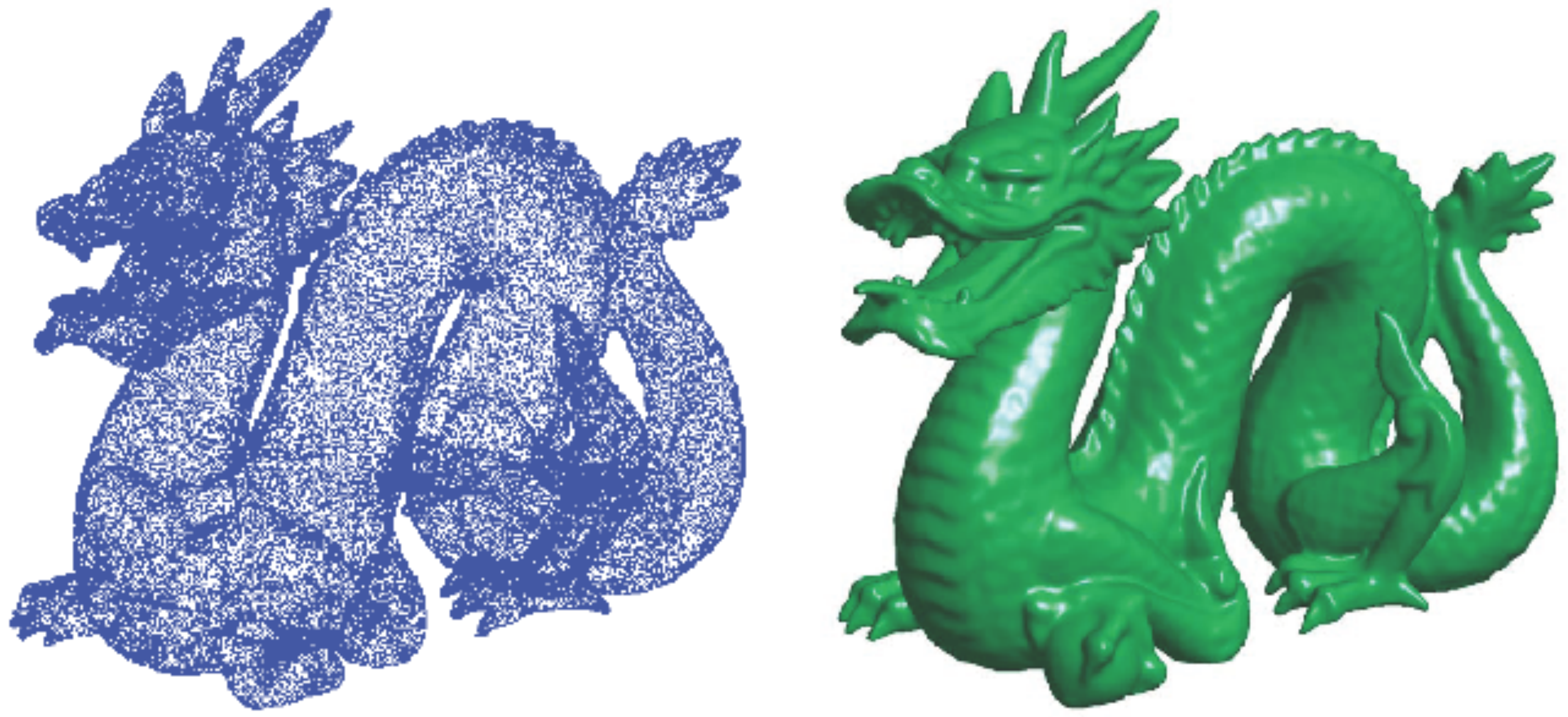
Normal Estimation

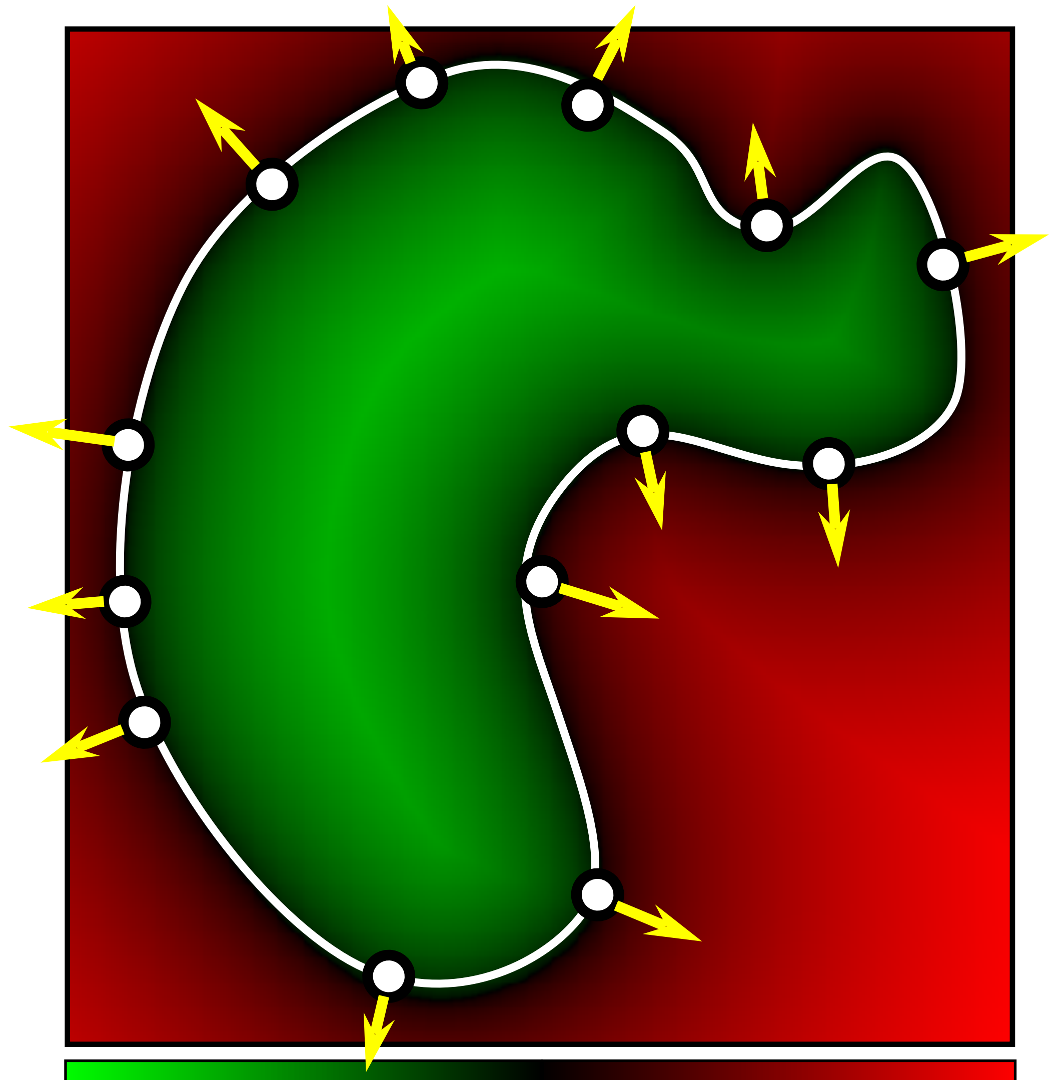
Acknowledgements: Daniele Panozzo

Implicit Surface Reconstruction

- Implicit function from point clouds



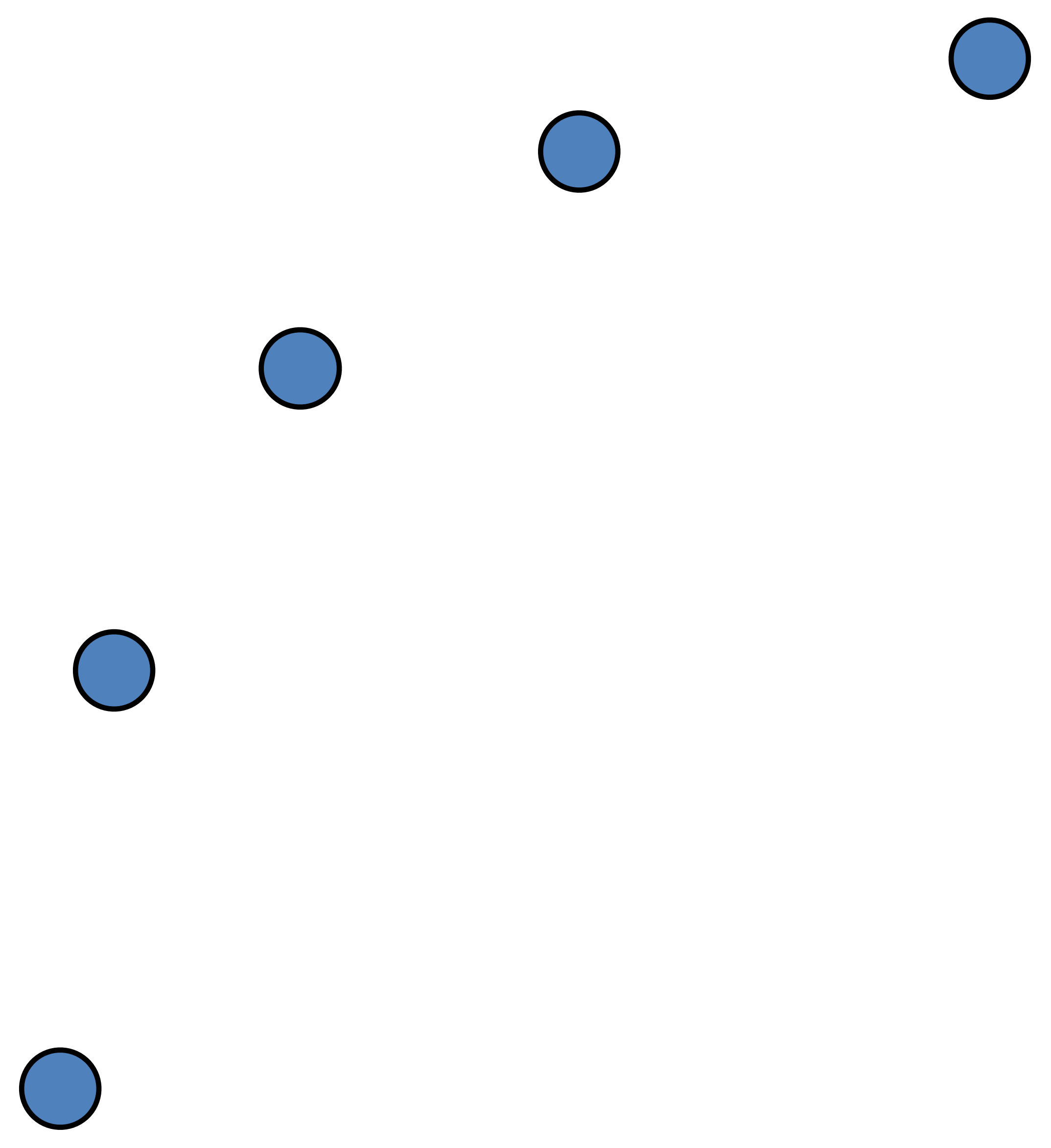
- Need consistently oriented normals



University of Victoria | Computer Science

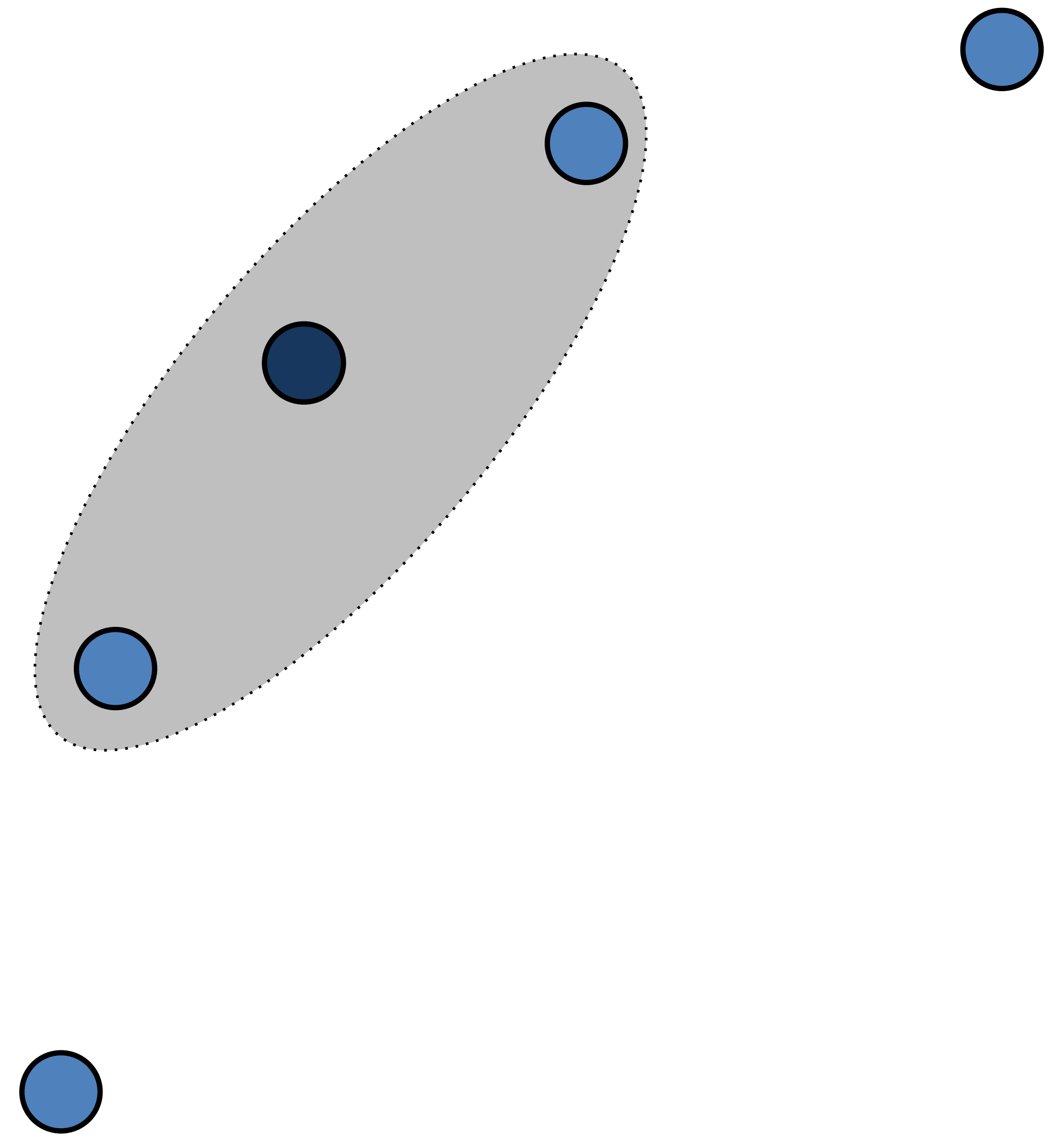
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



