Least-Squares Solution

詹冠其

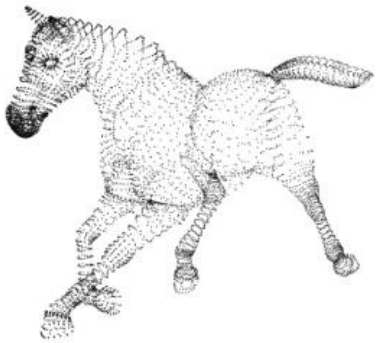
James Championchess

Overview:Why LS solution is important

- **The solution is easy to obtain.**

In this chapter, we have learned about Least-Squares fitting and solution. Recall that formulating the optimization using quadratic or *square* terms allows us to get a linear system after taking partial derivatives. The resultant linear system is simply solvable using basic linear algebra tools.

- **From discrete points, we can use it to approximate and obtain the surface of these points.**



北京大学

Content

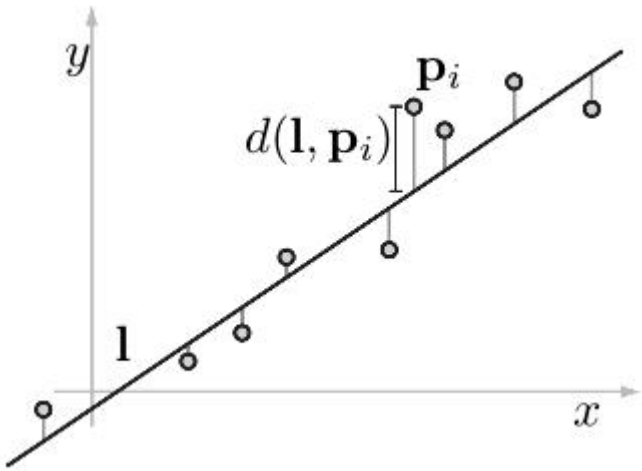
- LS line fitting
- LS polynomial fits
- Solving linear systems in the LS sense
- Weighed approach to deal with outliers
- Local surface fitting to 3D points
- Expansion: RANSAC
- Expansion: the relation with SVM
- Expansion: the relation with AdaBoost

1.Line fitting

- Can only give meaningful results if the underlying model from which data is being observed is indeed a line or close to being linear.
- From a point to the line, there are two kinds of distances as the following.

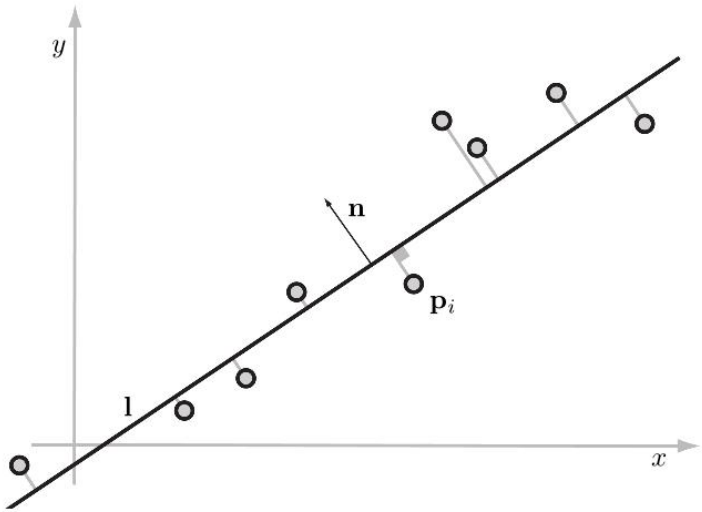
Two kind of distances and difference

- vertical distance



$$d(\mathbf{l}, \mathbf{p}_i) := y_i - ax_i - b.$$

projected distance



$$\mathbf{n}^\top \mathbf{p}_i + d$$

- Dependent of coordinate system/ easier to calculate
- Independent of coordinate system

When using vertical distance

- **Goal:** minimize

$$E(a,b) := \sum_{i=1}^{n} (y_i - ax_i - b)^2,$$

- **Remark:** $\sum_i |d(\mathbf{l}, \mathbf{p}_i)|^k$ **is also ok, but if k>2, we will get a non-linear optimization system, which is more challenging to deal with.**

- **Solution:**

$$\partial E(a,b)/\partial a = \sum_{i=1}^{n} (-2x_i)(y_i - ax_i - b) = 0\,,$$

$$\partial E(a,b)/\partial b = \sum_{i=1}^{n} (-2)(y_i - ax_i - b) = 0\,.$$

$$\left(\sum_{i=1}^{n} x_i^2\right) a + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} x_i y_i\,,$$

$$\left(\sum_{i=1}^{n} x_i\right) a + nb = \sum_{i=1}^{n} y_i\,,$$

$$\left(\sum_{i=1}^{n} \begin{bmatrix} x_i^2 & x_i \\ x_i & 1 \end{bmatrix}\right) \begin{bmatrix} a \\ b \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} x_i y_i \\ y_i \end{bmatrix}.$$

