \langle \vec{x} | y_f(\vec{x}) = y \rangle_{\vec{x}}$$

- Note that each point of the curve is also the mean of all data points that have this point as their nearest representant.



This means that each point of the curve is the extremal point of a quadratic distance function

$$E(y^\star) = \frac{1}{2} \int_{y_f(\vec{x})=y^\star} (\vec{x} - f(y^\star))^2 p(\vec{x}) d\vec{x}$$

- here the integration is over all points that project on $y^\star$
- that this is an optimization problem follows from our first section (principal point)

Algorithm:

1. start with $\vec{f}(y) = \langle \vec{x} \rangle + \hat{u}_1 y$ (1st axis of PCA)
   - i.e. set for each data point $\vec{x}^\alpha$: $\quad y^\alpha = \hat{u}_1^\tau(\vec{x}^\alpha - \bar{x})$
2. fix y and minimize $\langle \|\vec{x} - \vec{f}(y)\|^2 \rangle$ by setting
   $$f_j(y) = \langle x_j | y_f(\vec{x}) = y \rangle \ \forall\, j$$
   - Note that in case of finite (discrete) data sets we need a neighborhood window so that all data points in the vicinity of y are averaged. A practical choice for that is:
   $$\vec{f}(y) = \frac{\sum_\alpha \vec{x}^\alpha \exp\left(-\frac{(y(\vec{x}^\alpha)-y)^2}{2\sigma^2}\right)}{\sum_\alpha \exp\left(-\frac{(y(\vec{x}^\alpha)-y)^2}{2\sigma^2}\right)}$$
   where σ is the width of the neighborhood window. Practically this is not done for all possible y, but only for the y^α on which data points currently project so that the curve is represented by a polygonal line of N samples.
3. Hold $\vec{f}$ fixed and calculate new $y = y_{\vec{f}}(\vec{x}^\alpha)\ \forall\ \alpha$
4. goto 2 as long as the error is reduced by more than a threshold difference

3.6.4 Principal Surfaces in general

- are the result of a generalization of above principal curve optimization
- instead of a linear combination of eigenvectors we regard a parameterized q-surface $\tilde{\vec{x}}(y_1, \dots y_q; \vec{w}) \in \mathbb{R}^d$.
- Determination of parameters $\vec{w}$ according to an extremality requirement

$$\begin{aligned} D[\vec{w}] &:= \frac{1}{2}\left\langle \left(\vec{x} - \tilde{\vec{x}}(y_1 \dots y_q; \vec{w})\right)^2 \right\rangle \overset{!}{=} \text{minimal} \\ &= \frac{1}{2}\int \left(\vec{x} - \tilde{\vec{x}}(y_1 \dots y_q; \vec{w})\right)^2 p(\vec{x})\, d\vec{x} \;\text{ for given density }\; p(\vec{x}) \end{aligned}$$

- Optimization gives 'Principal Surfaces' (-curves, -manifolds)
- Note: Caution: additional smoothness is to be required, otherwise the manifold will overfit.

Smoothness constraints are sometimes implicit in the chosen parameterization of $\tilde{x}$

- e.g. for instance, if a polynom of low order is chosen.

Computational Procedure:

- Gradient descent

$$\frac{\partial D}{\partial w_j} = -\left\langle \left(\vec{x} - \tilde{\vec{x}}(y_1, \dots, y_q; \vec{w})\right)^\tau \cdot \frac{\partial}{\partial w_j}\tilde{\vec{x}}(y_1, \dots, y_q; \vec{w})\right\rangle_{p(\vec{x})}$$

- In case of finite data sets: "`stochastic approximation"'

$$\left.\frac{\partial D}{\partial w_j}\right|_{\vec{x}^\alpha} \approx -\left(\vec{x}^\alpha - \tilde{\vec{x}}(y_1^\alpha, \dots, y_q^\alpha; \vec{w})\right)^\tau \cdot \frac{\partial}{\partial w_j}\tilde{\vec{x}}(y_1^\alpha, \dots, y_q^\alpha; \vec{w}) \text{ und damit}$$

$$\Delta\vec{w} = -\eta\left.\frac{\partial D}{\partial w_j}\right|_{\vec{x}^\alpha} = \eta \cdot (\vec{x}^\alpha - \tilde{\vec{x}}^\alpha(y_1^\alpha, \dots, y_q^\alpha; \vec{w})) \cdot \frac{\partial}{\partial w_j}\tilde{\vec{x}}(y_1^\alpha, \dots, y_q^\alpha; \vec{w})$$

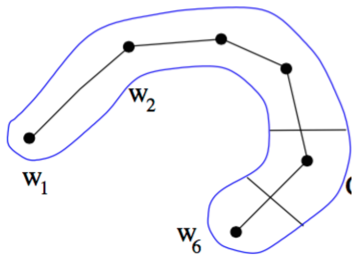
- Attention: $y_i(\vec{x})$ depend on $\vec{x}$ and $\vec{w}$! (*best-match-point*)

Example: algorithmic procedure e.g. on a discrete grid A (e.g. $q = 2$): $\tilde{\vec{x}}(y_1, y_2; \vec{w})$.

- surface parameters y_1, y_2 become discrete grid index points $r \in A$,
- function parameter $\vec{w}$ become total set of $\vec{w} = \{\vec{w}_{\vec{r}} | \vec{r} \in A\}$ of lattice point positions $\vec{w}_{\vec{r}}$ in the embedding space $\mathbb{R}^d$.

With this we can represent it in a 'lattice representation':

$$\tilde{\vec{x}}(y_1, y_2; \vec{w}) \equiv \vec{w}_{\vec{r}} \in \mathbb{R}^d$$



Mean distance between grid and data

$$D[\vec{w}] = \frac{1}{2}\sum_r \int_{F_r} (\vec{x} - \vec{w}_{\vec{r}})^2 p(\vec{x}) d\vec{x}$$

$$F_{\vec{r}} = \{\vec{x} | \, \|\vec{x} - \vec{w}_r\| \le \|\vec{x} - \vec{w}_{r'}\| \; \forall \, \vec{r}' \neq \vec{r}\}$$

- i.e. 'winner takes all'

$F_{\vec{r}}$ is called *Voronoi cell* around r.

$$D^*[\vec{w}] = \frac{1}{2}\sum_r \int_{F_r} (\vec{x} - \vec{w}_r)^2 p(\vec{x}) d\vec{x}$$

$$\frac{\partial D^*}{\partial \vec{w}_r} = -\int_{F_r} (\vec{x} - \vec{w}_r) \cdot p(\vec{x}) d\vec{x}$$

$$\Delta\vec{w}_r = -\epsilon\frac{\partial D^*}{\partial \vec{w}_r} = \epsilon\langle\vec{x} - \vec{w}_r\rangle|_{\vec{x}\in F_r}$$

Stochastic approximation:

$$\Delta\vec{w}_r \overset{def}{=} \epsilon(\vec{x}^\alpha - \vec{w}_r)$$

where r = index of the center $\vec{w}_r$ with smallest distance to the actual learning example $\vec{x}^\alpha$.

- Integration of an additional smoothness constraint / condition for the elimination of the problem of lattice folding: "`blurring"' of each adaptation step over the neighborhood of r.

$$\Delta\overrightarrow{w_{r'}} = \underbrace{\epsilon \cdot \exp\left(-\frac{(r - r')^2}{2\sigma^2}\right)}_{\text{blurring function}}(\vec{x} - \overrightarrow{w_{r'}}),$$

where r is the index of the Voronoi cell in which $\vec{x}$ falls.

- → learning rule of Kohonen nets / self-organizing networks

[ws18EOT1207]