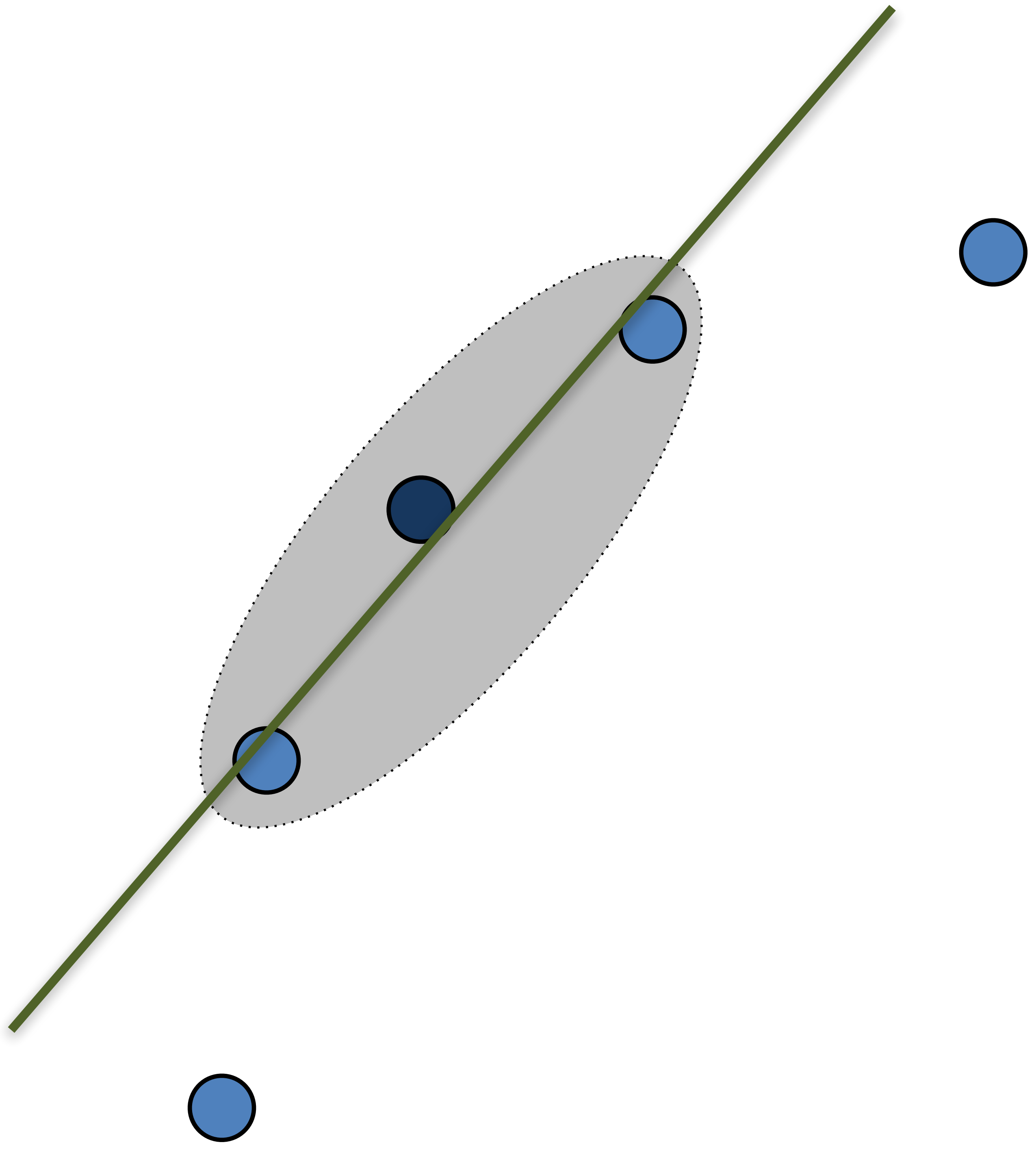
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



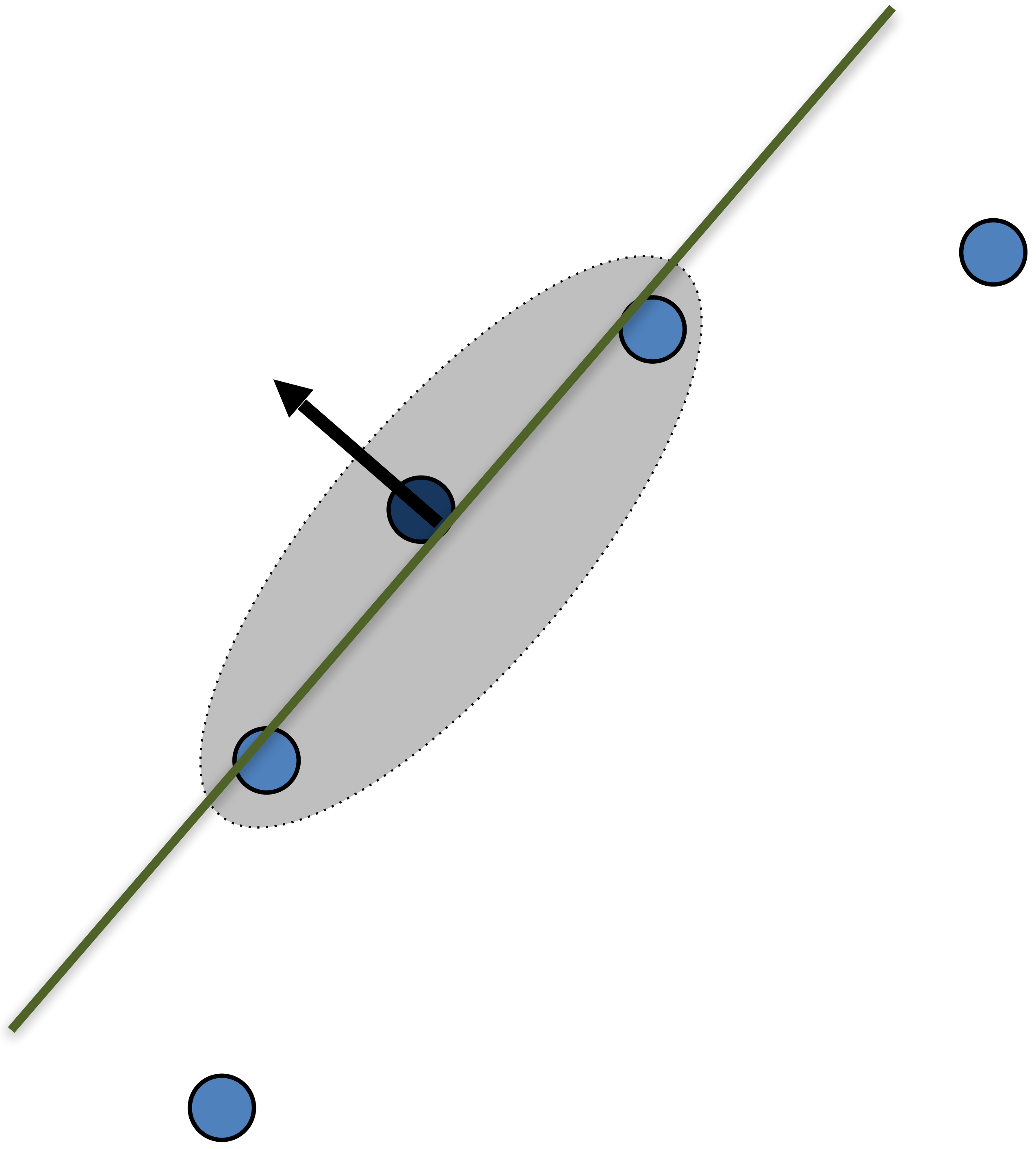
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



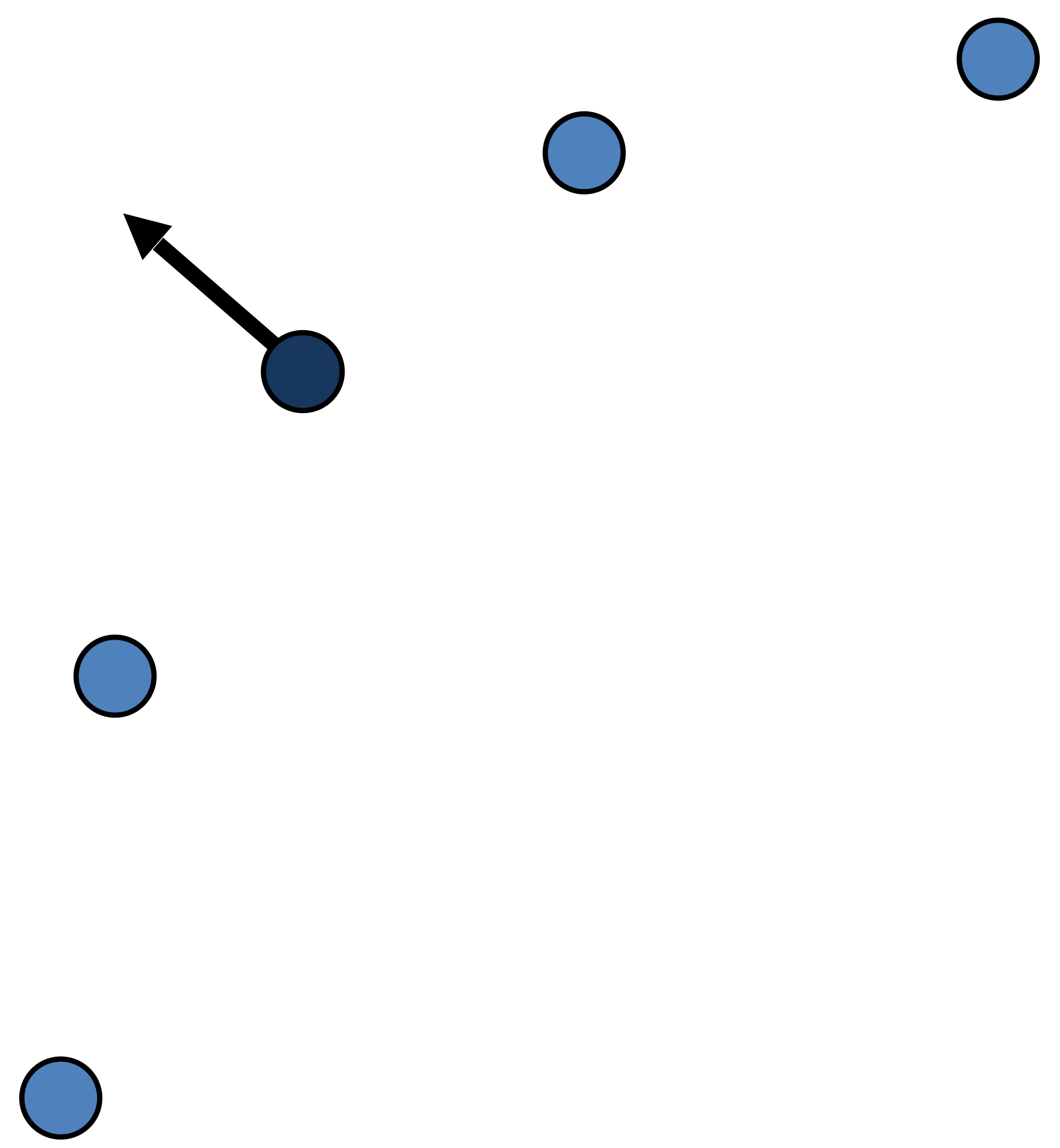
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



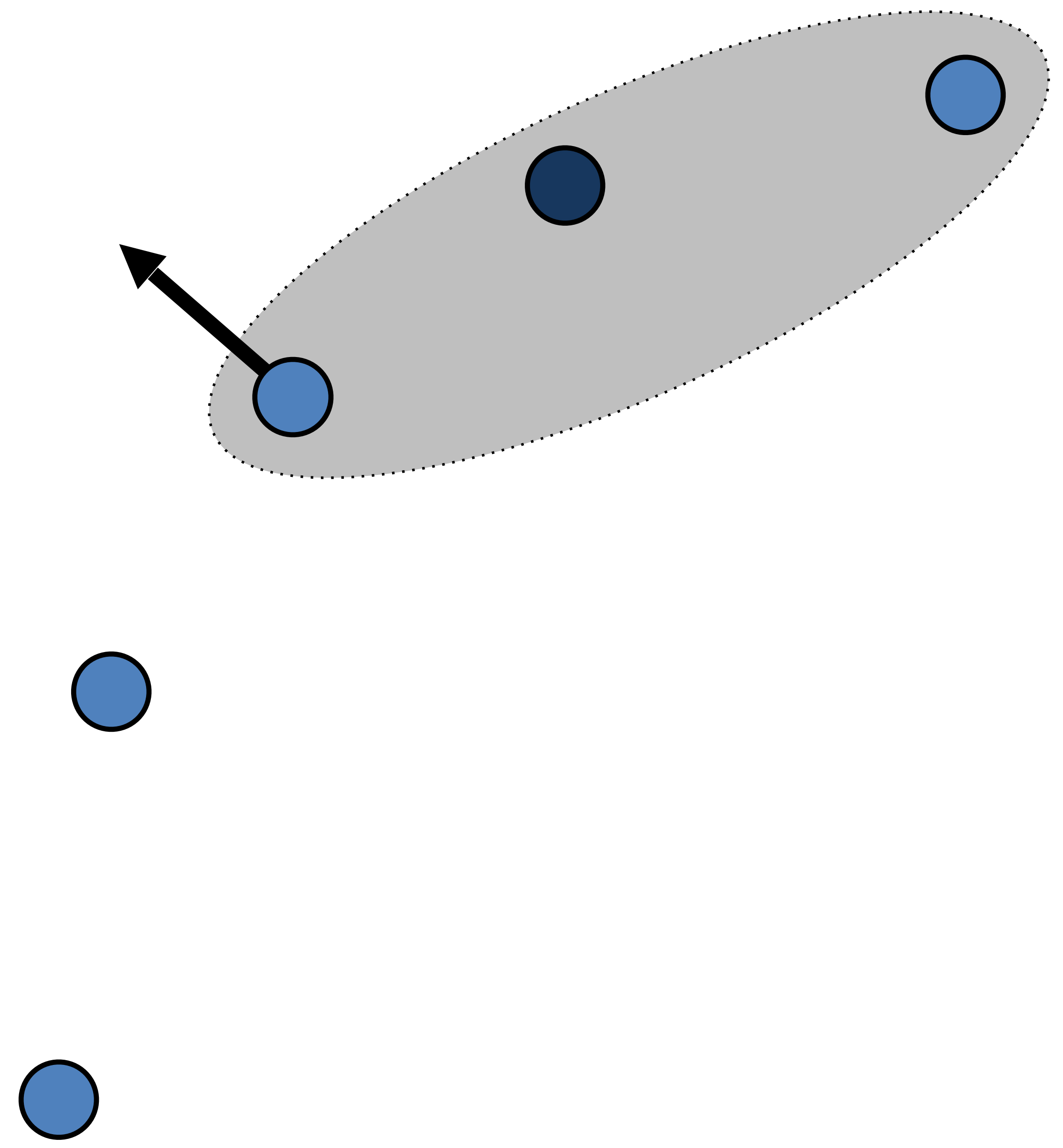
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



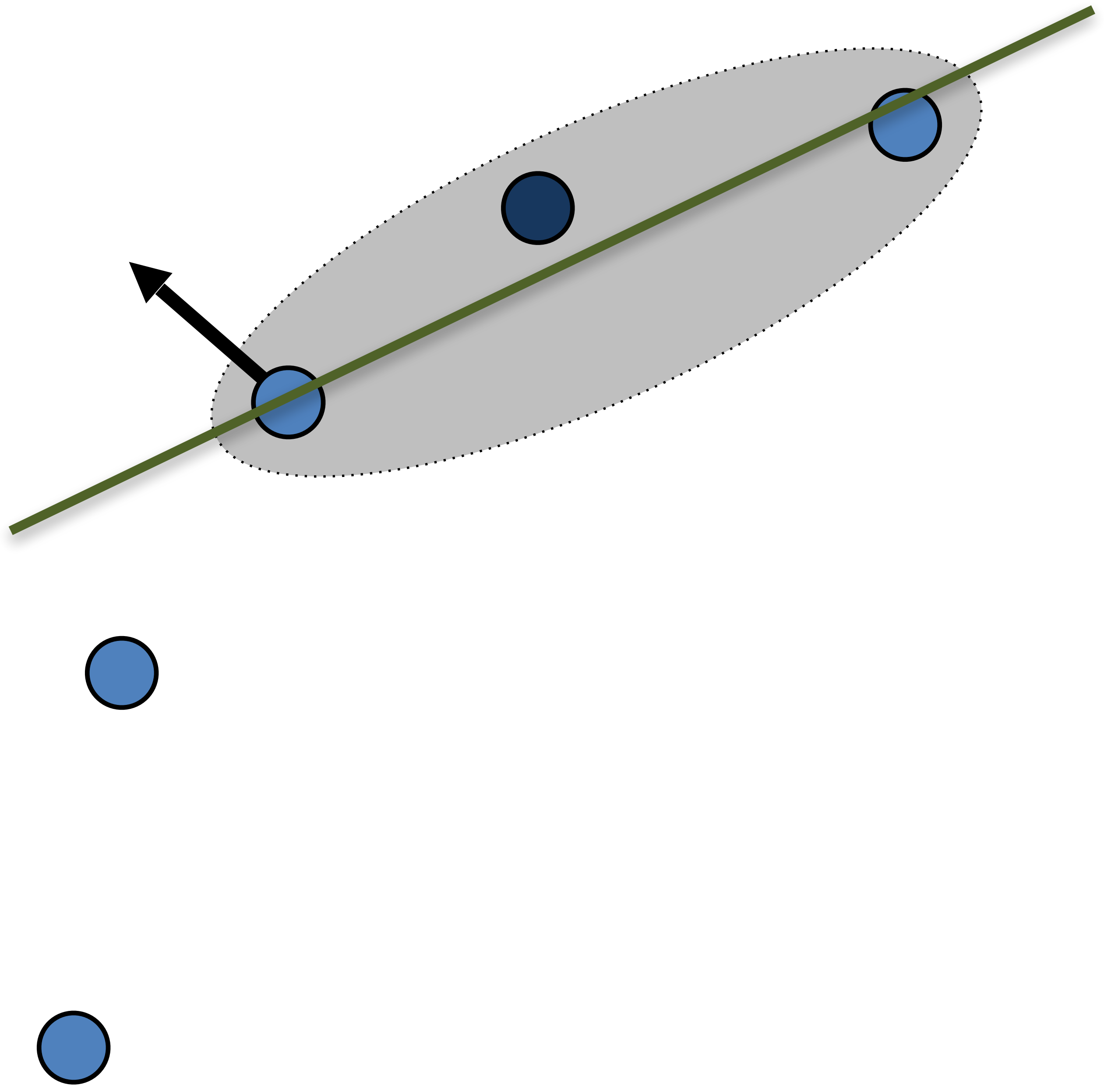
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