When using projected distance

- **Goal: Find a line** $\mathbf{n}^\top \mathbf{x} + d = 0, \quad \min_{\mathbf{n},d} E(\mathbf{n}, d) = \sum_{i=1}^{n} (\mathbf{n}^\top \mathbf{p}_i + d)^2, \quad \text{s.t. } \|\mathbf{n}\| = 1\,.$
- **Solution:**

$$\partial E(\mathbf{n}, d)/\partial d = 0$$

$$\Rightarrow \quad 2\sum_i (\mathbf{n}^\top \mathbf{p}_i + d) = 0$$

$$\Rightarrow \quad nd = -\mathbf{n}^\top \sum_i \mathbf{p}_i$$

$$\Rightarrow \quad d = -\mathbf{n}^\top \bar{\mathbf{p}}\,, \qquad \bar{\mathbf{p}} = \textstyle\sum_i \mathbf{p}_i / n$$

then plug d into the objective function

$$\min_{\mathbf{n}} E(\mathbf{n}) = \sum_{i=1}^{n} (\mathbf{n}^\top \mathbf{p}_i - \mathbf{n}^\top \bar{\mathbf{p}})^2 = \sum_{i=1}^{n} (\mathbf{n}^\top \tilde{\mathbf{p}}_i)^2, \quad \text{s.t. } \|\mathbf{n}\| = 1\,, \quad \tilde{\mathbf{p}}_i = \mathbf{p}_i - \bar{\mathbf{p}}$$

$$\min_{\mathbf{n}} \Big(\sum_i (\mathbf{n}^\top \tilde{\mathbf{p}}_i)^2 + \lambda(1 - \mathbf{n}^\top \mathbf{n}) \Big) = \min_{\mathbf{n}} (\mathbf{n}^\top C \mathbf{n} + \lambda(1 - \mathbf{n}^\top \mathbf{n}))\,, \quad C = \textstyle\sum_i \tilde{\mathbf{p}}_i \tilde{\mathbf{p}}_i^\top$$

$$2C\mathbf{n} - 2\lambda\mathbf{n} = 0 \quad \Rightarrow \quad C\mathbf{n} = \lambda\mathbf{n}\,.$$

and we choose the minimal λ

Plug n back into the expression of d, then we get d, so we get the line.n is actually the smallest eigenvector of the covariance matrix

2.LS polynomial fitting

- Background: if the points come from a parabola, or some other higher-order function.
- The m-th order polynomial function: $y = f(x) = \sum_{k=0}^{m} a_k x^k.$
- Goal: minimize $E(a_0, \ldots, a_m) := \sum_{i=1}^{n} (f(x_i) - y_i)^2 = \sum_{i=1}^{n} \left(\sum_{k=0}^{m} a_k x_i^k - y_i \right)^2$
- Solution: by setting $\partial E(a_0, \ldots, a_m)/\partial a_k = 0$, for $k = 0, \ldots, m.$
- Example: $m = 2,\ y = f(x) = \sum_{k=0}^{2} a_k x^k = a_0 + a_1 x + a_2 x^2.$

  $\partial \sum_i \left(y_i - \sum_{k=0}^{2} a_k x_i^k\right)^2 / \partial a_j$ for $j = 0, 1, 2,$

$$\left(\sum_{i=1}^{n} \begin{bmatrix} x_i^4 & x_i^3 & x_i^2 \\ x_i^3 & x_i^2 & x_i \\ x_i^2 & x_i & 1 \end{bmatrix} \right) \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} x_i^2 y_i \\ x_i y_i \\ y_i \end{bmatrix}.$$

北京大学

General cases

- If only the expression f(x) linearly depends on the parameters a,b,c,..., we can also extract an LS solution using a linar system. We can also set the partial derivatives to 0:

$$E(a,b,c,\dots) := \sum_i (y_i - f_{abc\dots}(x_i))^2$$

$$\begin{aligned}
\partial E(a,b,c,\dots)/\partial a &= \sum_i (-2\partial f_{abc\dots}(x_i)/\partial a)[y_i - f_{abc\dots}(x_i)] = 0, \\
\partial E(a,b,c,\dots)/\partial b &= \sum_i (-2\partial f_{abc\dots}(x_i)/\partial b)[y_i - f_{abc\dots}(x_i)] = 0, \\
&\vdots \qquad (3.9)
\end{aligned}$$

- f(x) doesn't need to be polynome

$$f_{\lambda_1,\lambda_2,\lambda_3}(x) = \lambda_1 \sin^2 x + \lambda_2 \exp(-x^2/\sigma) + \lambda_3 x^{17},$$

$$f_{\lambda_1,\lambda_2,\dots,\lambda_k}(x) = \lambda_1 f_1(x) + \lambda_2 f_2(x) + \cdots + \lambda_k f_k(x)\,.$$

北京大学

3. Solving linear systems in the LS sense

- **Background: When in a set of linear equations, the number of equation is more than the number of unknowns, it's called an overdetermined system. Each equation contains some noise, or simply inaccurate observations.**
- **Goal:** $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2 = \min_{\mathbf{x}} (\mathbf{x}^\top A^\top A\mathbf{x} - 2\mathbf{x}^\top A^\top \mathbf{b} + \mathbf{b}^\top \mathbf{b})\,.$
- **Solution: Set the partial derivative of x to 0:**

$$A^\top (A\mathbf{x} - \mathbf{b}) = 0$$
$$A^\top A\mathbf{x} = A^\top \mathbf{b}\,.$$