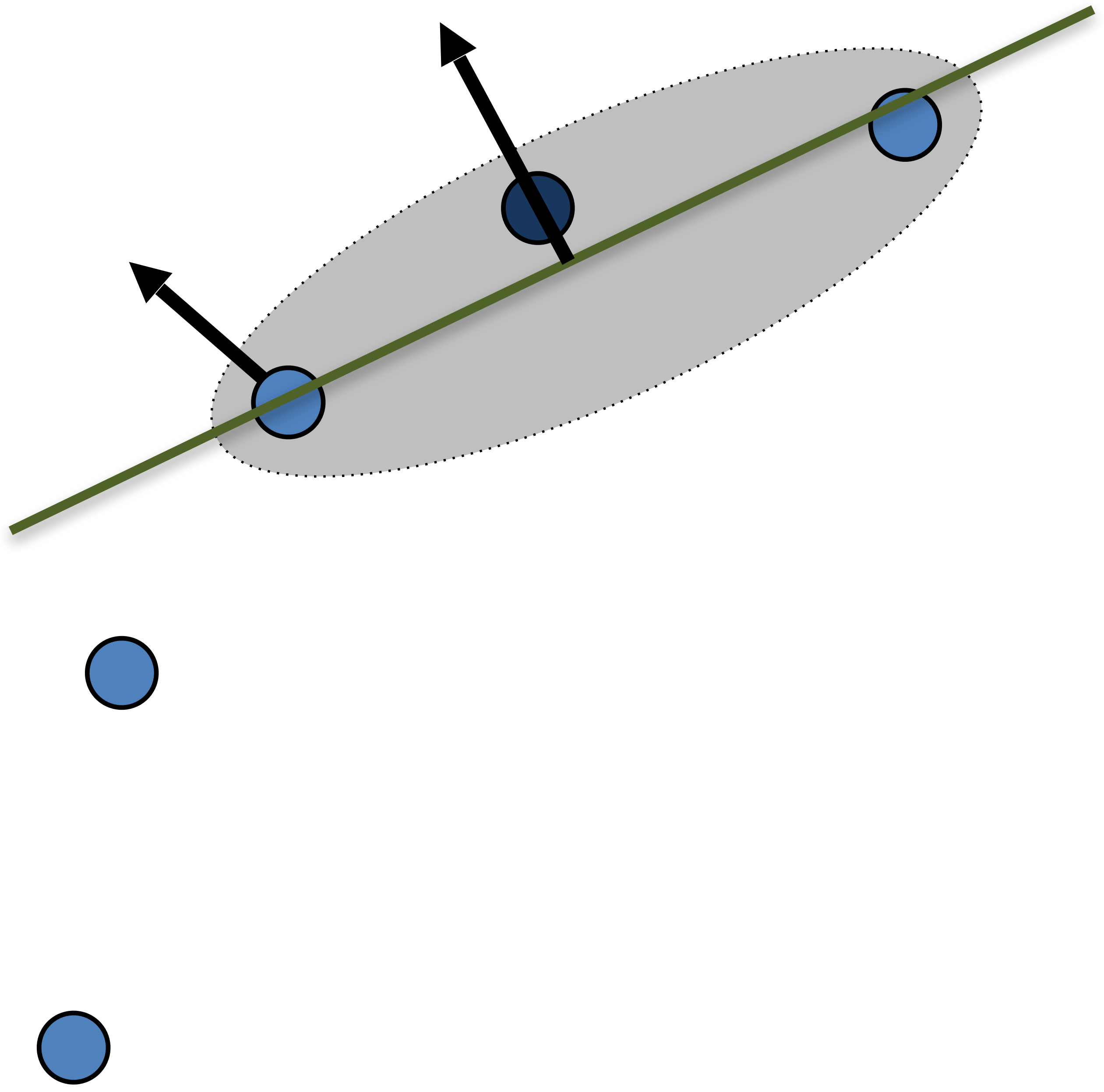
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



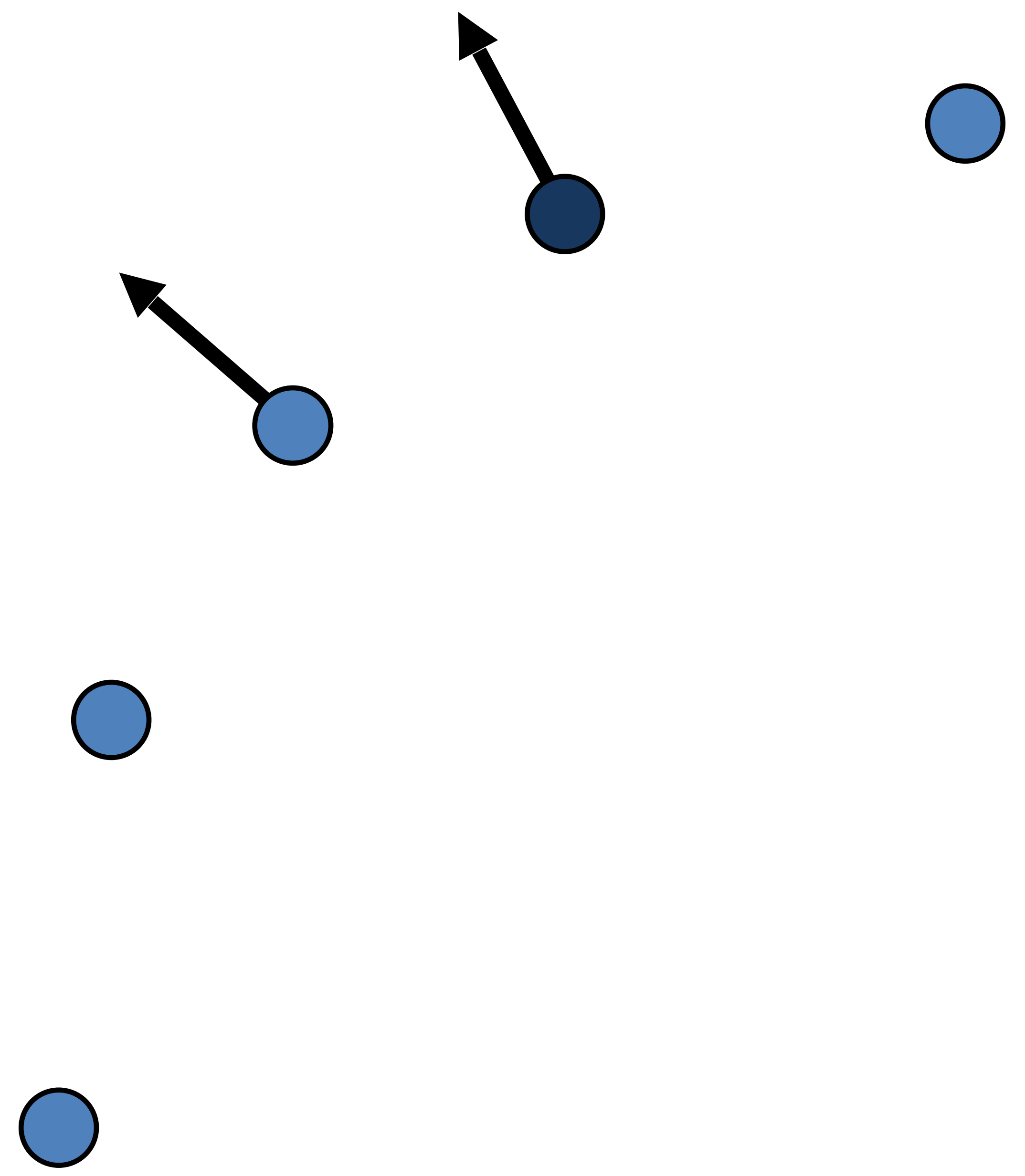
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



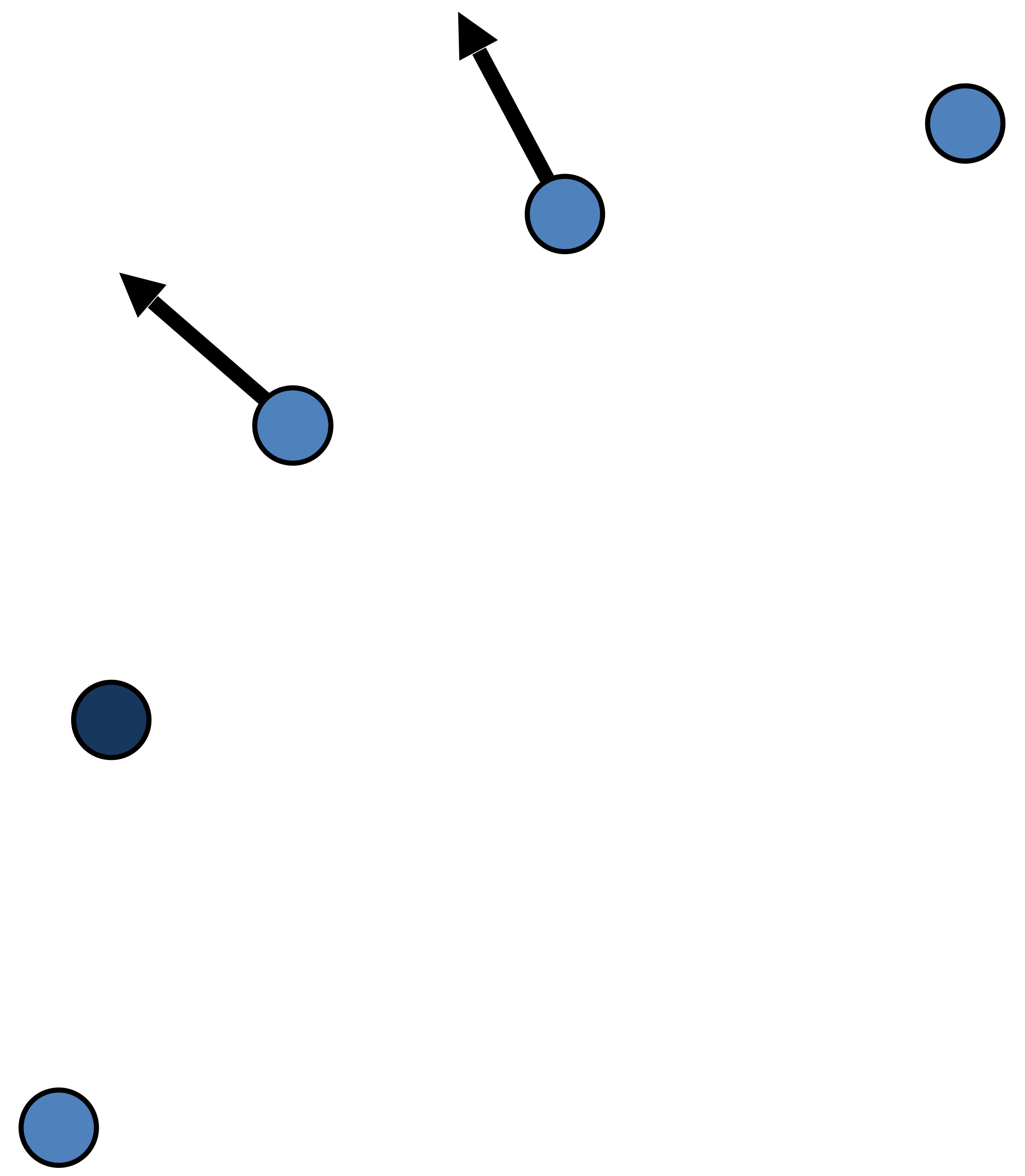
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
    - Estimate the direction by fitting a local plane
    - Find consistent global orientation by propagation (spanning tree)



Normal Estimation

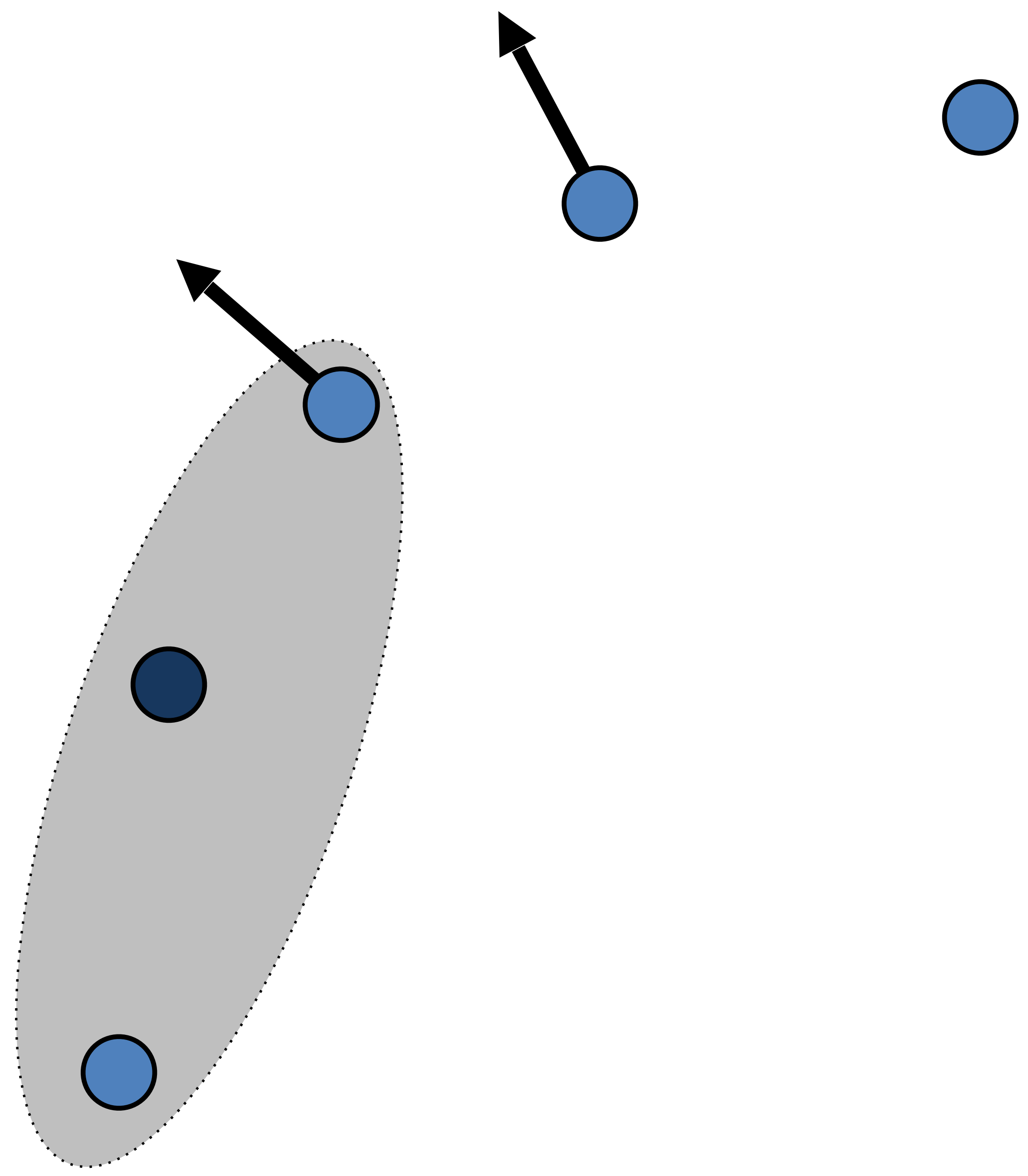
- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



University of Victoria | Computer Science

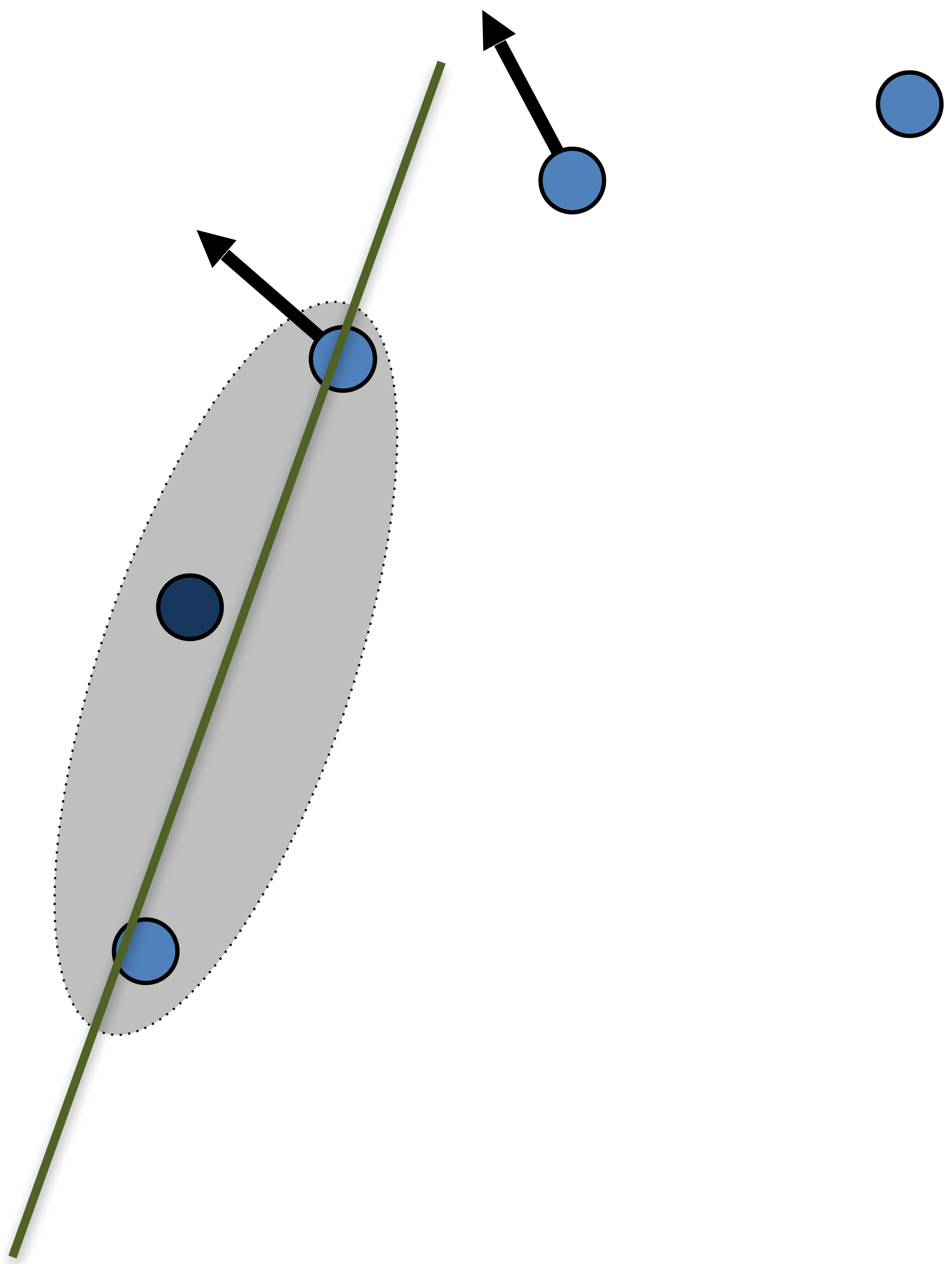
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



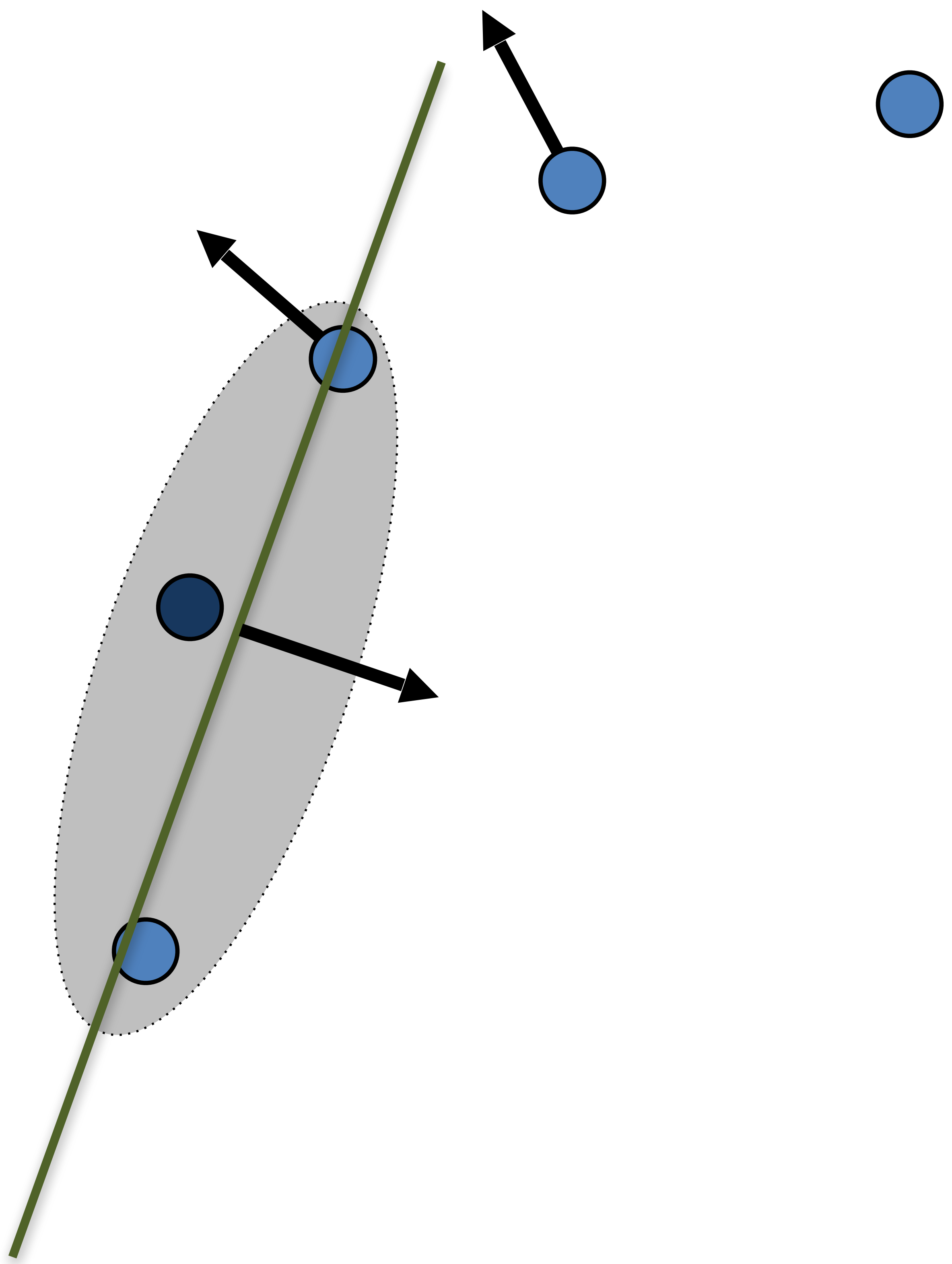
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



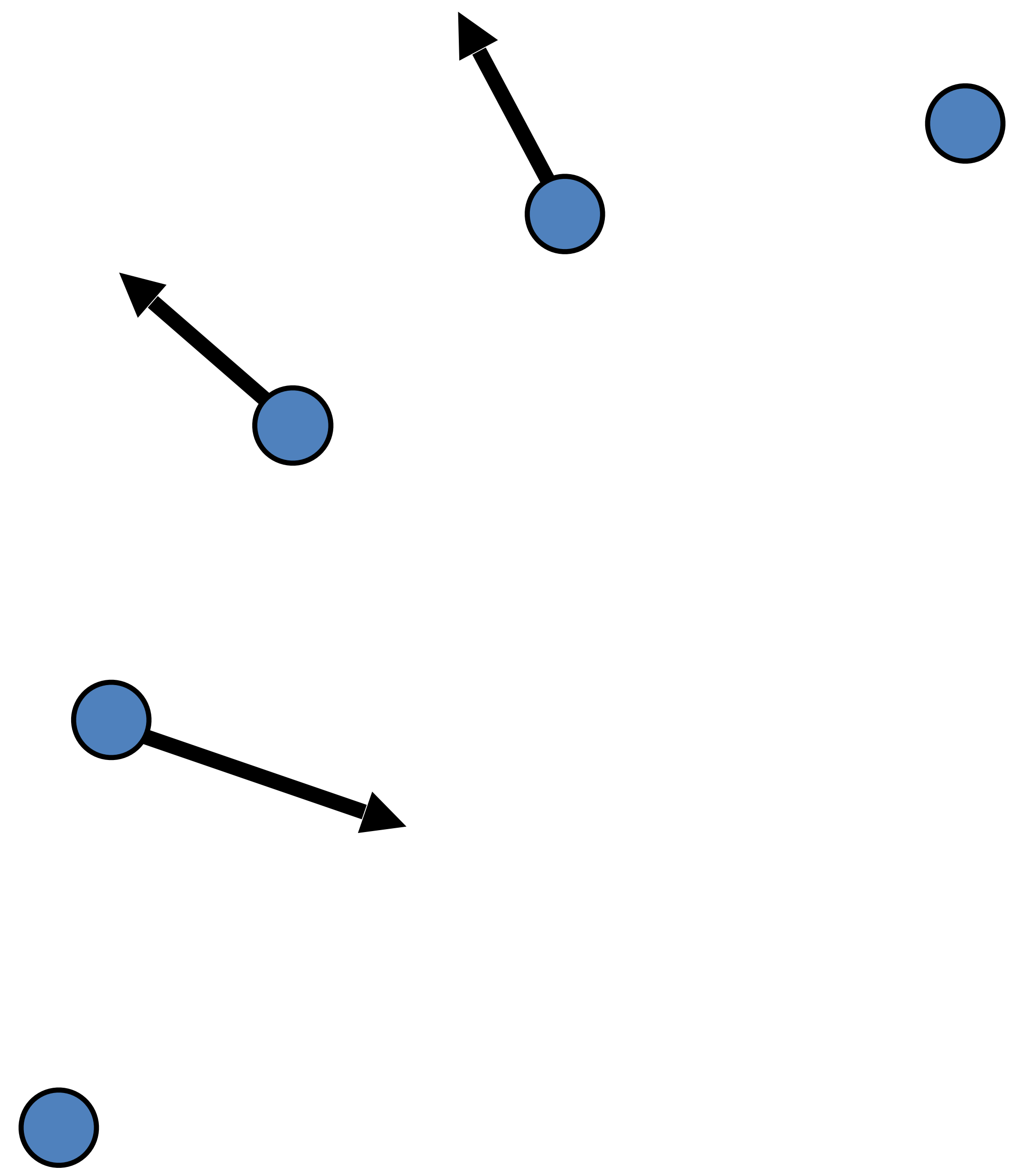
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



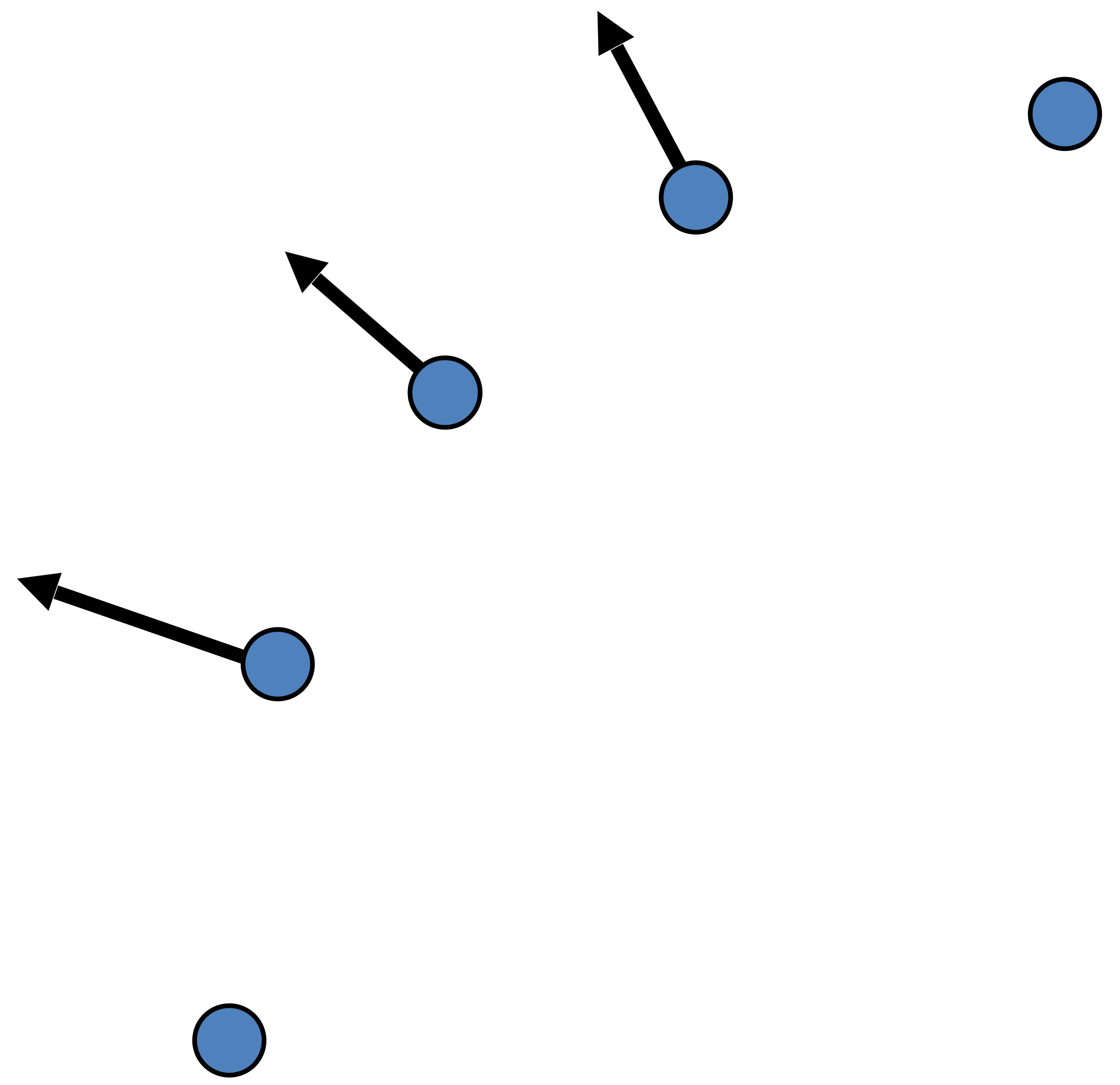
Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
    - Estimate the direction by fitting a local plane
    - Find consistent global orientation by propagation (spanning tree)



Normal Estimation

- Assign a normal vector **n** at each point cloud point **x**
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)