- **Remark:**

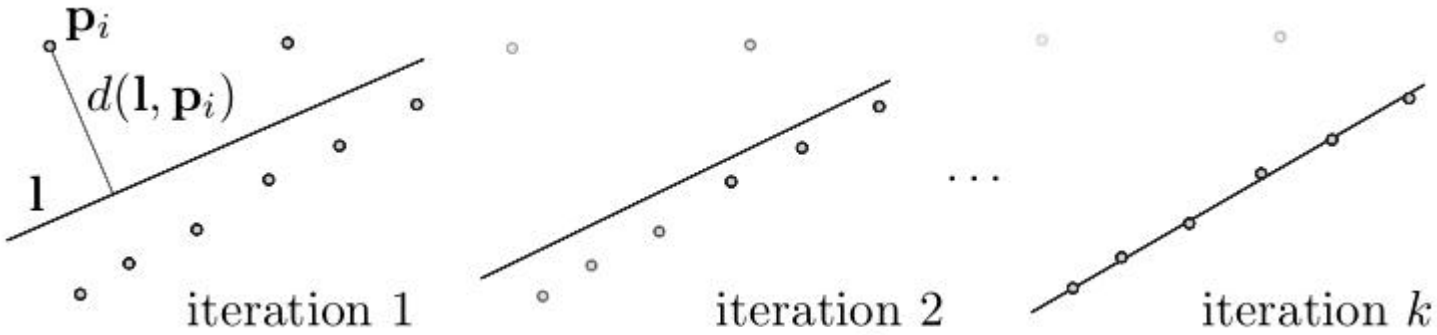
  The term $(A\mathbf{x} - \mathbf{b})$ is called the *residual vector*, the part of the data that is not explained by the LS solution.

- **Conclusion: if A has full rank,** $\mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{b}.$

北京大学

4. Weighed approach

- **Background: when there are some outliers in the sample points, the traditional LS solution will be quite skewed.**



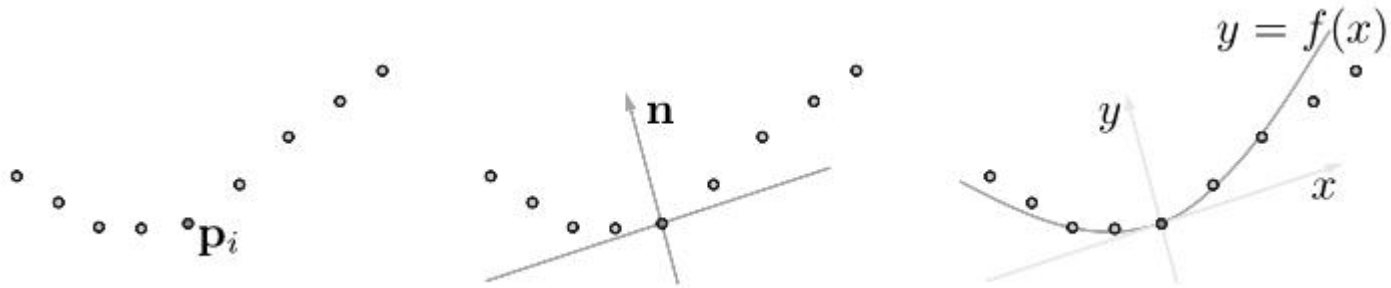
- **Goal and solution:** $\min_{\mathbf{l}} E(\mathbf{l}) := \sum_{i=1}^{n} w_i \, d(\mathbf{l}, \mathbf{p}_i)^2.$

$$\left(\sum_i \begin{bmatrix} w_i x_i^2 & w_i x_i \\ w_i x_i & w_i \end{bmatrix}\right) \begin{bmatrix} a \\ b \end{bmatrix} = \sum_i \begin{bmatrix} w_i x_i y_i \\ w_i y_i \end{bmatrix}. \qquad A^\top W A \mathbf{x} = A^\top W \mathbf{b}\,,$$

- **Iterations:** $w_{j+1} = \exp(-d(\mathbf{l}_j, \mathbf{p}_i)/\sigma)$
- **This method can handle data with a moderate amount of outliers.**

5. Local surface fitting to 3D points

- Background: in 3D cases, we need to reconstruct or approximate the local surface of each point. Once we have a local surface normal, we have a local reference plane.



- Firstly, find the local surface normal of each point and obtain the local reference plane at each point.

$\mathbf{p} \in \mathbf{P}$, let $N_r(\mathbf{p}) = \{\mathbf{q}_i\}$ i.e., $\|\mathbf{p} - \mathbf{q_i}\| \leq r$.