University of Victoria | Computer Science

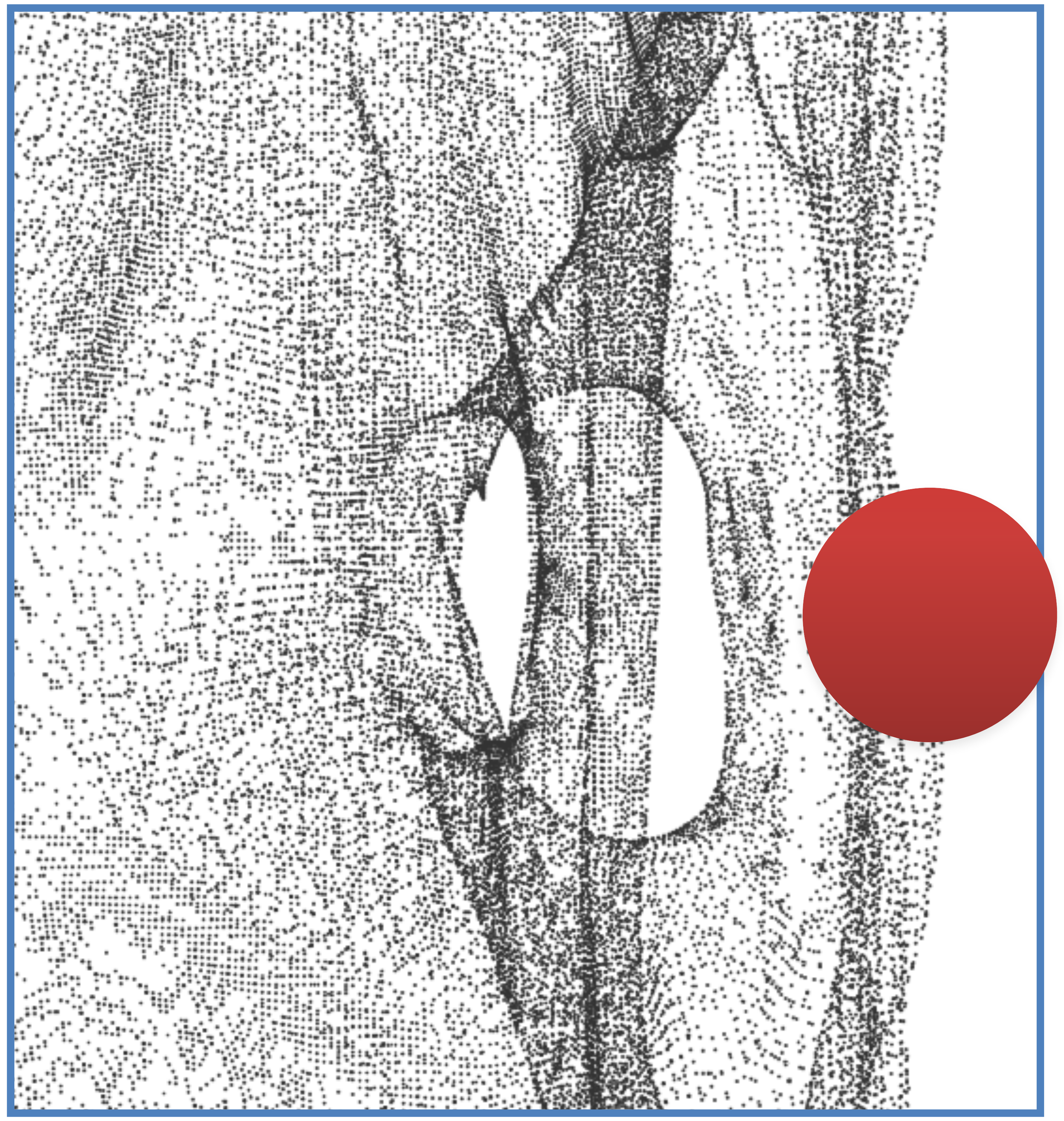
Local Plane Fitting

- For each point $\mathbf{x}$ in the cloud, pick k nearest neighbors or all points in r-ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$

- Find a plane Π that minimizes the sum of square distances:

$$\min \sum_{i=1}^{n} \operatorname{dist}(\mathbf{x}_i, \Pi)^2$$



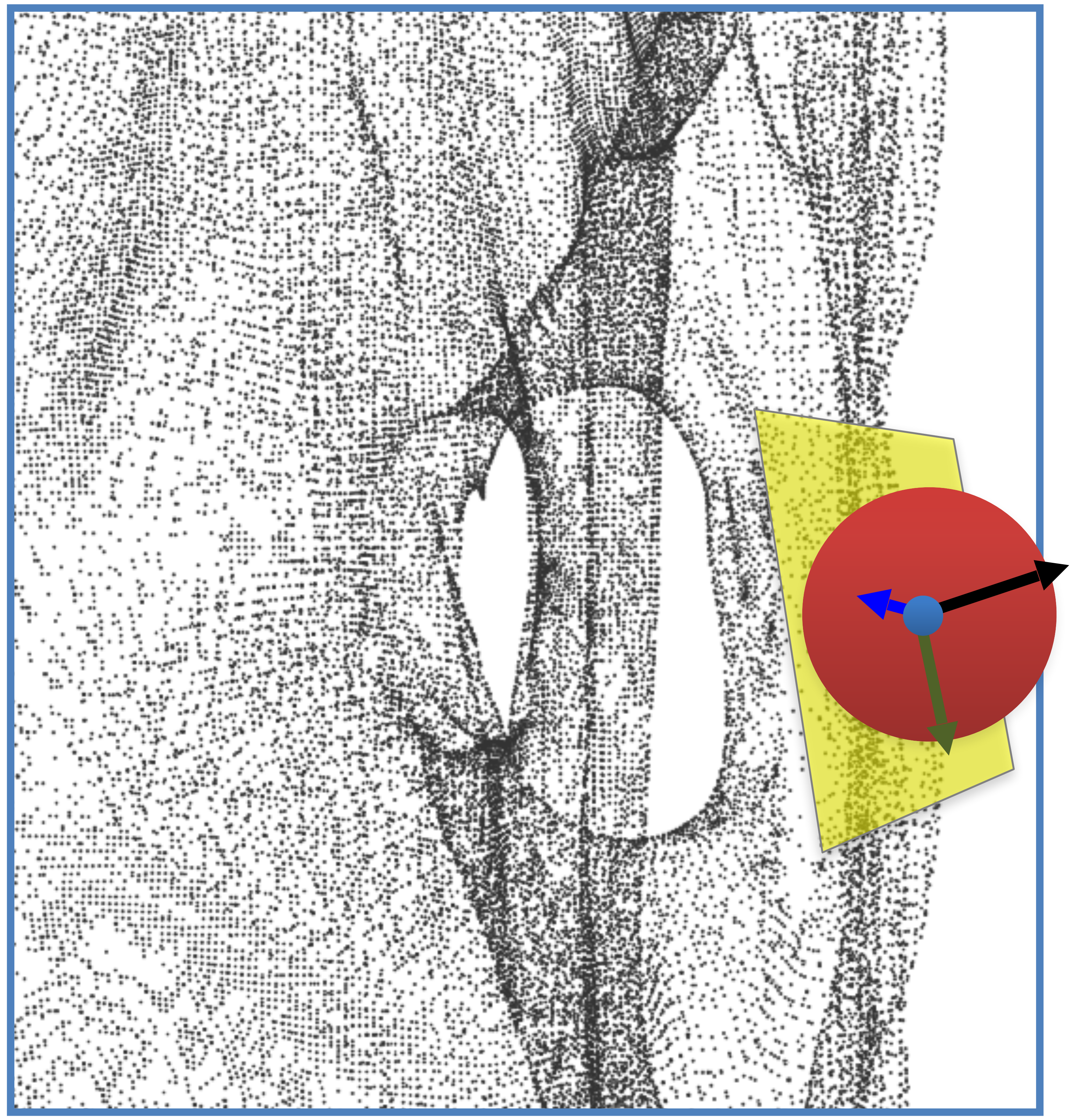
Local Plane Fitting

- For each point **x** in the cloud, pick *k* nearest neighbors or all points in *r*-ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

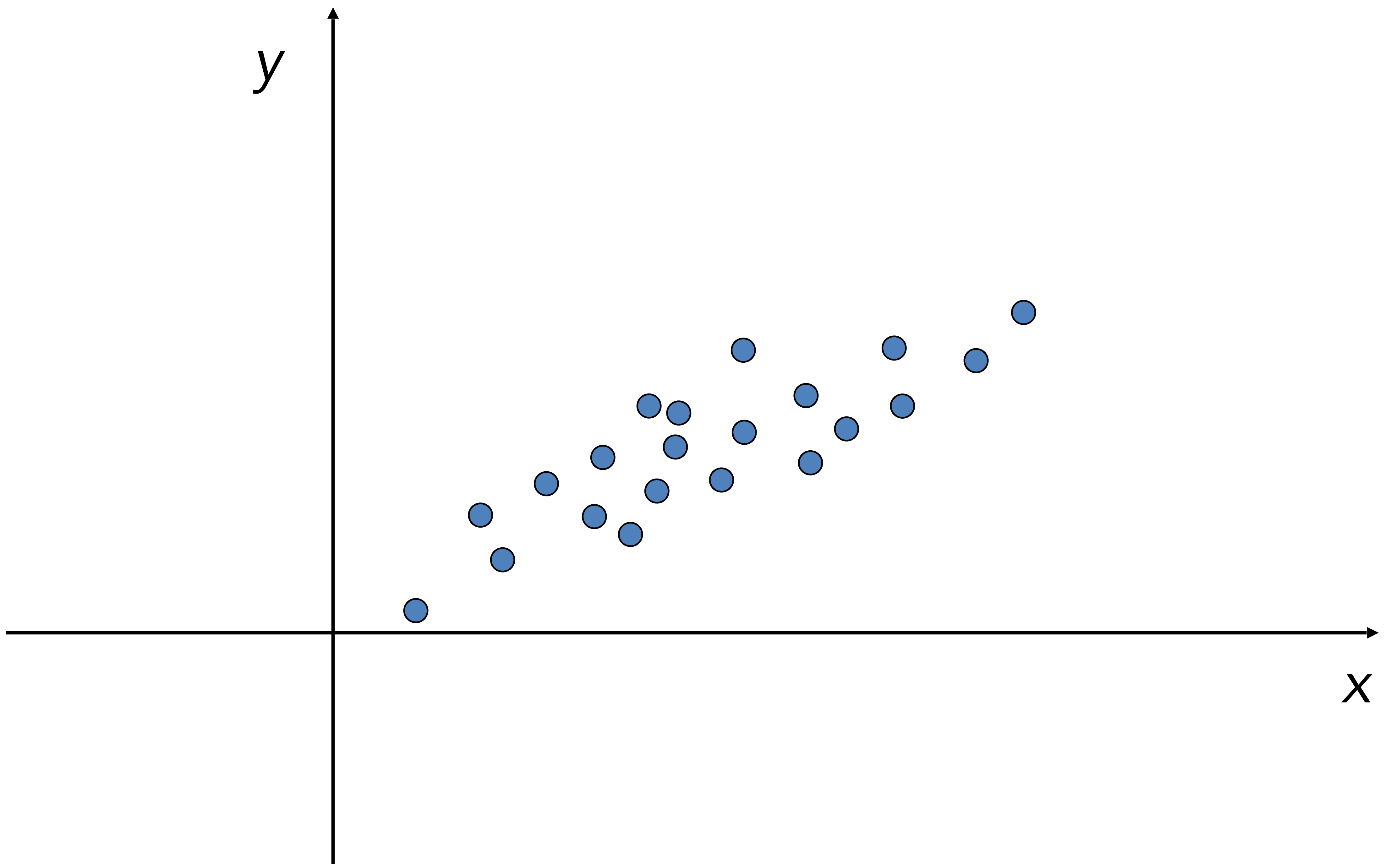
$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$

- Find a plane **Π** that minimizes the sum of square distances:

$$\min \sum_{i=1}^{n} \operatorname{dist}(\mathbf{x}_i, \Pi)^2$$



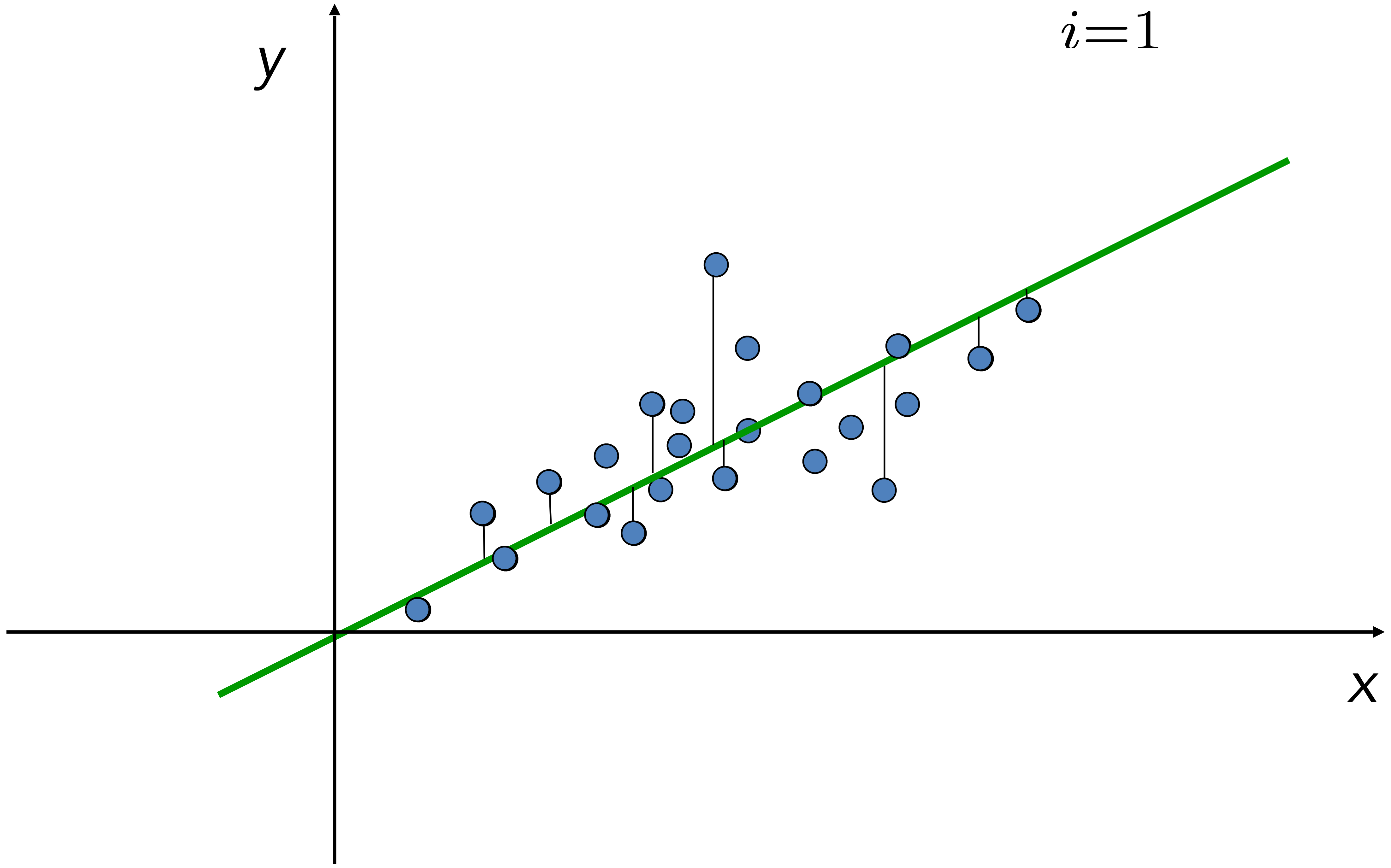
Linear Least Squares?



Linear Least Squares?

- Find a line *y* = *ax*+*b* s.t.

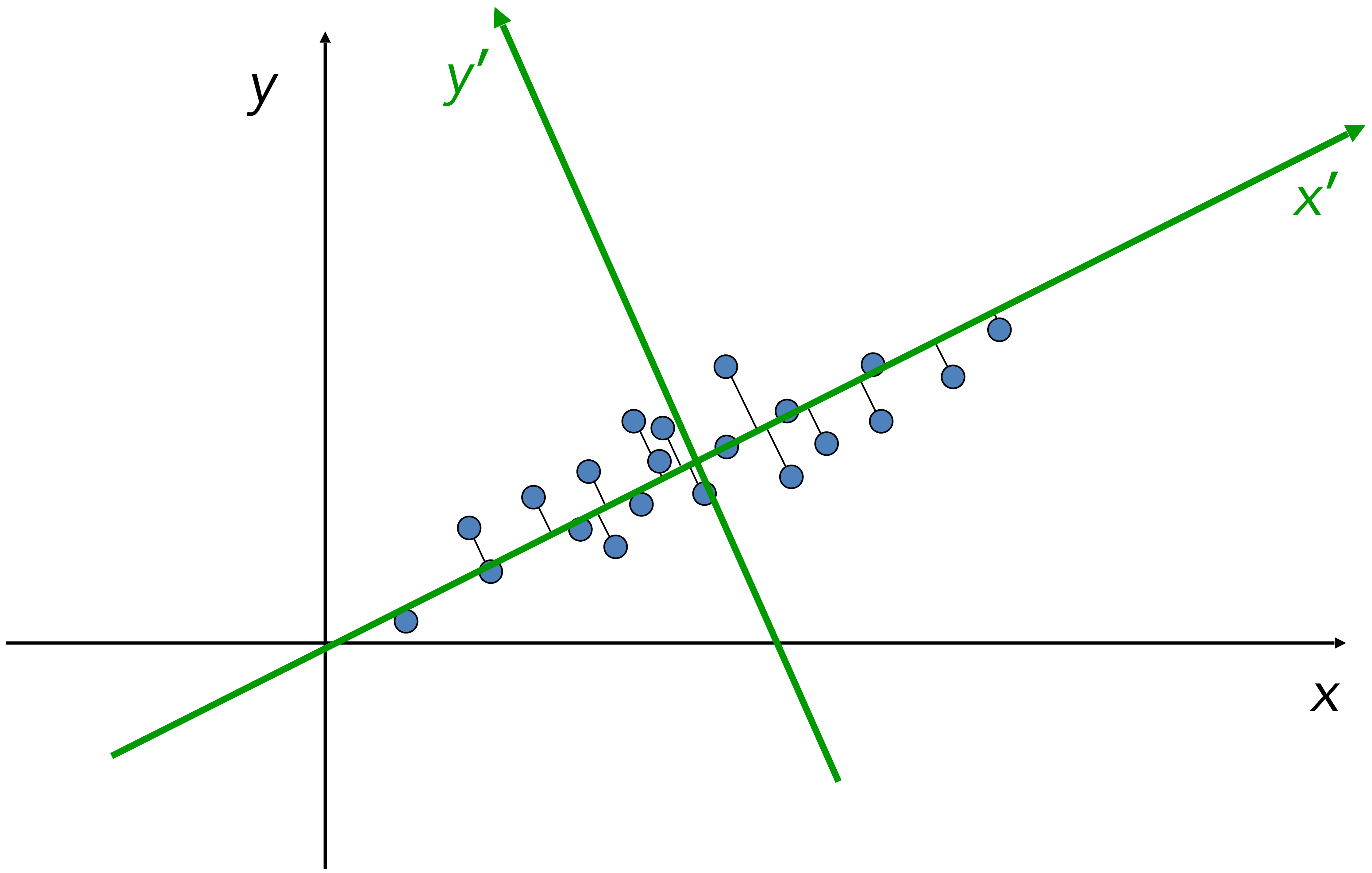
$$\min \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$



- But we would like true orthogonal distances!

University of Victoria | Computer Science

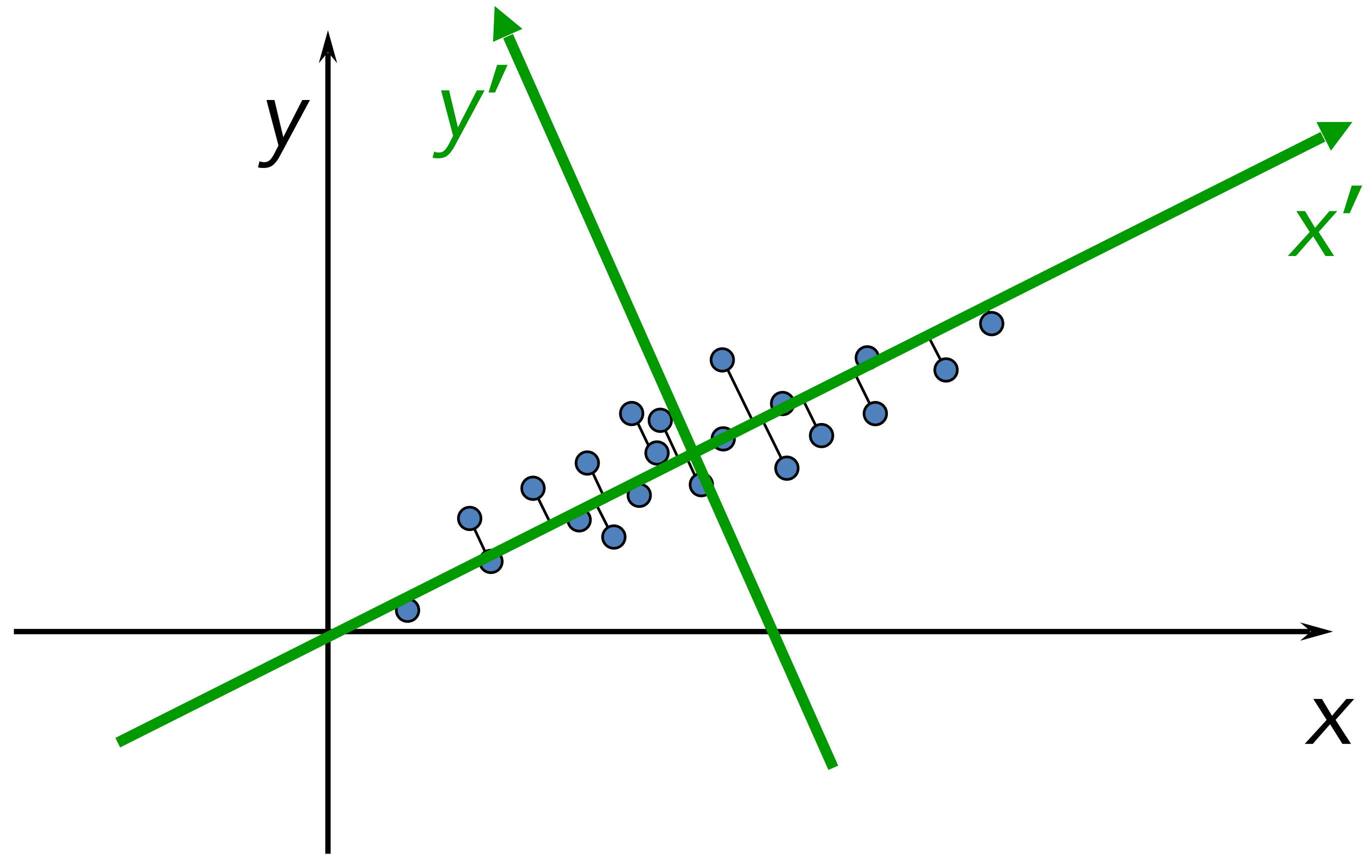
Best Fit with SSD

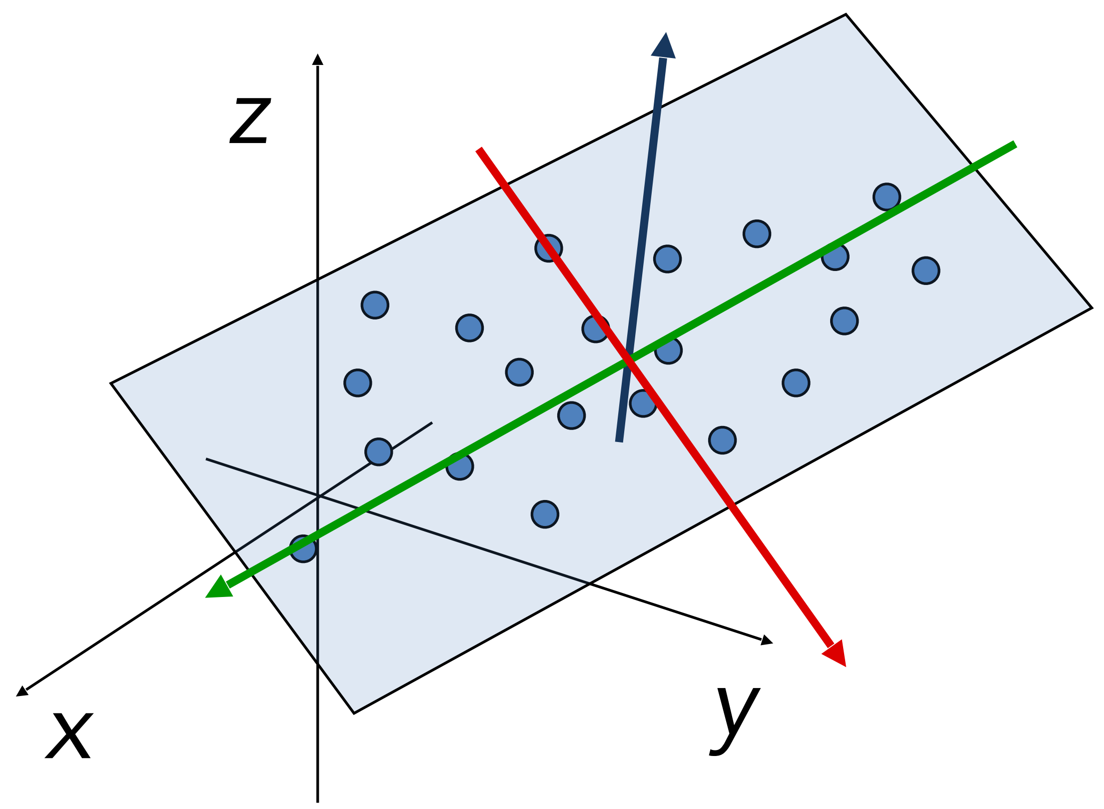


SSD = sum of squared distances (or differences)

Principle Component Analysis (PCA)

- PCA finds an orthogonal basis that best represents a given data set





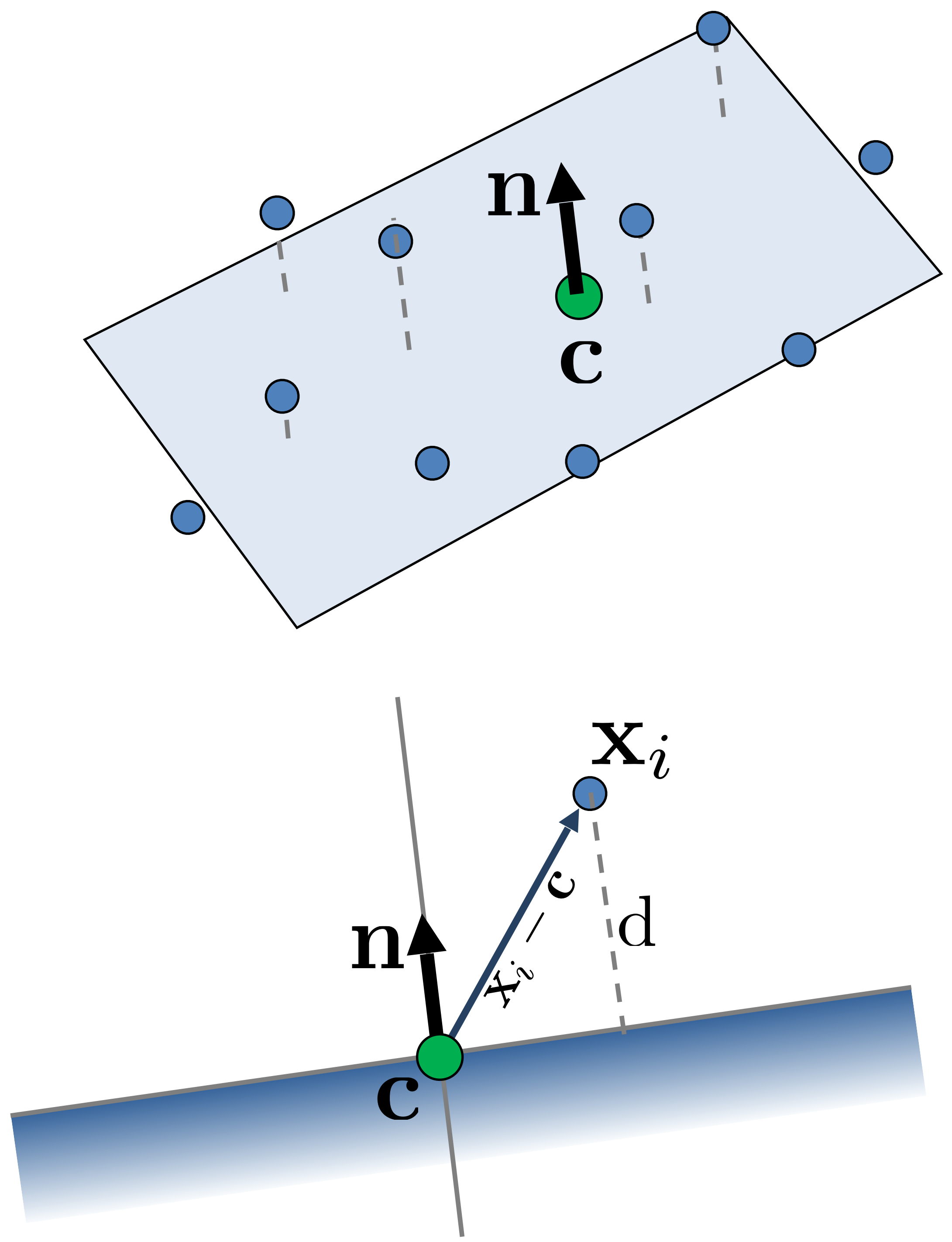
- PCA finds the best approximating line/plane/orientation… (in terms of $\Sigma_{distances^2}$)

Notations

- Input points: $$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

- Looking for a (hyper) plane passing through **c** with normal **n** s.t.

$$\min_{\mathbf{c}, \mathbf{n}, \|\mathbf{n}\|=1} \sum_{i=1}^{n} \left((\mathbf{x}_i - \mathbf{c})^T \mathbf{n} \right)^2$$



University of Victoria | Computer Science

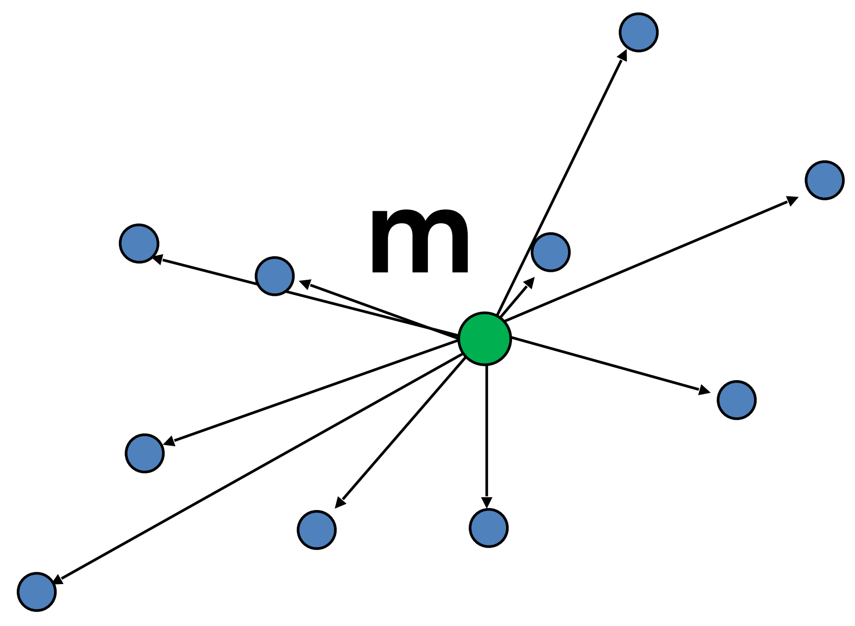
Notations

- Input points:

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

- Centroid:

$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$



- Vectors from the centroid:

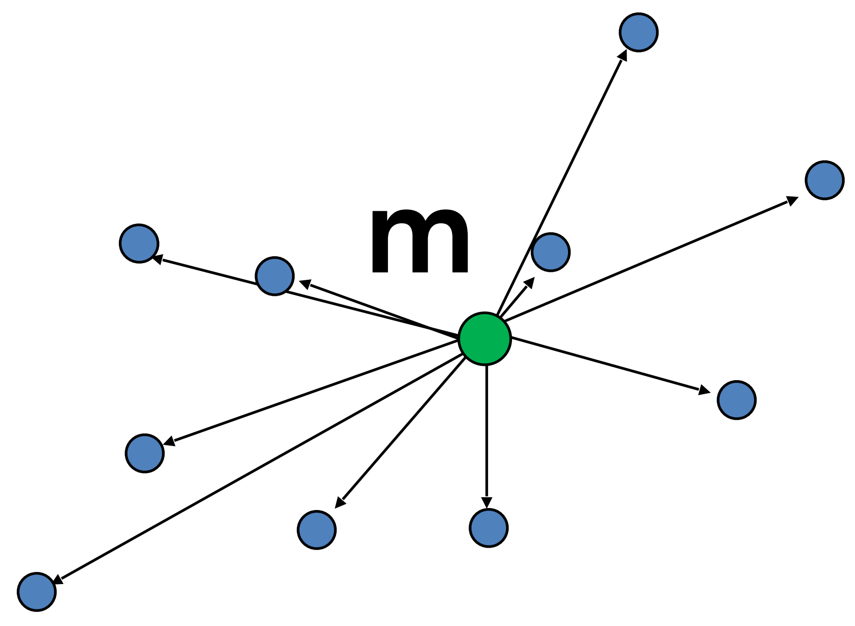
$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

Centroid: 0-dim Approximation

- It can be shown that:

$$\mathbf{m} = \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{i=1}^{n} \left((\mathbf{x}_i - \mathbf{c})^T \mathbf{n}\right)^2$$

$$\mathbf{m} = \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{c}\|^2$$



$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

- **m** minimizes SSD
- **m** will be the origin of the (hyper)-plane
- Our problem becomes:

$$\min_{\|\mathbf{n}\|=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2$$

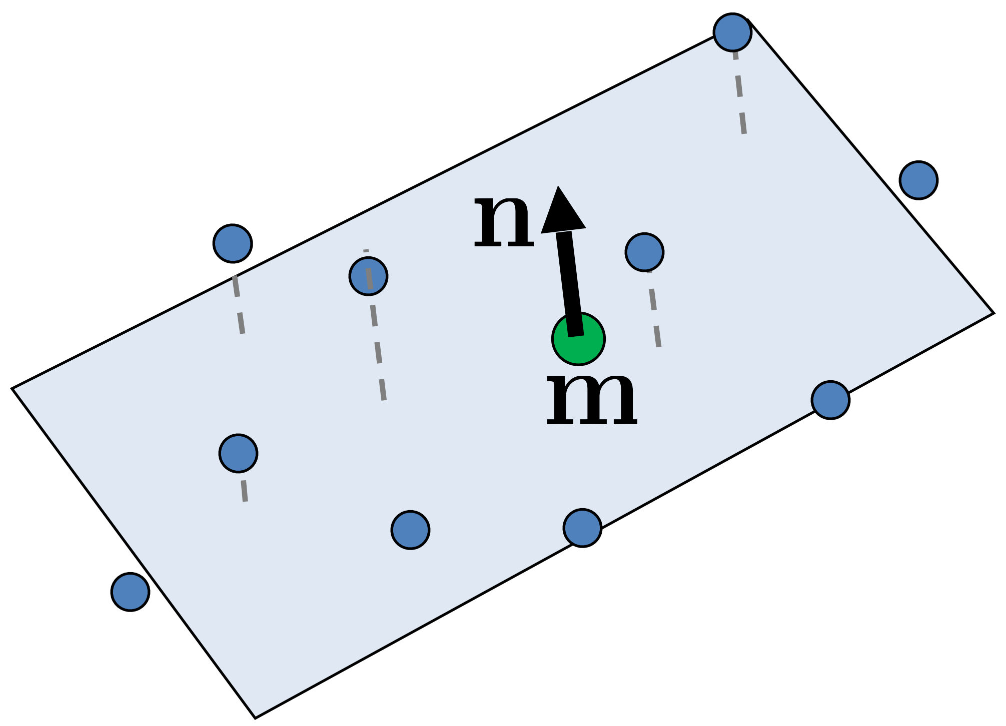
Hyperplane Normal

- Minimize!