$\mathbf{n}$ be the unit normal to $\mathcal{S}$ at $\mathbf{p}$,

$\mathbf{n}^\top \mathbf{x} + d = 0$. the local reference plane

- **Goal:** $\min_{\mathbf{n},d}(\mathbf{n}^\top \mathbf{q}_i + d)^2$ **s.t.** $\|\mathbf{n}\| = 1.$
- **Solution:** $\mathbf{n}$ is the smallest eigenvector of the covariance matrix

  $\sum_i (\mathbf{q}_i - \bar{\mathbf{q}})(\mathbf{q}_i - \bar{\mathbf{q}})^\top,$ $\bar{\mathbf{q}} = \sum_i \mathbf{q}_i / |N_r(\mathbf{p})|$
- **Remark: r should be appropriate. If too small, the fit might be quite susceptible to noise; If too large, local variations such as sharp features can get averaged out.**
- **Secondly, locally fit a polynomial around the point.**

$$z = F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f,$$

$$\begin{aligned}
ax_1^2 + bx_1y_1 + cy_1^2 + dx_1 + ey_1 + f &= z_1, \\
ax_2^2 + bx_2y_2 + cy_2^2 + dx_2 + ey_2 + f &= z_2, \\
&\dots \\
ax_n^2 + bx_ny_n + cy_n^2 + dx_n + ey_n + f &= z_n\,.
\end{aligned}$$

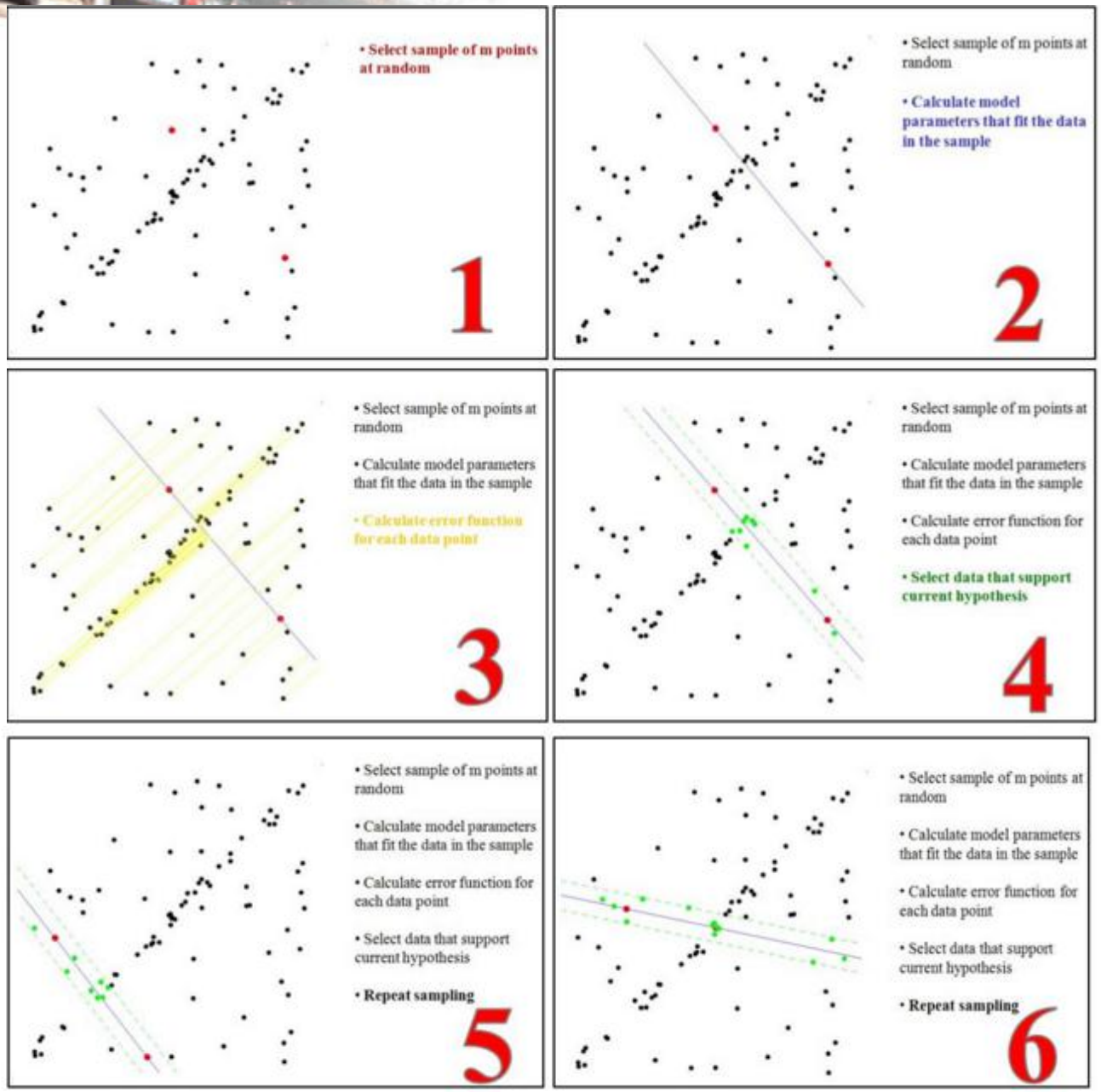
$$\text{minimize } \sum_i \|z_i - F(x_i, y_i)\|^2.$$

$$\text{minimize } \sum_i w_i \|z_i - F(x_i, y_i)\|^2,$$

北京大学

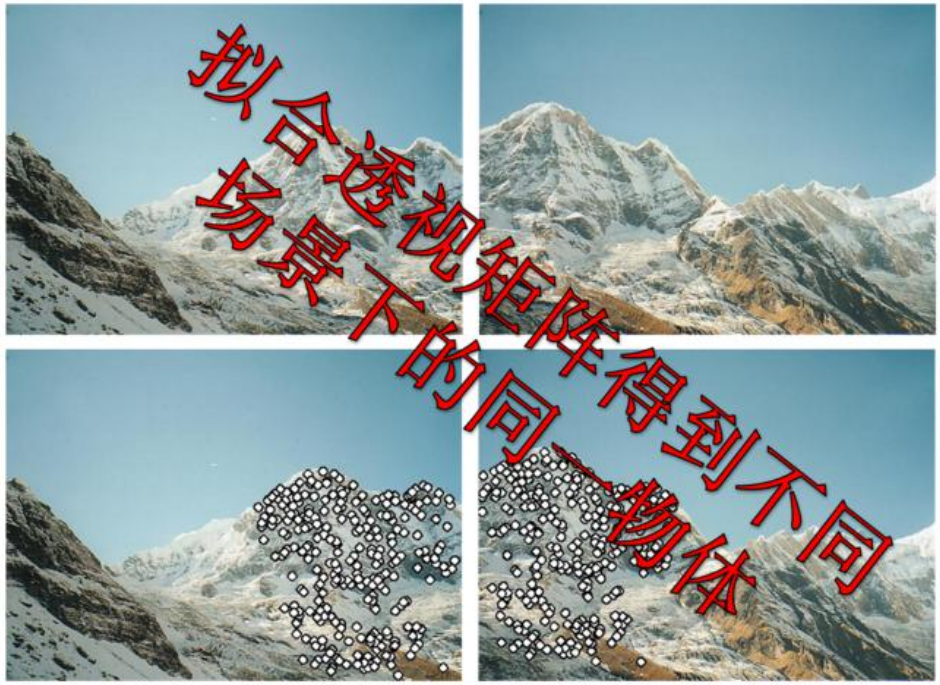
Expansion: RANSAC

- When there are a large proportion of points that are outliers, weighed approach can't handle either.
- Random Sample Consensus can deal with it!
- Five steps to realize the algorithm:
- 有一个模型适应于假设的局内点，即所有的未知参数都能从假设的局内点计算得出。
- 用1中得到的模型去测试所有的其它数据，如果某个点适用于估计的模型，认为它也是局内点。
- 如果有足够多的点被归类为假设的局内点，那么估计的模型就足够合理。
- 然后，用所有假设的局内点去重新估计模型（譬如使用最小二乘法），因为它仅仅被初始的假设局内点估计过
- 最后，通过估计局内点与模型的错误率来评估模型。
- 上述过程被重复执行固定的次数，每次产生的模型要么因为局内点太少而被舍弃，要么因为比现有的模型更好而被选用。


• Select sample of m points at random
1
• Select sample of m points at random
• Calculate model parameters that fit the data in the sample
2
• Select sample of m points at random
• Calculate model parameters that fit the data in the sample
• Calculate error function for each data point
3
• Select sample of m points at random
• Calculate model parameters that fit the data in the sample
• Calculate error function for each data point
• Select data that support current hypothesis
4
• Select sample of m points at random
• Calculate model parameters that fit the data in the sample
• Calculate error function for each data point
• Select data that support current hypothesis
• Repeat sampling
5
• Select sample of m points at random
• Calculate model parameters that fit the data in the sample
• Calculate error function for each data point
• Select data that support current hypothesis
• Repeat sampling
6

北京大学

Applications



Expansion: the relation with SVM

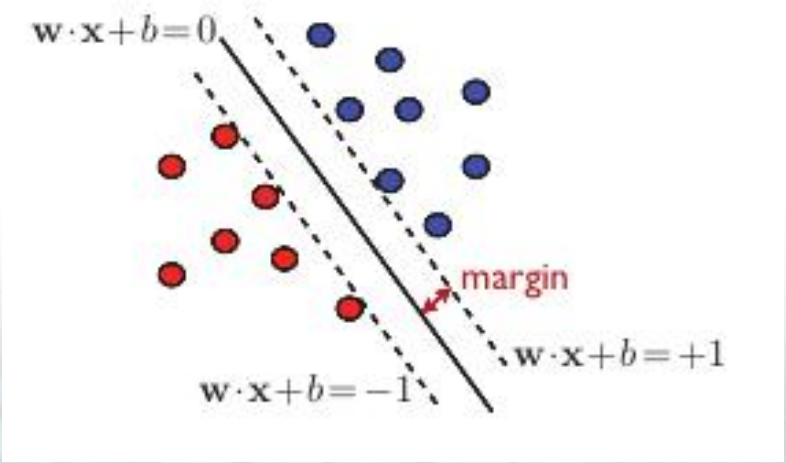
- **The key idea of dealing with noise is similar**

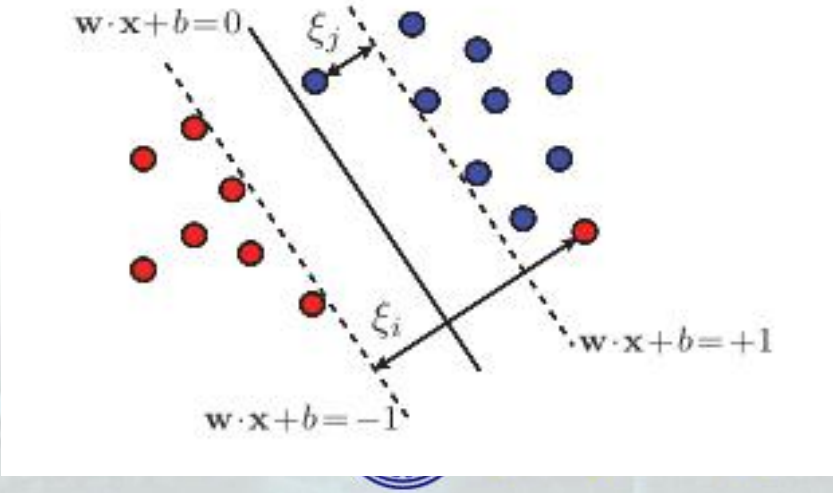
This leads to the following general optimization problem defining SVMs in the non-separable case where the parameter $C \geq 0$ determines the trade-off between margin-maximization (or minimization of $\|\mathbf{w}\|^2$) and the minimization of the slack penalty $\sum_{i=1}^{m} \xi_i^p$:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i^p \qquad (4.23)$$

$$\text{subject to} \ \ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \ \wedge \ \xi_i \geq 0, i \in [1, m],$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\top$. The parameter C is typically determined via n-fold cross-validation (see section 1.3).





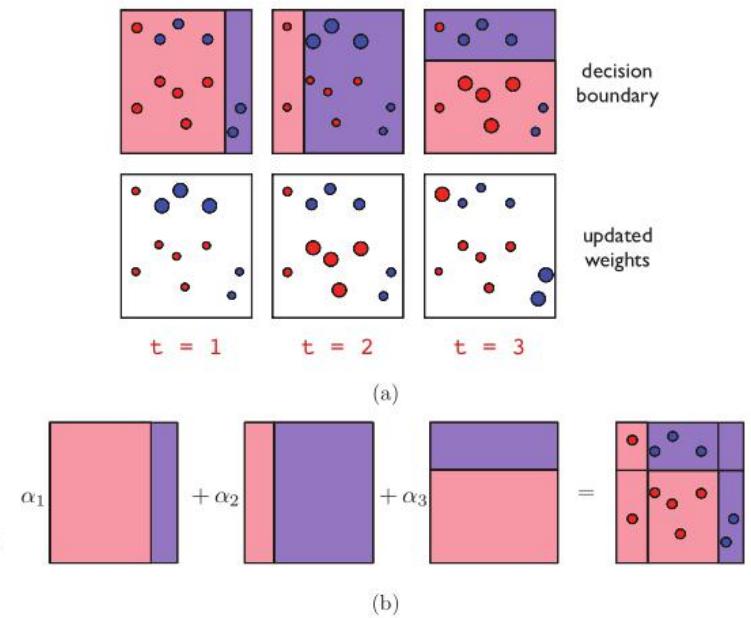
Expansion: the relation with AdaBoost

- Adjust the weight of each item through iterations
- Give the "important" ones more weight and "unimportant ones" less weight.

$\textsc{AdaBoost}(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

for $i \leftarrow 1$ **to** m **do**
  $D_1(i) \leftarrow \frac{1}{m}$
for $t \leftarrow 1$ **to** T **do**
  $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
  $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  $Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$ ▷ normalization factor
  for $i \leftarrow 1$ **to** m **do**
    $D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
$g \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
return $h = \mathrm{sgn}(g)$

Figure 6.1 AdaBoost algorithm for $H \subseteq \{-1, +1\}^{\mathcal{X}}$.



- 谢谢
- thanks
- ありがとうございます
- 감사합니다
- Merci
- Danke
- gracias

北京大學

School of EECS, Peking University

LS Applications: Mesh reconstruction & Skin calculation

Pengcheng Xu
jsteward@pku.edu.cn

March 4, 2019

Content

Mesh reconstruction
- Problem definition
- Formal definition
- Solution

Skin reconstruction
- Introduction
- Definitions
- Skin estimation methods
- Overfitting

Mesh reconstruction
Problem definition

- Definition: construct mesh from a set of point samples
- Abstraction: given a planar graph with arbitrary connectivity and a sparse set of **control points** that lies on the surface, reconstruct the rest of the surface.

Mesh reconstruction
Formal definition

- Mesh graph: a graph whose drawing, embedded in some Euclidean space $\mathbb{R}^n$, forms a regular tiling.
- For a given mesh graph $G = (V, E)$, the following defines a fairness and smoothness condition for vertex $\mathbf{v}_i$:

$$\mathbf{v}_i - \frac{1}{d_i} \sum_{j:(i,j)\in E} \mathbf{v}_j = 0 \tag{1}$$

- For 3D meshes, the linear system can be written as the following:

$$L\mathbf{x} = 0, L\mathbf{y} = 0, L\mathbf{z} = 0 \tag{2}$$

where L is the **Laplacian** of the mesh:

$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Mesh reconstruction
Formal definition

- To keep the sharp features of the mesh, consider a (sparse) set of **control points**:

$$\mathbf{v}_s = (x_s, y_s, z_s), s \in C, \text{ where } C = \{s_1, s_2, \ldots, s_m\} \tag{4}$$

- The system becomes:

$$A\mathbf{x} = \mathbf{b} \tag{5}$$

$$A = \begin{pmatrix} L \\ F \end{pmatrix}, F_{ij} = \begin{cases} 1 & j = s_i \in C \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

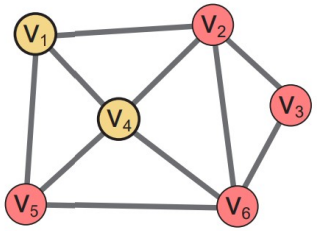
$$b_k = \begin{cases} 0 & k \leq n \\ x_{s_{k-n}} & n < k \leq n + m \end{cases} \tag{7}$$

Mesh reconstruction
Solution

- To reconstruct the mesh, find $\mathbf{x}$ that minimizes

$$\|A\mathbf{x} - \mathbf{b}\|^2 = \|L\mathbf{x}\|^2 + \sum_{s \in C} |x_s - \mathbf{v}_s^{(x)}|^2 \tag{8}$$

  in least-squares sense.
- Mesh visualization:



$$\begin{bmatrix} 1 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & -\frac{1}{2} & 1 & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{4} & 0 & 1 & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & -\frac{1}{3} \\ 0 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure: Example of graph and corresponding A matrix. Yellow vertices are control points.

Least-squares Meshes by Sorkine, Cohen-Or, Tel Aviv University

Mesh reconstruction
Visualization

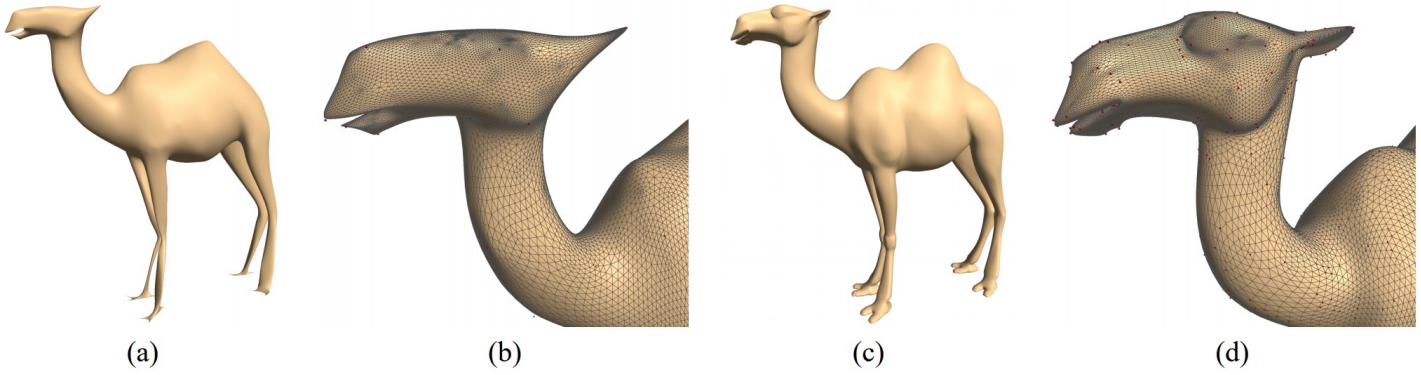


Figure: Camel: 30974 vertices in connectivity graph. (a): 1k control points; (c): 2k control points.

Mesh reconstruction
Visualization

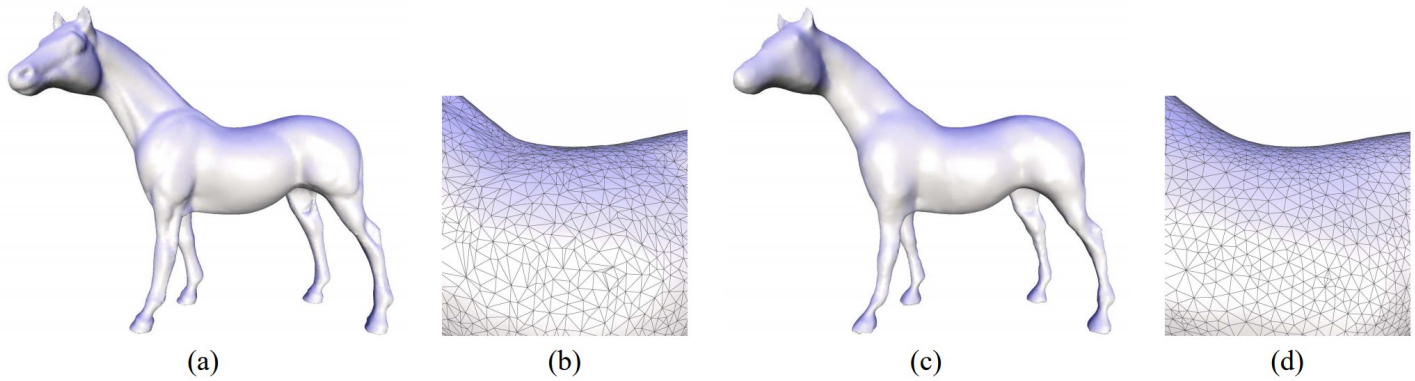


Figure: Horse: 19851 vertices in connectivity graph. (a): original graph; (c): 1k control vertices.

Skin calculation
Introduction

- Animating articulated characters such as virtual humans is a fundamental operation in computer graphics and interactive applications
- Techniques for rigging character skins by weighting vertices to an associated skeleton are widely used in video games and the computer animation industry

Skin calculation
Definitions

- Construct a vertex skinning transform, $T = (T_i)$, that approximates a sequence of input meshes.
- Let the sequence of S meshes have deformed vertex positions $\mathbf{P} = (p^1, p^2, \ldots p^S)$, where $p^t \in \mathbb{R}^{3N}$ for N vertices. (t: time). Seek a **skinning transformation** T^t at each sequence step t, that transforms the undeformed rest pose points, $\widetilde{p}$, to approximate p^t:

$$p^t \approx T^t \widetilde{p}, t = 1 \ldots S \tag{9}$$

Use linear blend skinning, where the transform for vertex i is

$$T_i^t = \sum_{b \in \mathcal{B}_i} w_{ib} \widetilde{T}_b^t \tag{10}$$

Skin calculation

Skin estimation methods

How to estimate which bones can influence each vertex and what the corresponding vertex weights are, given a mesh sequence and bone transformations $\{\widetilde{T}_b^t\}$?

- Estimating bone influences: pick β bones (β is given) that influences vertex i via LS:

$$\gamma_{bi} = \sum_{t=1\ldots S} \|p_i^t - \widetilde{T}_b^t \widetilde{p}_i\|_2^2, b = 1 \ldots B \tag{11}$$

- Estimating vertex weights: given vertex-bone influence sets $\{\mathcal{B}_i\}$, the associated weights are computed using (constrained) LS:

$$\sum_{b \in \mathcal{B}_i} (\widetilde{T}_b^t \widetilde{p}_i) w_{ib} = p_i^t, t = 1 \ldots S \tag{12}$$

of the form $\mathbf{A}^{(i)}\mathbf{w}^{(i)} = \mathbf{b}^{(i)}$, along with the affine constraint:

$$\sum_b w_{ib} = 1 \tag{13}$$

Not directly solving this due to potential "overfitting"

Weight overfitting problem

Direct solution of constrained LS leads to large positive and negative values, which is suboptimal; use the augmented system instead:

$$\begin{bmatrix} c\mathbf{A}^{(i)} \\ 1 \dots 1 \end{bmatrix} \mathbf{w}^{(i)} = \begin{pmatrix} c\mathbf{b}^{(i)} \\ 1 \end{pmatrix} \tag{14}$$

of the form $\widetilde{\mathbf{A}}^{(i)}\mathbf{w}^{(i)} = \widetilde{\mathbf{b}}^{(i)}$, where $c = \frac{1}{\|\mathbf{b}^{(i)}\|_2}$ is a scaling factor, and the ones in the last rows are for the affine constraint.

Weight overfitting problem

Solving the augmented system

- Truncated Singular Value Decomposition (TSVD): Truncate singular values at $10^{-5}\|\widetilde{\mathbf{A}}^{(i)}\|_2$ to avoid overfitting
- Non-Negative Least Squares (NNLS, variant of constrained LS):

$$\text{Solve } \widetilde{\mathbf{A}}^{(i)}\mathbf{w}^{(i)} = \widetilde{\mathbf{b}}^{(i)} \text{ subjects to } \mathbf{w}^{(i)} \geq 0. \tag{15}$$

Skinning Mesh Animations by James, Twigg, Carnegie Mellon Univ.

前沿二作业

邵俊宁
俞昊君

March 5, 2019

1 第一题

证明最小二乘法 $min|Ax-b|$ 的解为 $\{x|A^TAx=A^Tb\}$, 且任意最小二乘解 $x=$ 固定最小二乘解 x_0+kerA。

2 第二题

补全 RANSAC 的核心代码（可以写伪代码，意思正确即可）

```
template<class T, class S>
double Ransac<T,S>::compute(std::vector<S> &parameters,
                            ParameterEsitmator<T,S> *paramEstimator ,
                            std::vector<T> &data,
                            int numForEstimate)
{
    std::vector<T *> leastSquaresEstimateData;
    int numDataObjects = data.size();
    int numVotesForBest = -1;
    int *arr = new int[numForEstimate];
    // numForEstimate表示拟合模型所需要的最少点数，
    //对本例的直线来说，该值为2
    short *curVotes = new short[numDataObjects];
    //one if data[i] agrees with the current model, otherwise zero
    short *bestVotes = new short[numDataObjects];
    //one if data[i] agrees with the best model, otherwise zero


```

```
    //there are less data objects than the minimum
    //required for an exact fit
    //complete the code here

    // 计算所有可能的直线，寻找其中误差最小的解。
    //对于100点的直线拟合来说，大约需要100*99*0.5=4950次运算，
    //复杂度无疑是庞大的。一般采用随机选取子集的方式。
    computeAllChoices(paramEstimator,data,numForEstimate,
                      bestVotes, curVotes, numVotesForBest, 0,
                      data.size(), numForEstimate, 0, arr);

    //compute the least squares estimate using the
    //largest sub set
    //complete the code here

    // 对局内点再次用最小二乘法拟合出模型
    paramEstimator->leastSquaresEstimate(
    leastSquaresEstimateData,parameters);

    delete [] arr;
    delete [] bestVotes;
    delete [] curVotes;

    return (double)leastSquaresEstimateData.size()/
    (double)numDataObjects;
}
```