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T\mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T\mathbf{y}_i\mathbf{y}_i^T\mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i\mathbf{y}_i^T\right) \mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right) \mathbf{n}$$

$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$



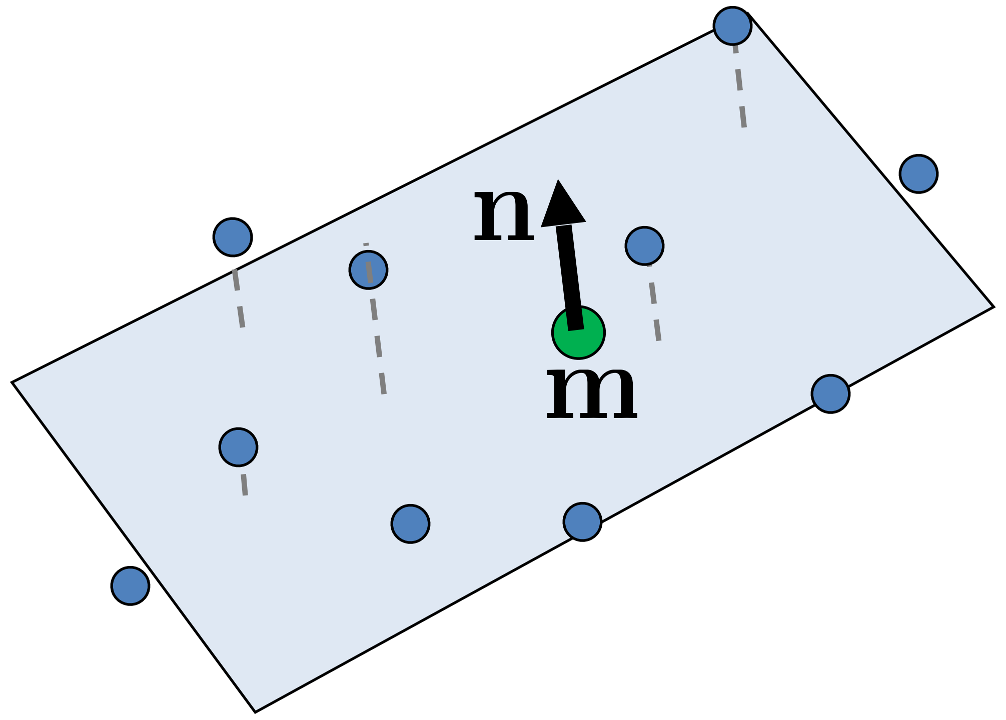
Hyperplane Normal

- Minimize!

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T\mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T\mathbf{y}_i\mathbf{y}_i^T\mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i\mathbf{y}_i^T\right)\mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right)\mathbf{n}$$

$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$



$$f(\mathbf{n}) = \mathbf{n}^T\mathbf{S}\mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$

$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$

Hyperplane Normal

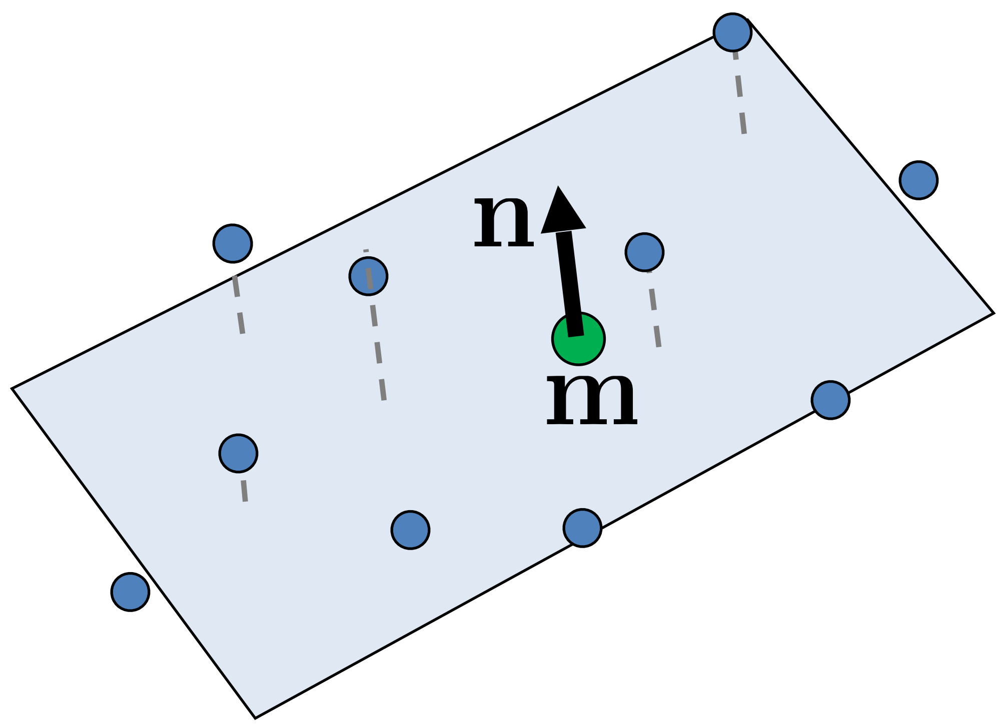
- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T\mathbf{S}\mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T\mathbf{n} - 1)$$
$$\nabla\mathcal{L} = 0$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{n}} = \frac{\partial}{\partial\mathbf{n}}f(\mathbf{n}) - \lambda\frac{\partial}{\partial\mathbf{n}}(\mathbf{n}^T\mathbf{n} - 1)$$
$$\frac{\partial\mathcal{L}}{\partial\lambda} = \mathbf{n}^T\mathbf{n} - 1$$

$$\frac{\partial}{\partial\mathbf{n}}f(\mathbf{n}) - \lambda\frac{\partial}{\partial\mathbf{n}}(\mathbf{n}^T\mathbf{n} - 1) = (\mathbf{S} + \mathbf{S}^T)\mathbf{n} - \lambda(\mathbf{I} + \mathbf{I}^T)\mathbf{n} = 2\mathbf{S}\mathbf{n} - 2\lambda\mathbf{n}$$



Matrix Cookbook!

https://archive.org/det

University of Victoria | Computer Science

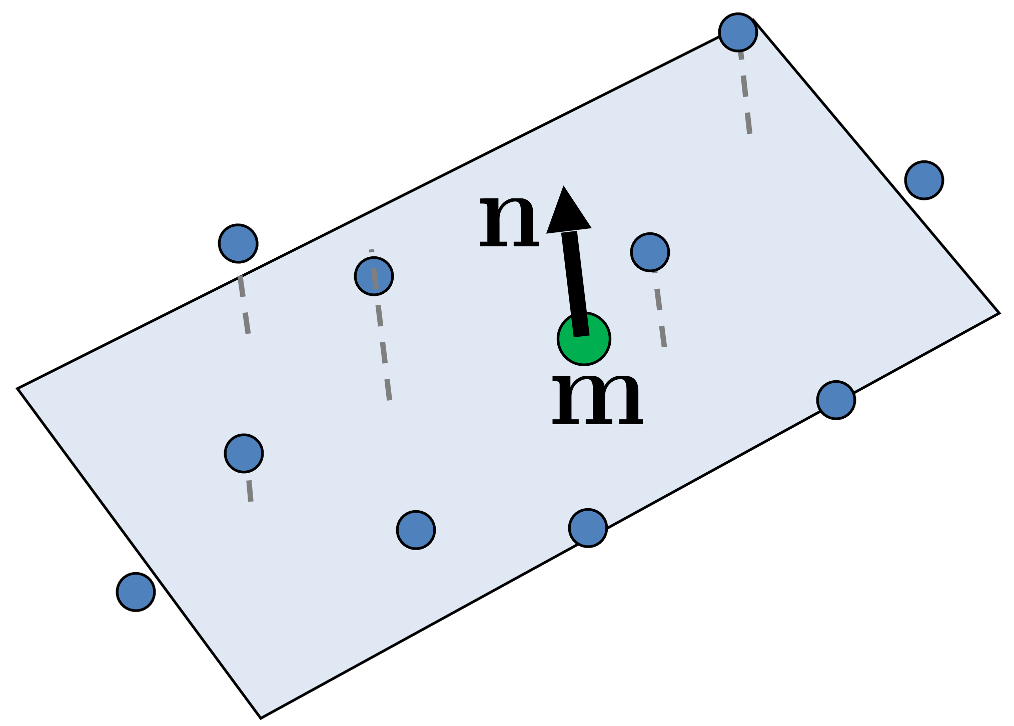
Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T\mathbf{S}\mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T\mathbf{n} - 1)$$
$$\nabla\mathcal{L} = 0$$

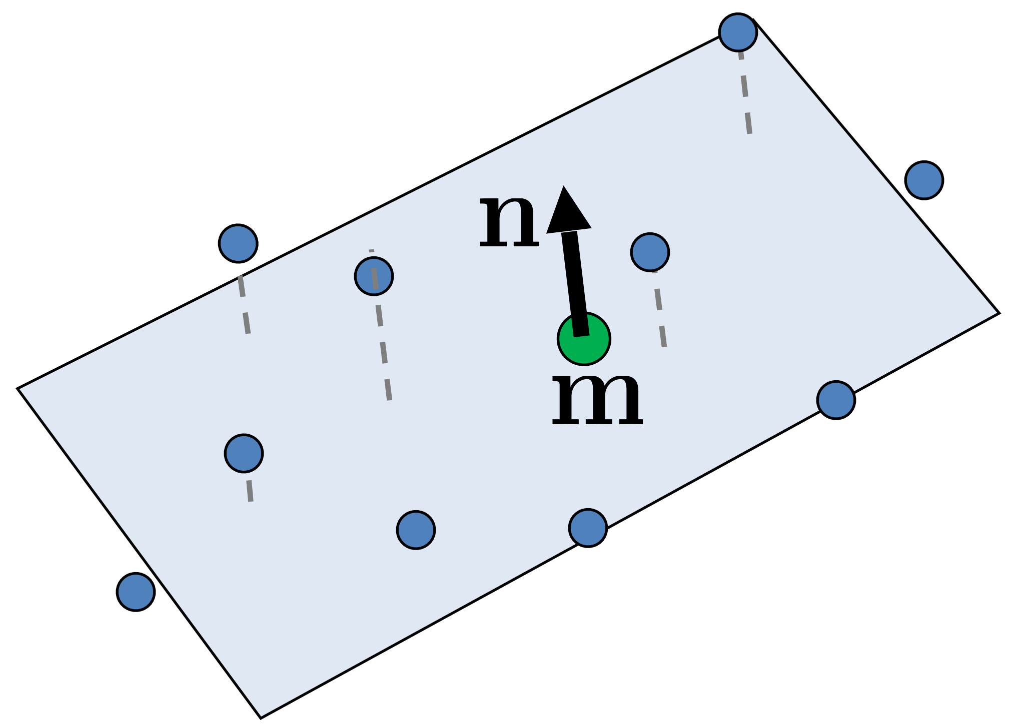
$$\frac{\partial\mathcal{L}}{\partial\mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda\mathbf{n}$$
$$\frac{\partial\mathcal{L}}{\partial\lambda} = 0 \iff \mathbf{n}^T\mathbf{n} = 1$$



Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T \mathbf{n} = 1$$



$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$
$$\nabla \mathcal{L} = 0$$

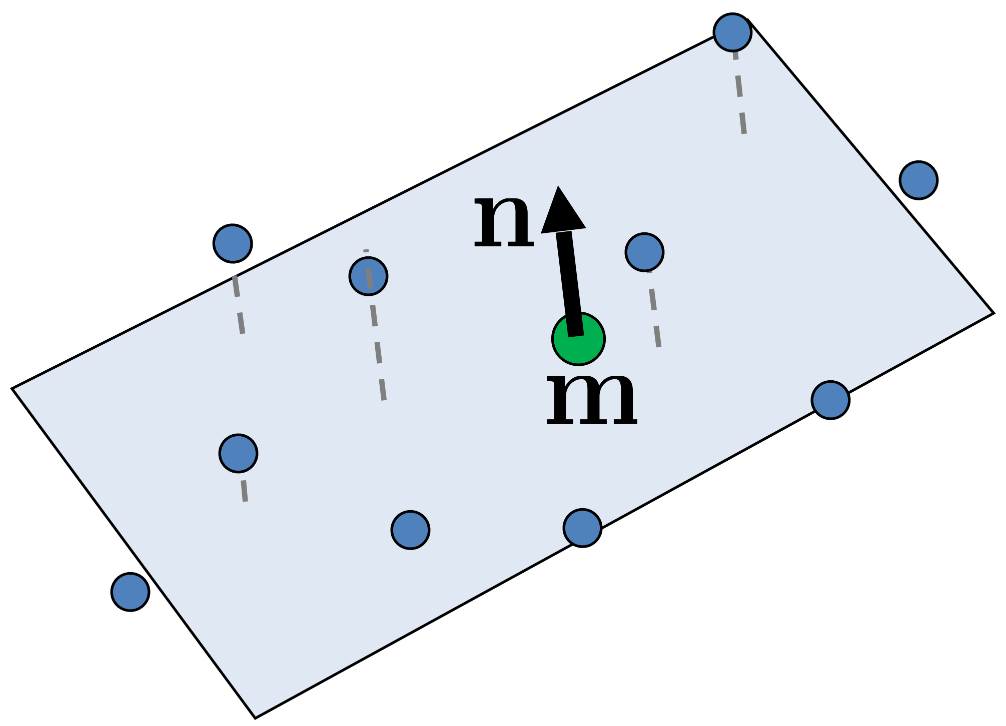
$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda \mathbf{n}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$

What can be said about **n** ??

Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T \mathbf{n} = 1$$



$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$
$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda \mathbf{n}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$

n is the eigenvector of **S** with the smallest eigenvalue

Summary – Best Fitting Plane Recipe

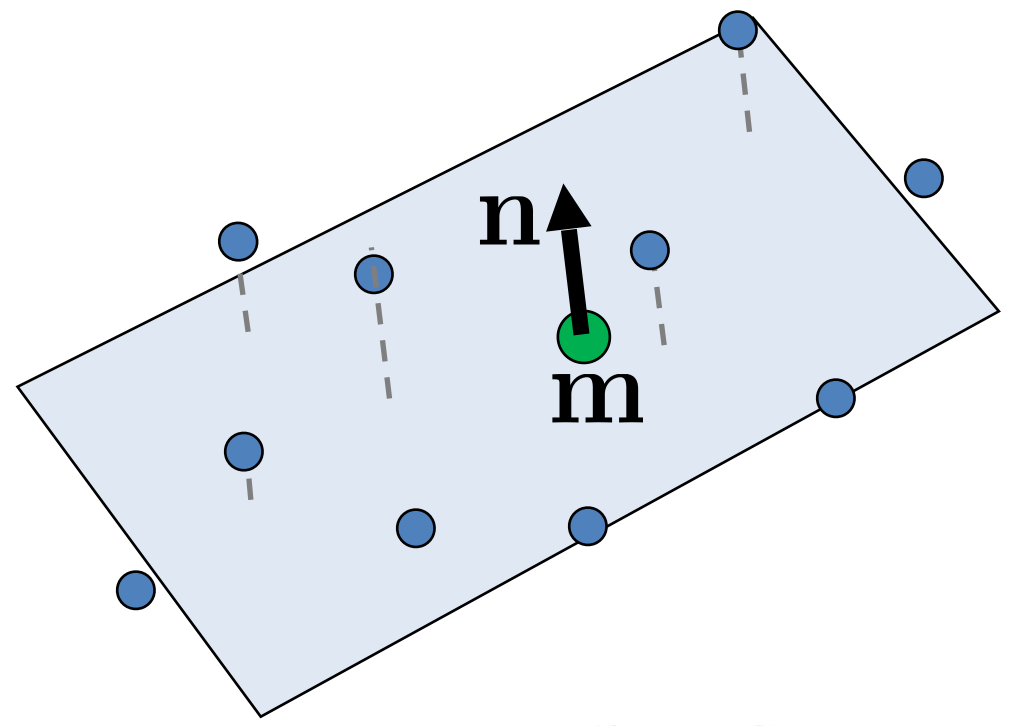
- Input: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- Compute centroid = plane origin $\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$
- Compute scatter matrix $\mathbf{S} = \mathbf{Y}\mathbf{Y}^T$

$$\mathbf{Y} = (\mathbf{y}_1 \ \mathbf{y}_2 \ \ldots \ \mathbf{y}_n)$$

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

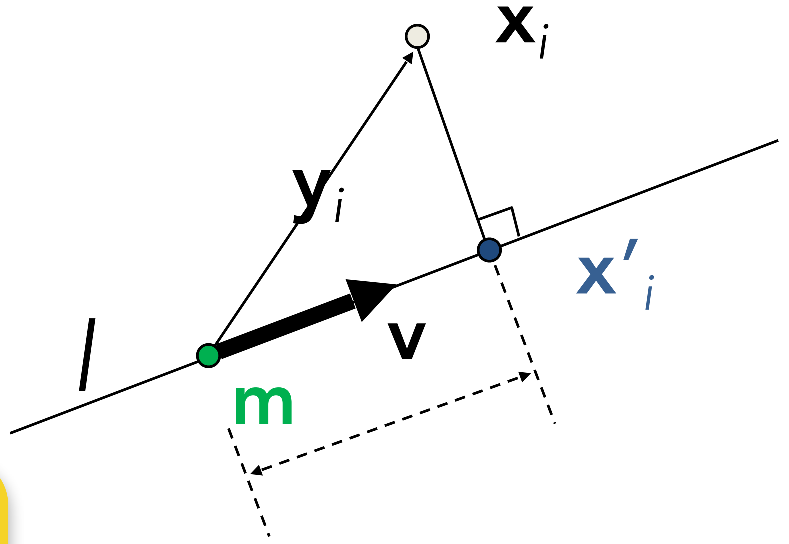
- The plane normal **n** is the eigenvector of **S** with the smallest eigenvalue

$$\mathbf{S} = \mathbf{V}\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix}\mathbf{V}^T$$



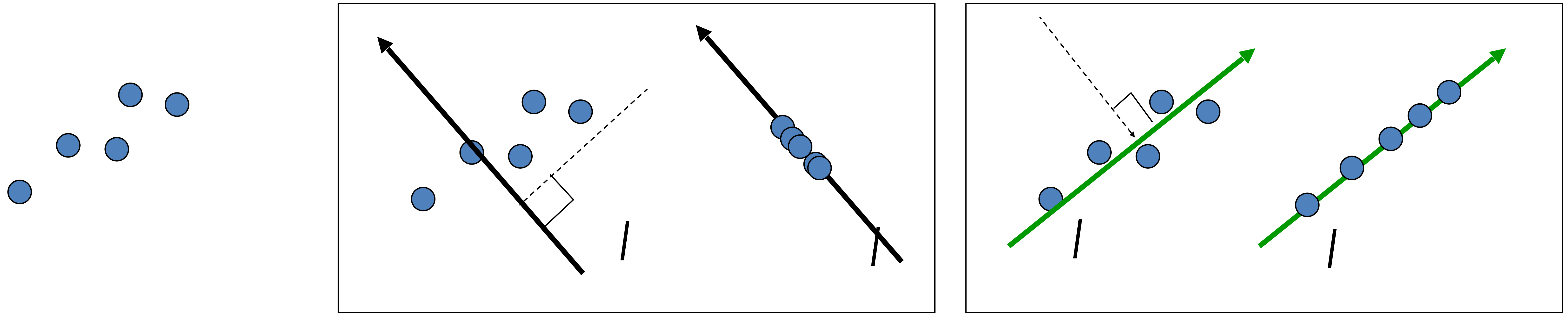
What does the Scatter Matrix do?

- Let's look at a line *l* through the center of mass **m** with direction vector **v**, and project our points $\mathbf{x}_i$ onto it. The variance of the projected points $\mathbf{x'}_i$ is:



$$\mathrm{var}(\mathbf{x}_1, \ldots \mathbf{x}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}'_i - \mathbf{m}\|^2 =$$

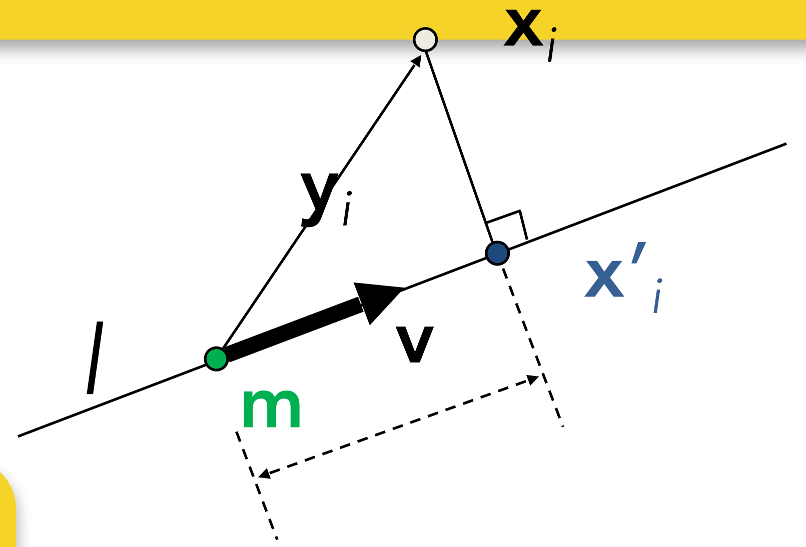
$$= \frac{1}{n}\sum_{i=1}^{n} \|(\mathbf{m} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{y}_i^T\mathbf{v})^2 = \boxed{\frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}}$$



Original set | Small variance | Large variance

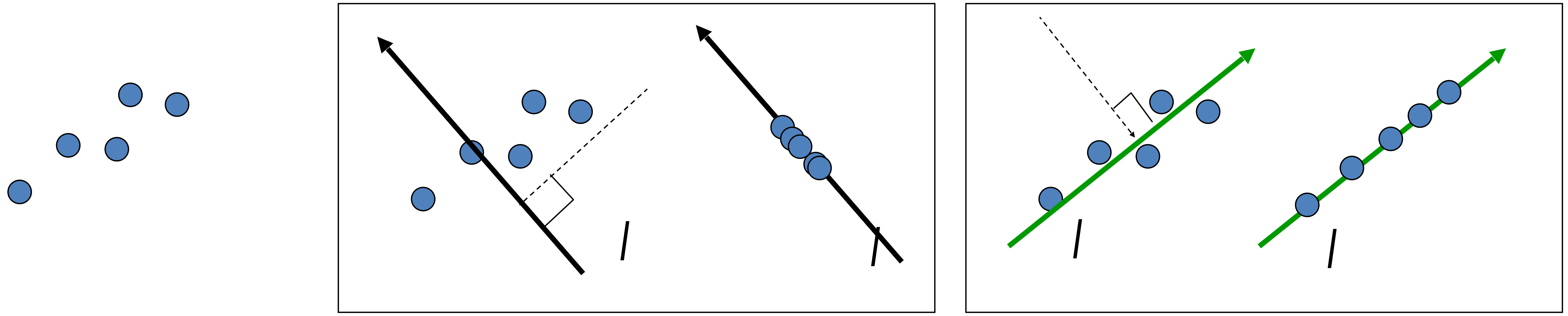
What does the Scatter Matrix do?

- The scatter matrix measures the variance of our data points along the direction **v**



$$\mathrm{var}(\mathbf{x}_1, \ldots \mathbf{x}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}'_i - \mathbf{m}\|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n} \|(\mathbf{m} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{y}_i^T\mathbf{v})^2 = \boxed{\frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}}$$



Original set | Small variance | Large variance

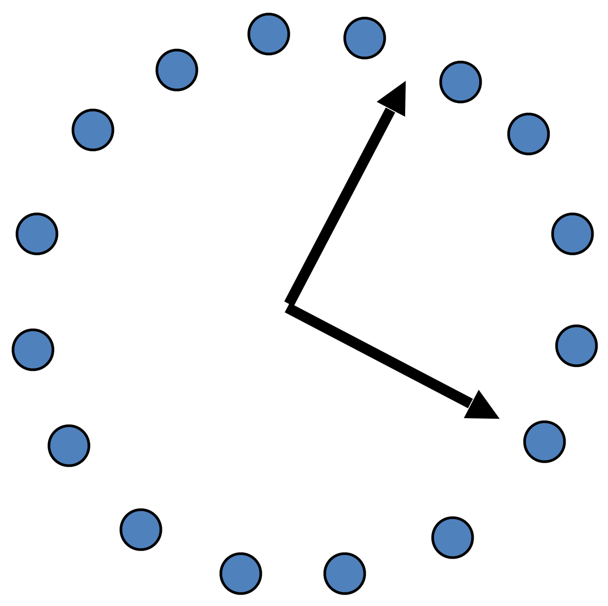
University of Victoria | Computer Science

Principal Components

- Eigenvectors of **S** that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^T$$

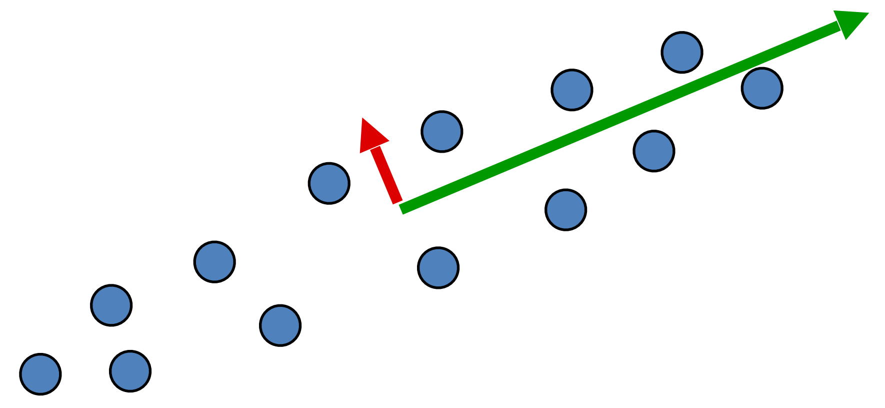
Principal Components



- There’s no preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix} \mathbf{V}^T$$

- Any vector is an eigenvector



- There’s a clear preferable direction
- S looks like this:

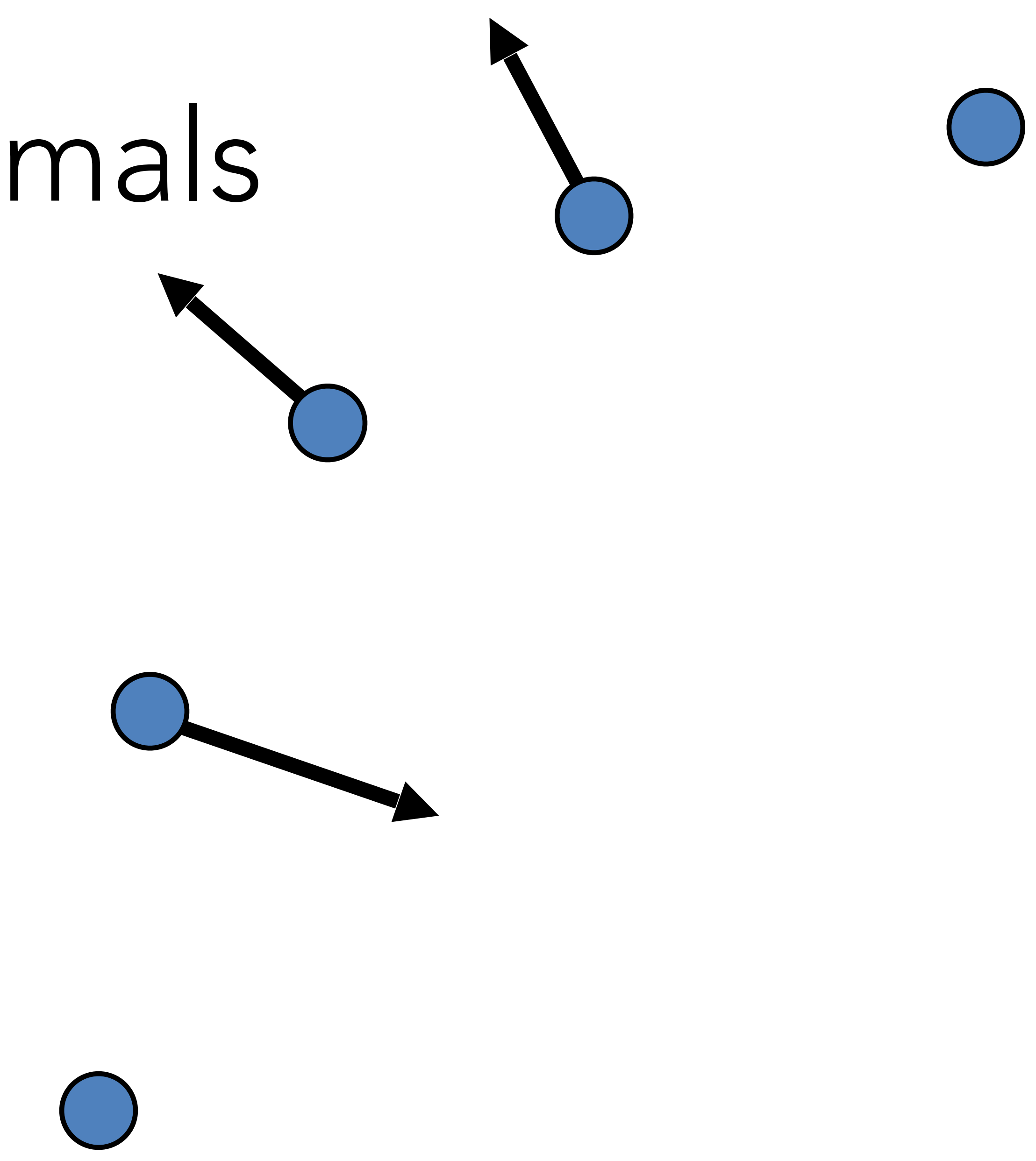
$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} \mathbf{V}^T$$

- μ is close to zero, much smaller than λ

University of Victoria | Computer Science

Normal Orientation

- PCA may return arbitrarily oriented eigenvectors
- Wish to orient consistently
- Neighboring points should have similar normals



Normal Orientation

- Build graph connecting neighboring points
  - Edge (i,j) exists if $\mathbf{x}_i \in \mathrm{kNN}(\mathbf{x}_j)$ or $\mathbf{x}_j \in \mathrm{kNN}(\mathbf{x}_i)$
- Propagate normal orientation through graph
  - For neighbors $\mathbf{x}_i$, $\mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^T\mathbf{n}_j < 0$
  - Fails at sharp edges/corners
- Propagate along "safe" paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

$$w_{ij} = 1 - \left|\mathbf{n}_i^T \mathbf{n}_j\right|$$

"Surface reconstruction from unorganized points", Hoppe et al., SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf