Model fitting and optimization

Model fitting and optimization

- Some computer vision problems belong to this category.

- Given a set of incomplete data as input –we assume a model and seek an optimized output based on the model.

Input ---Model -----Output

Model fitting

- Scattered data interpolation
- Given $\{\mathbf{x}_k\}$ and $\{\mathbf{d}_k\}$, we want to find a function

$$\mathbf{f}(\mathbf{x}_k) = \mathbf{d}_k \qquad (4.1)$$

- or an approximation

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \qquad (4.2)$$

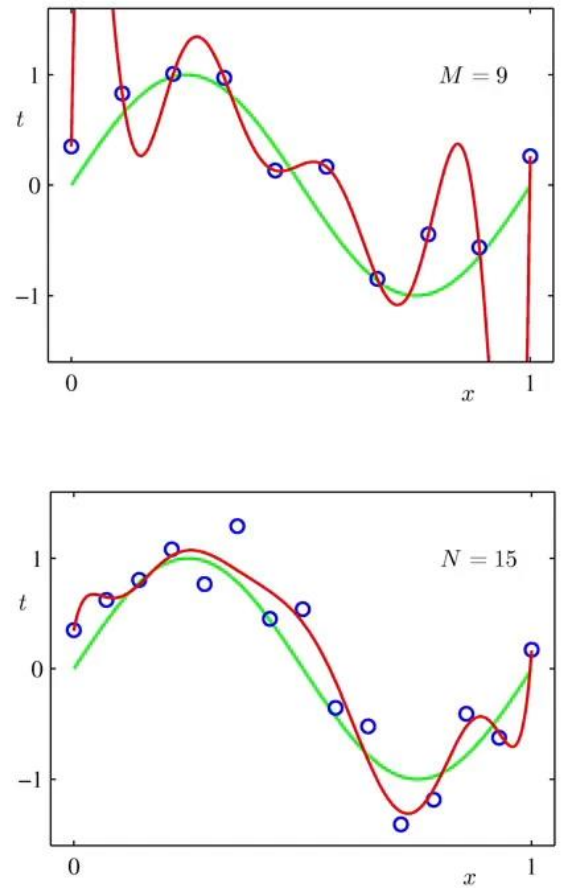
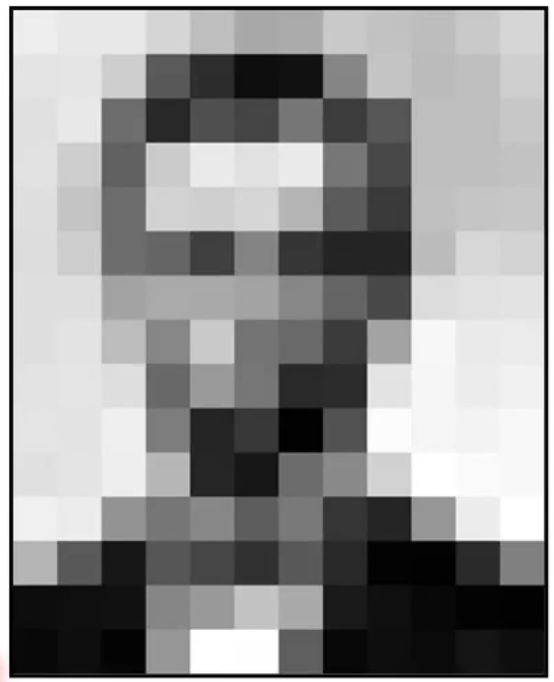


Figure 4.1(a)

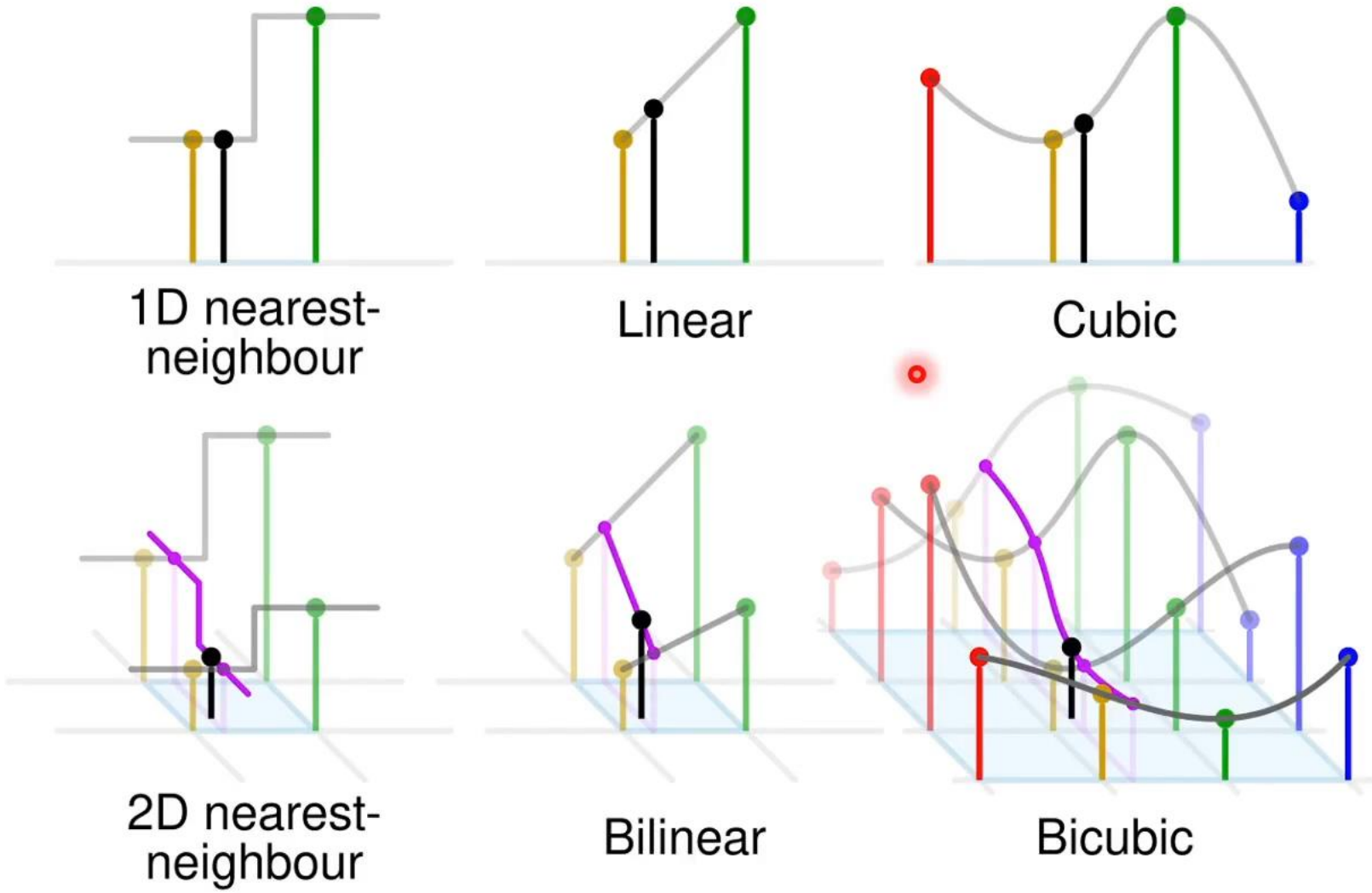
Pixels on an image

- regularly gridded



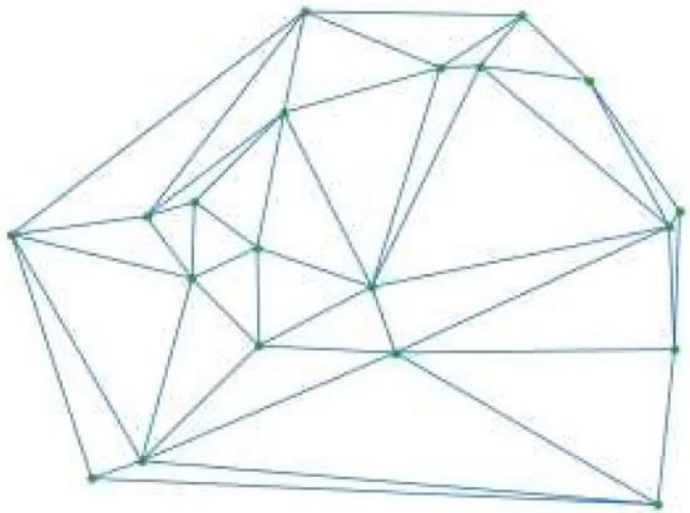
243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

Neareat/Linear/Cubic Interpolation

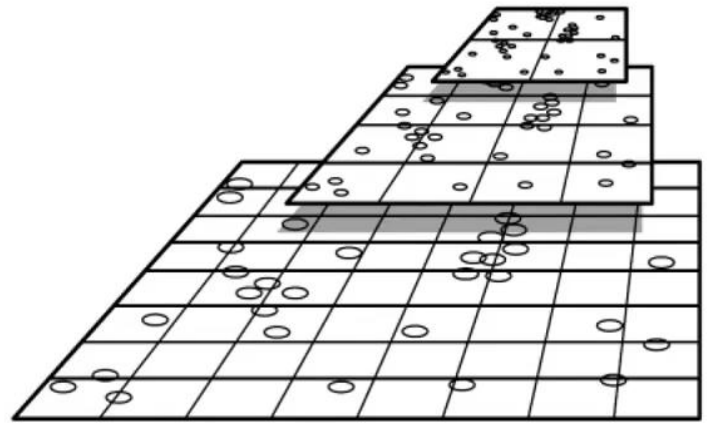


Scattered 2D Case

- Delaunay triangulation
- Pull-push algorithm



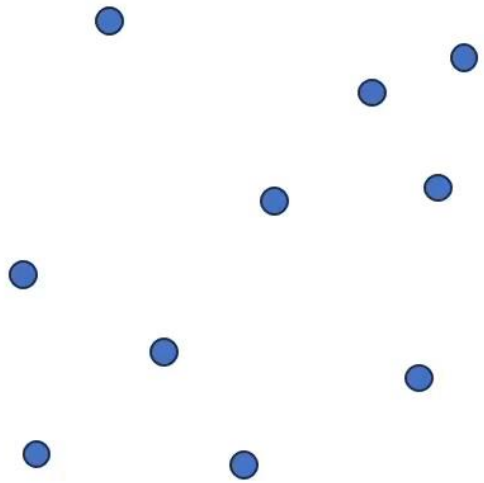
(a)



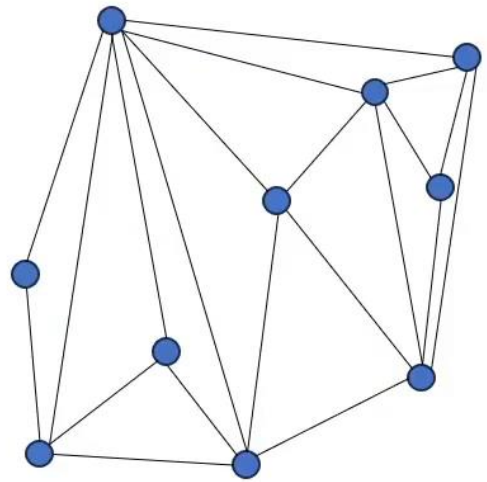
(b)

Figure 4.2

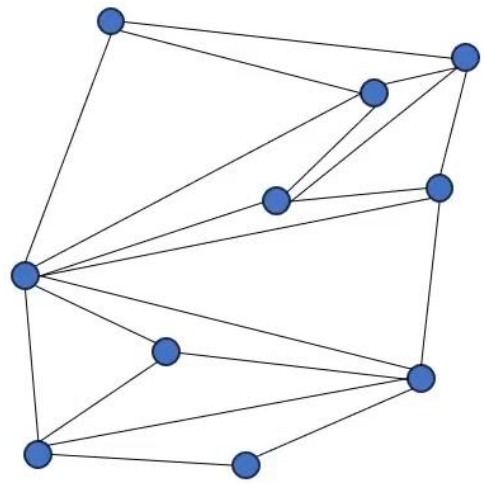
Delaunay Triangulation



Given a set of points, there are many possible meshes.

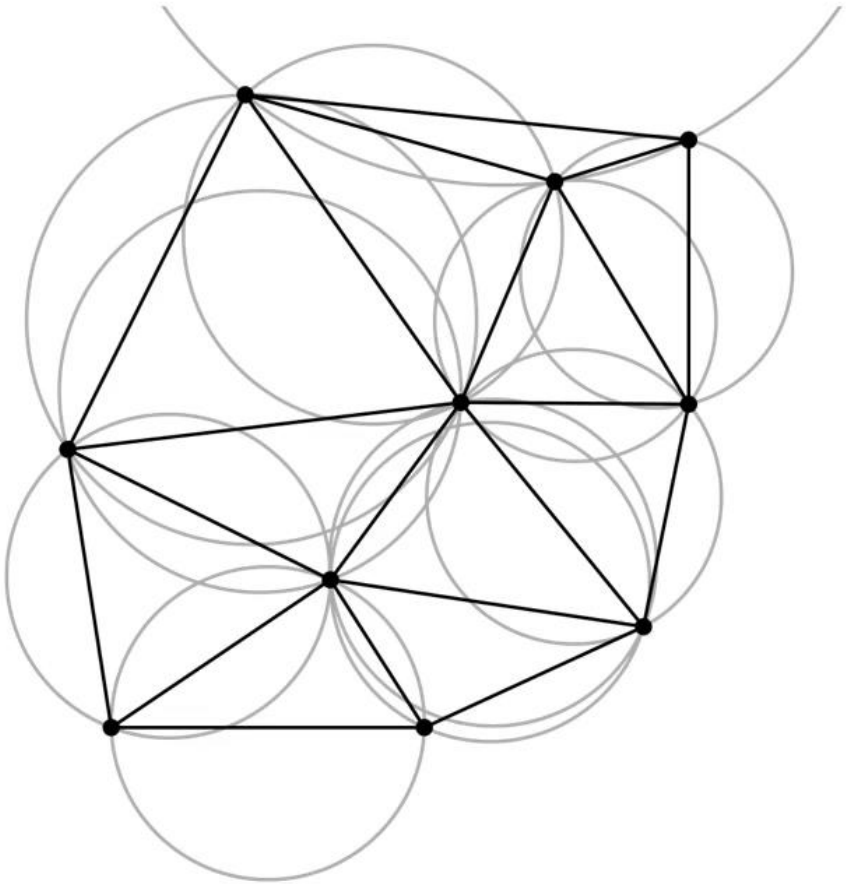


Case 1



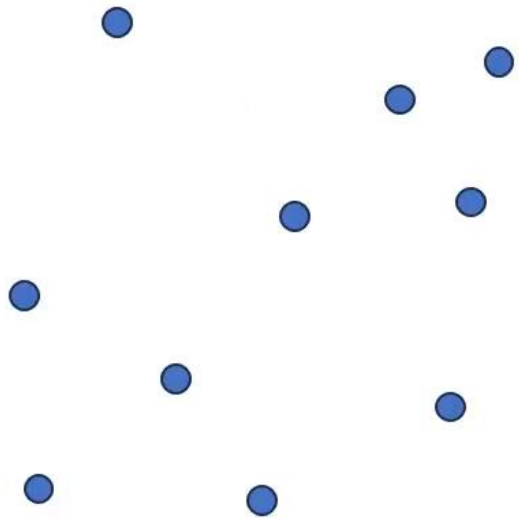
Case 2

Definition of Delaunay Triangulation

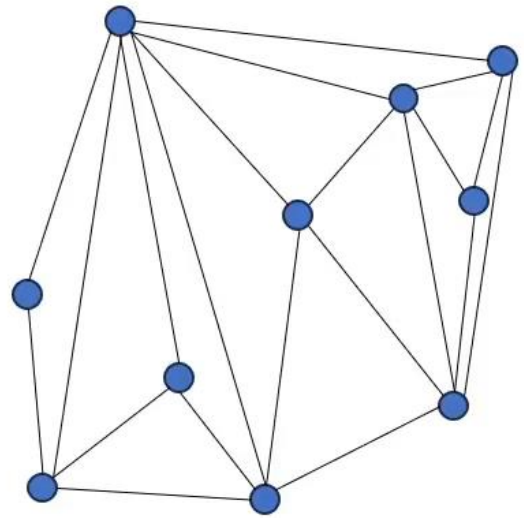


No vertex lies inside the circumcircle of any triangle.

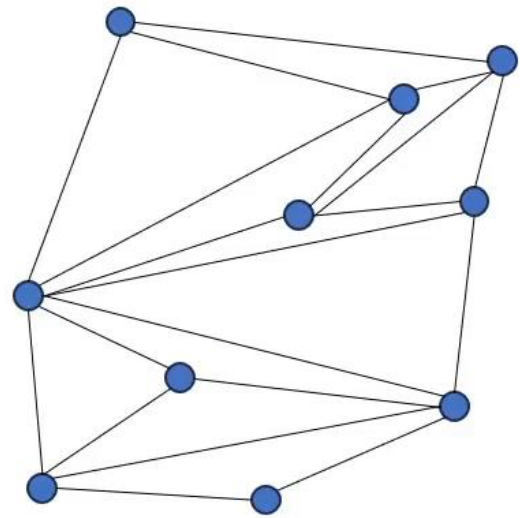
Delaunay Triangulation



Given a set of points, there are many possible meshes.



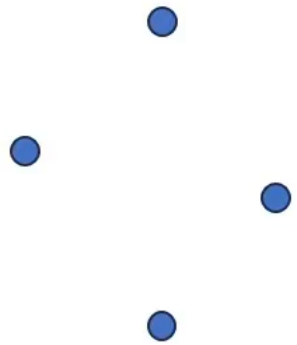
Case 1



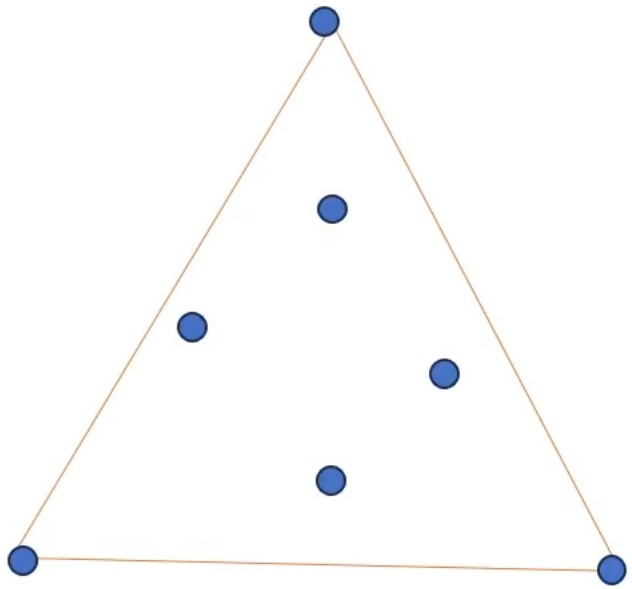
Case 2

illustration from YouTube “Delaunay Triangulation” by SCIco (Link)

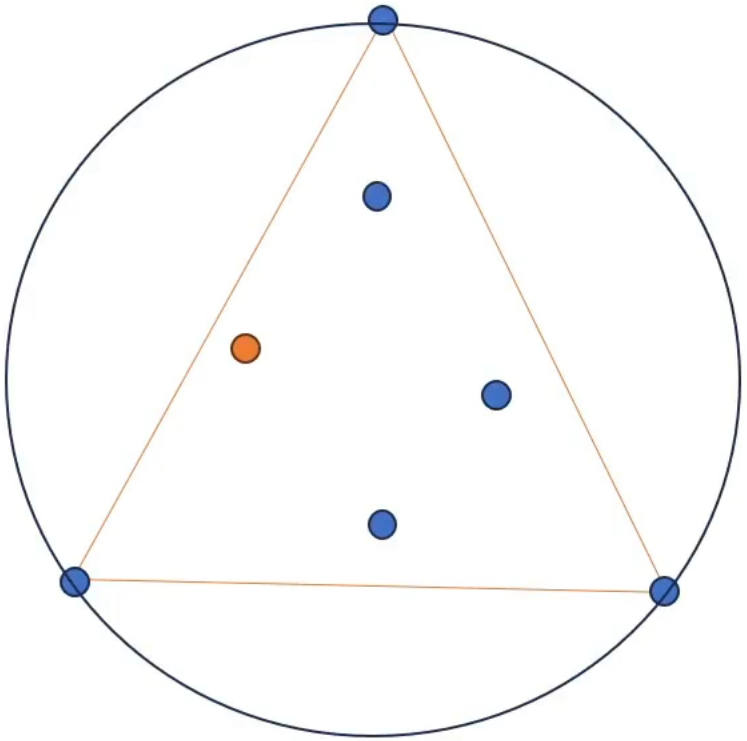
Bowyer-Watson algorithm



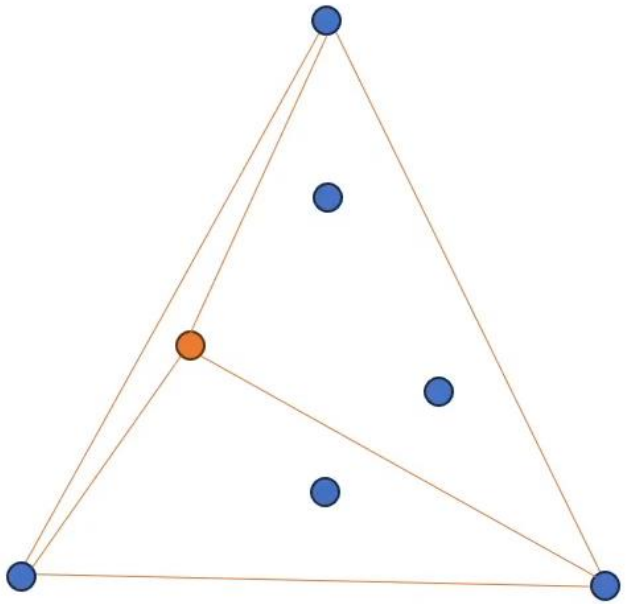
We use 4 points for illustration.



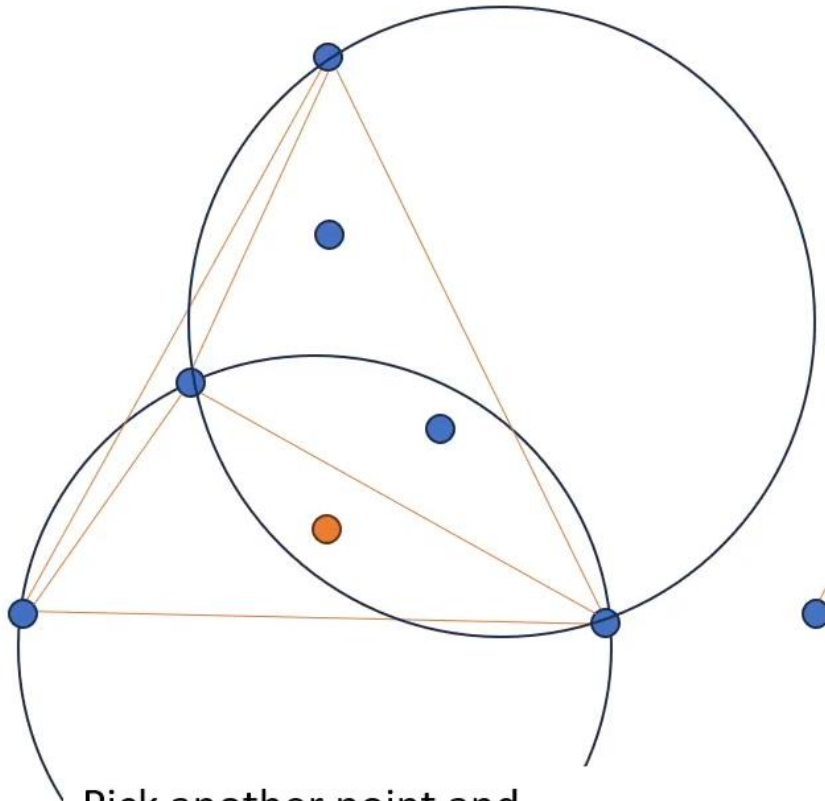
Create a super-triangle from three virtual vertices to enclose them.



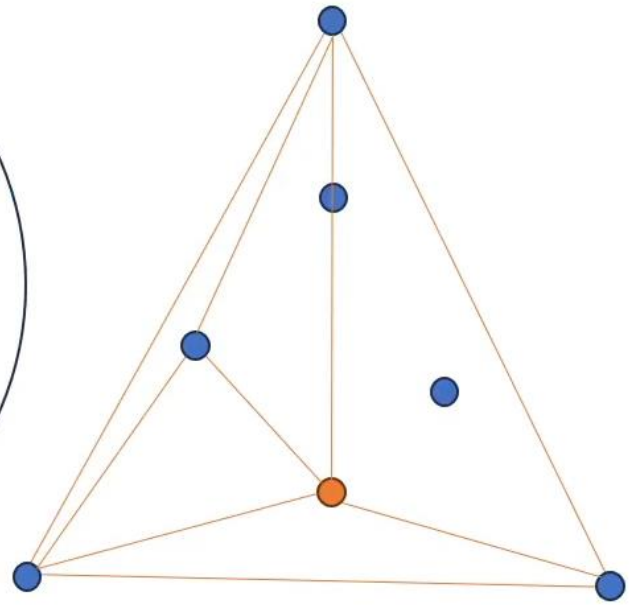
Select a point. Look for triangles whose circumcircles contain this point. It is the super-triangle this time. Thus, this is not a Delauney triangle.



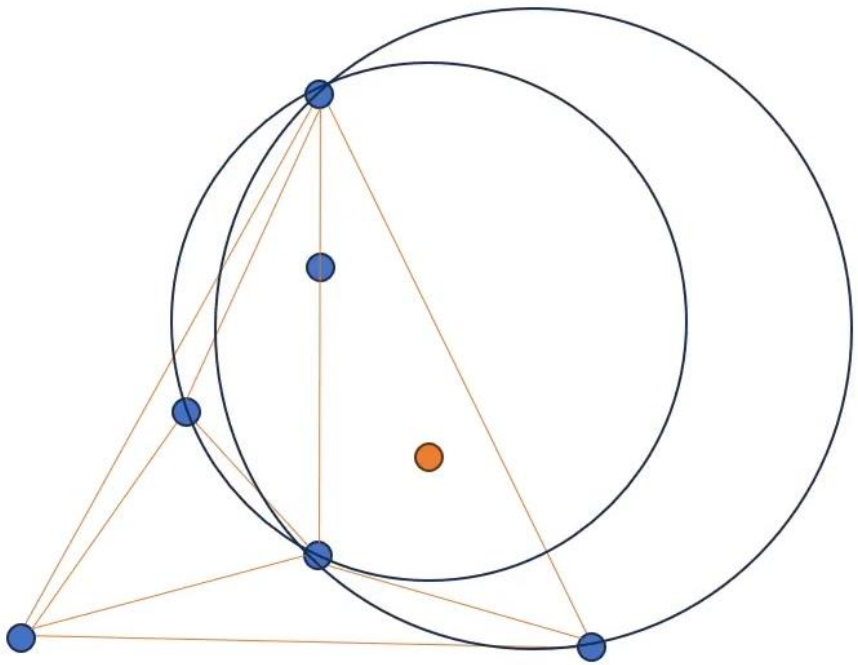
Connect this point to the three vertices to create new triangles.



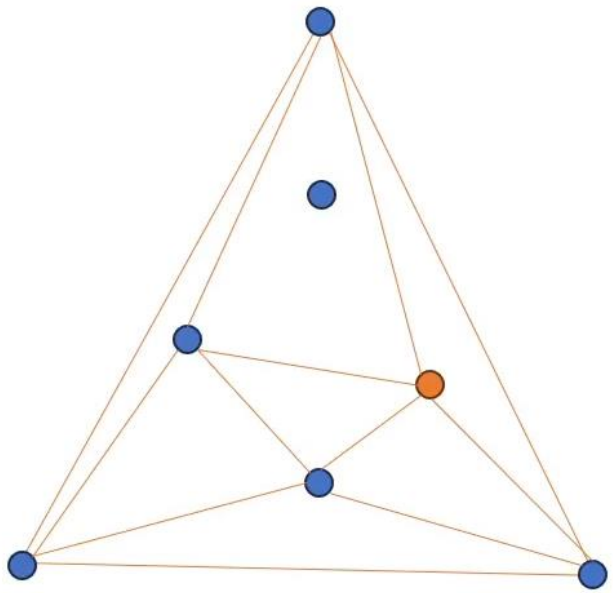
Pick another point and repeat. We have two triangles whose circumcircles contain this point.



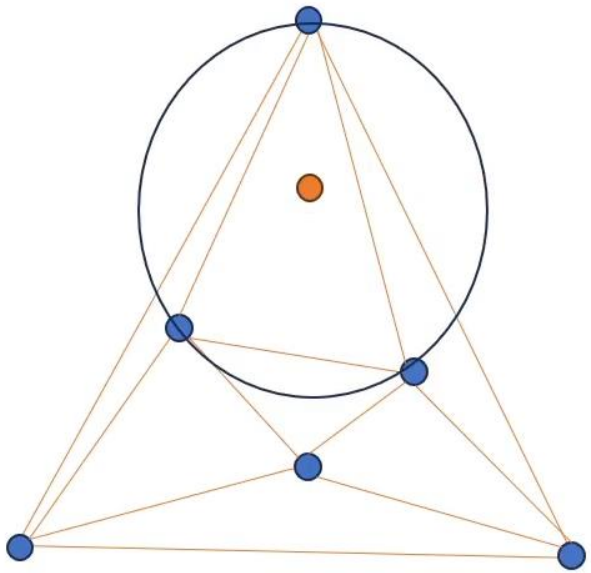
Remove their common edge and connect to their vertices.



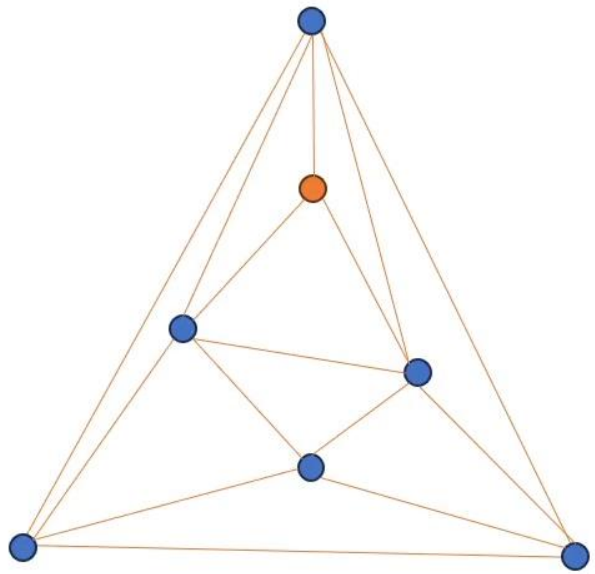
Pick another point and repeat. We have two triangles whose circumcircles cover this point.



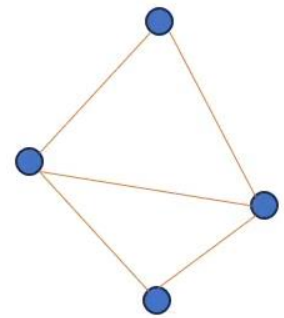
Remove their common edge and connect to their vertices.



No common edge to remove.



Connect the point to the triangle’s vertices.



Finally, delete the super-triangle vertices and their edges.

Pull-push algorithm

Images from KRAUS, Martin. The pull-push algorithm revisited. *Proceedings GRAPP*, 2009, 2: 3.

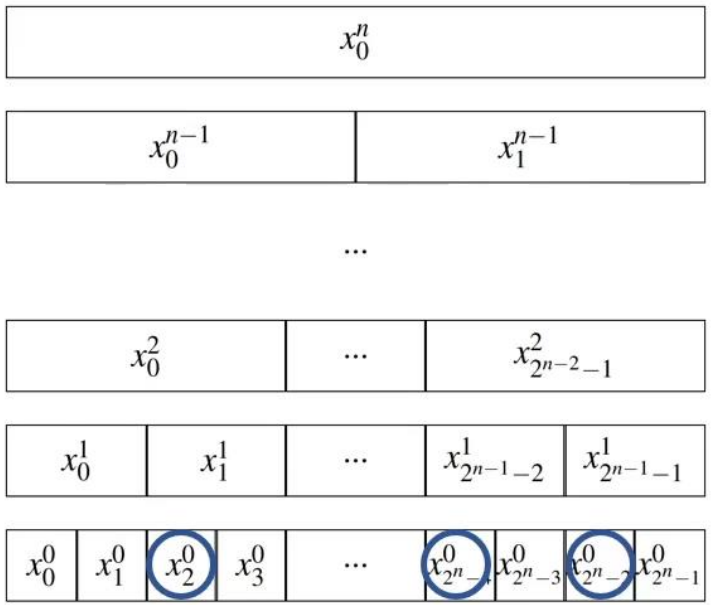
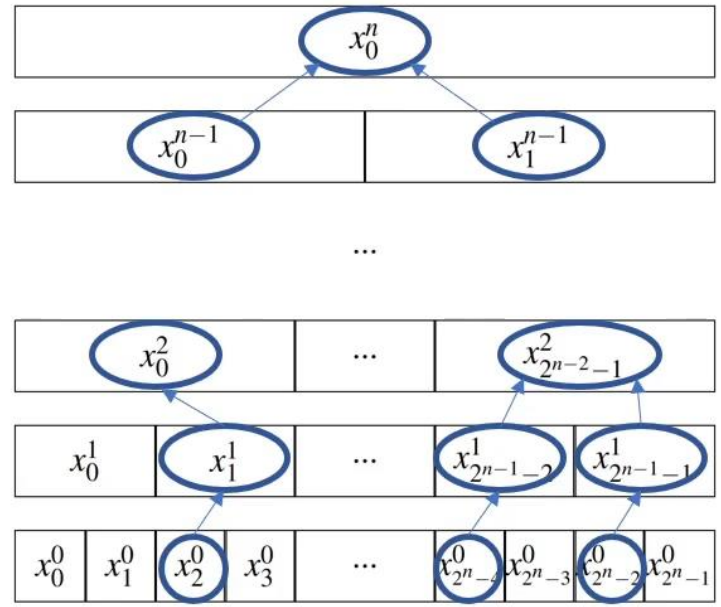
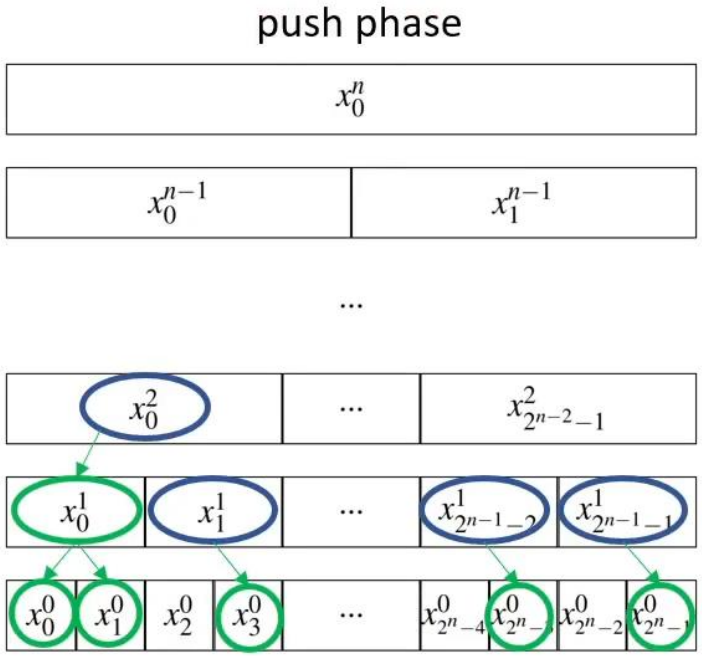
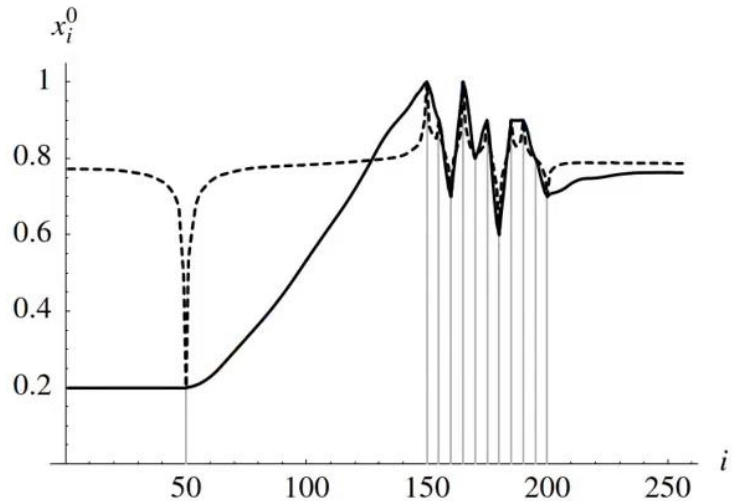


Illustration of the pixels x_i^r of an image pyramid for a one-dimensional input image x_i^0 of size 2^n.

pull phase



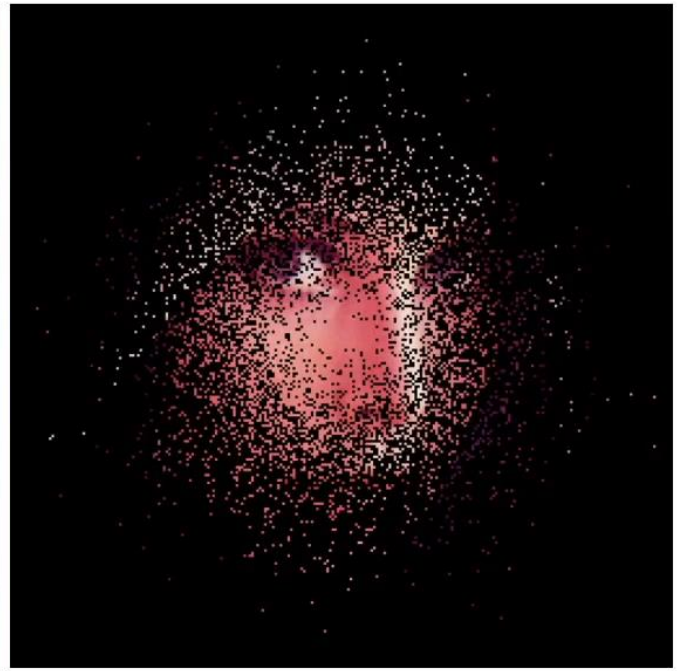




Interpolation of a set of scattered data (gray bars). Solid and gray lines are generated by different weight settings.

Image example

Lena
Original image as a reference



Input for the pull-push algorithm

Result

Images from KRAUS, Martin. The pull-push algorithm revisited. *Proceedings GRAPP*, 2009, 2: 3.

How to interpolate data in a high-dimensional space?

kernel regression $$\mathbf{f}(\mathbf{x}) = \sum_k \mathbf{w}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|) \quad (4.3)$$

$\phi(\cdot)$ is called a radial basis function (nickname kernel)

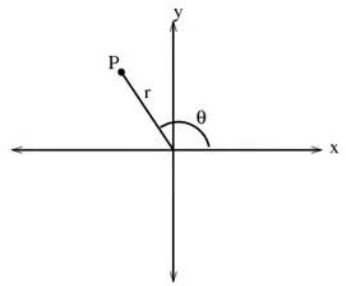


Image from mathinsight.org

Some widely used radial basis functions. It is called radial because it only takes distance into account.

$$\text{Gaussian} \qquad \phi(r) = \exp(-r^2/c^2) \quad (4.4)$$

$$\text{Hardy multiquadric} \qquad \phi(r) = \sqrt{(r^2 + c^2)} \quad (4.5)$$

$$\text{Inverse multiquadric} \qquad \phi(r) = 1/\sqrt{(r^2 + c^2)} \quad (4.6)$$

$$\text{Thin plate spline} \qquad \phi(r) = r^2 \log r. \quad (4.7)$$

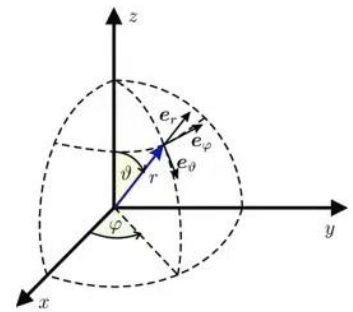
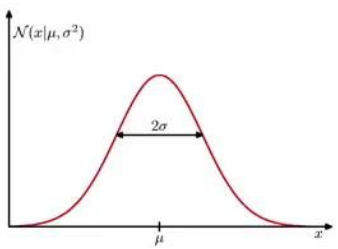


Image from Lotze, Jörg. (2006). Efficient Polarimetric Antenna Radiation Pattern Modeling.

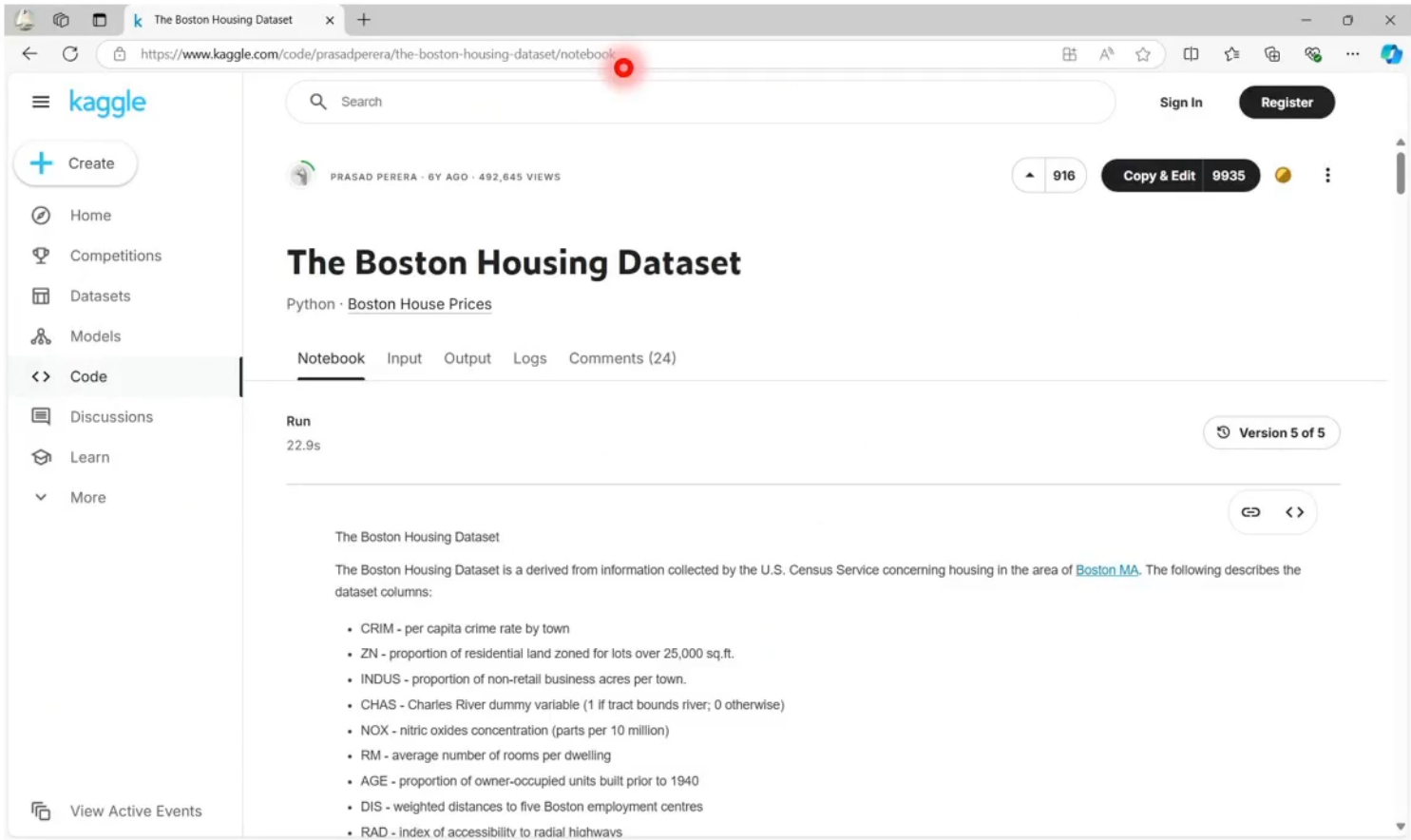


$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Gaussian distribution

Questions?

An example


The Boston Housing Dataset
https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset/notebook
kaggle
Search
Sign In
Register
Create
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
View Active Events
PRASAD PERERA · 6Y AGO · 492,645 VIEWS
916
Copy & Edit 9935
The Boston Housing Dataset
Python · Boston House Prices
Notebook Input Output Logs Comments (24)
Run
22.9s
Version 5 of 5
The Boston Housing Dataset
The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:
CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways

Feature dimension 14, number of instance 506

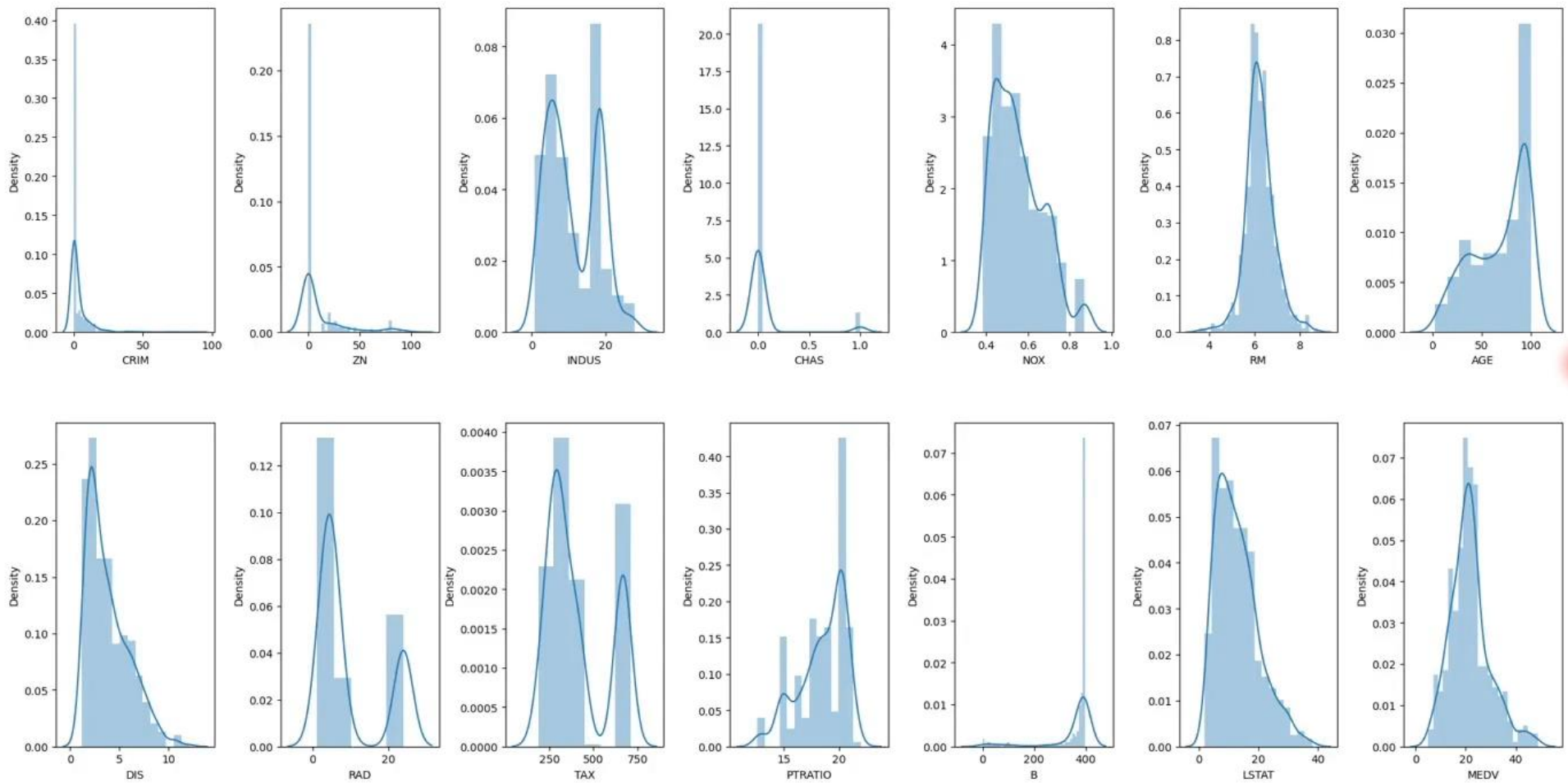
- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centers
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town (This variable has an ethical problem. It is non-invertible and assumes that racial self-segregation had a positive impact on house prices.)
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's **(House Price)**

Its data look like

```
['housing.csv']
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Histogram of the 14 features



Model the house price as a function of other variables

$$\mathbf{f}(\mathbf{x}) = \sum_k \mathbf{w}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|) \qquad (4.3)$$

For one-dimension output

$$PredictedHousePrice = \sum_{k=1}^{506} w_k \Phi(\|\mathbf{x} - \mathbf{x}_k\|) \qquad \mathbf{x}_k = \left\{ \begin{array}{c} CRIM \\ ZN \\ \vdots \\ LSTAT \end{array} \right\}$$

For two-dimension output

$$\left[\begin{array}{c} PredictedHousePrice \\ PredictedLSTAT \end{array} \right] = \sum_{k=1}^{506} \mathbf{w}_k \Phi(\|\mathbf{x} - \mathbf{x}_k\|) \qquad \mathbf{x}_k = \left\{ \begin{array}{c} CRIM \\ ZN \\ \vdots \\ B \end{array} \right\}$$

Solve the $\mathbf{w}_k$ paramters

- If we want our function to exactly interpolate the data values, we solve the linear system of equations

$$\mathbf{f}(\mathbf{x}_k) = \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) = \mathbf{d}_k \qquad (4.8)$$

data point as (4.1)

$$\begin{aligned}
Price_1 &= \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_1 - \mathbf{x}_k\|) \\
Price_2 &= \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_2 - \mathbf{x}_k\|) \\
&\vdots \\
Price_{506} &= \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_{506} - \mathbf{x}_k\|)
\end{aligned}$$

Ill-conditioned

$$\begin{bmatrix} Price_1 \\ Price_1 \\ \vdots \\ Price_{506} \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_{506}\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_{506}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_{506} - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_{506} - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_{506} - \mathbf{x}_{506}\|) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{506} \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{506} \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_{506}\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_{506}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_{506} - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_{506} - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_{506} - \mathbf{x}_{506}\|) \end{bmatrix}^{-1} \begin{bmatrix} Price_1 \\ Price_1 \\ \vdots \\ Price_{506} \end{bmatrix}$$

Ill-conditioned: a small change in a Price variable will cause huge changes of the output values of the $\mathbf{w}$ vector.

Regularizer

- Regularized data approximation problem

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \qquad (4.2)$$

$$E_W = \sum_k \|\mathbf{w}_k\|^p \qquad (4.9)$$

Given $\mathbf{d}_k, \mathbf{x}_k, \lambda$, minimize

$$E(\{w_k\}) = E_D + \lambda E_W \qquad (4.10)$$

$$= \sum_k \| \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k\|^2 + \lambda \sum_k \|\mathbf{w}_k\|^p \qquad (4.11)$$

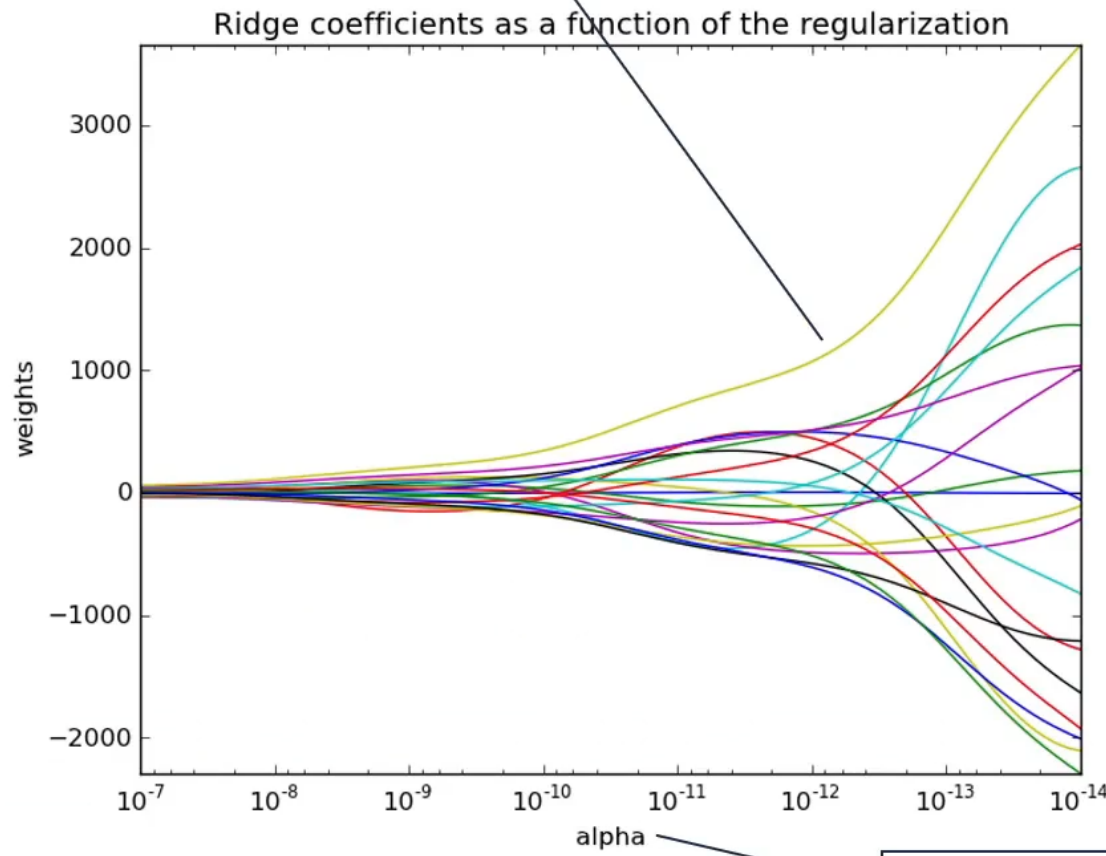
P	Machine Learning	Neural Network
2	Ridge regression	Weight decay (encourage weights to decay to zero)
1	Lasso (least absolute shrinkage and selection operator)	

Image from stock.adobe.com

Ridge

Image from YouTube *Stoos Ridge Hike | Switzerland* by Alex Kay Travels

As λ decreases, w_i increases.
ridge-like shape



same as λ in our textbook

Image from fa.bianp.net

Kernel regression

- If we just want to interpolate a set of data, we can use

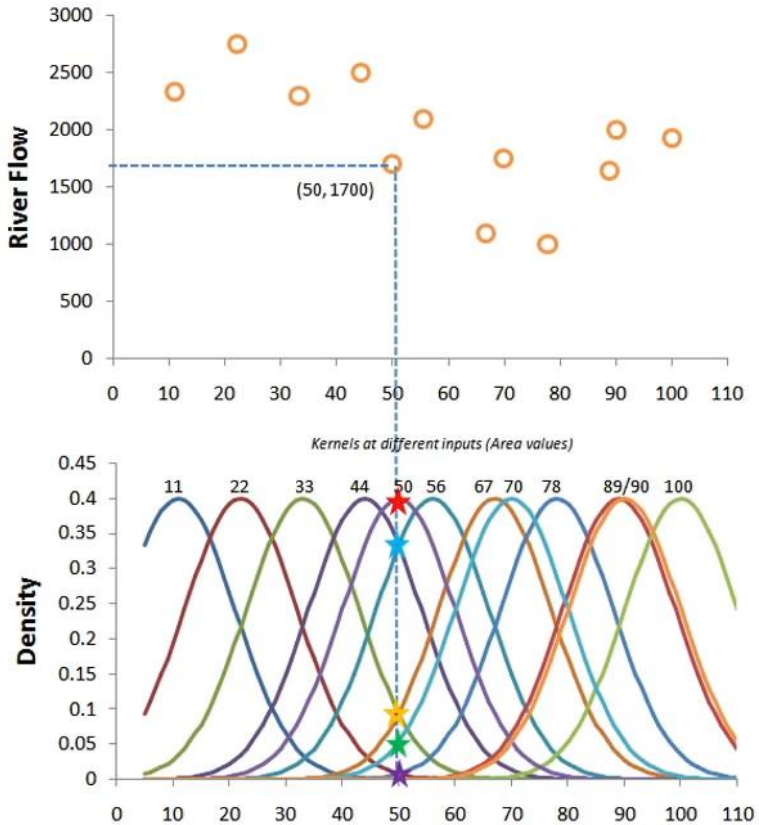
$$\mathbf{f}(\mathbf{x}) = \frac{\sum_k \mathbf{d}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|)}{\sum_l \phi(\|\mathbf{x} - \mathbf{x}_l\|)} \tag{4.12}$$

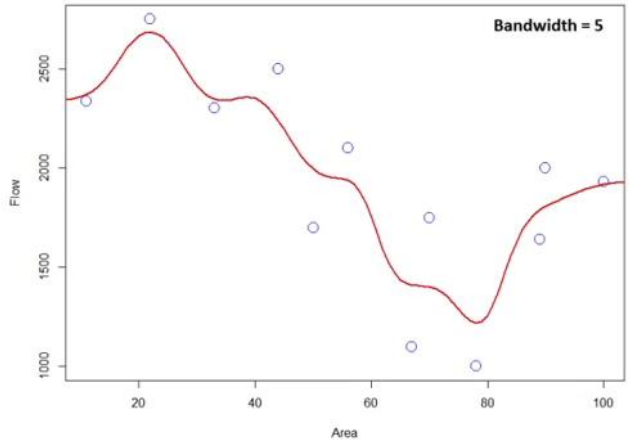
- There is no parameter to tune. It is just the weighted average of the data $\mathbf{d}_k$, and weights are decided by the radial basis function (kernel).

$$\phi'_k(\mathbf{x}) = \frac{\phi(\|\mathbf{x} - \mathbf{x}_k\|)}{\sum_l \phi(\|\mathbf{x} - \mathbf{x}_l\|)} \tag{4.13}$$

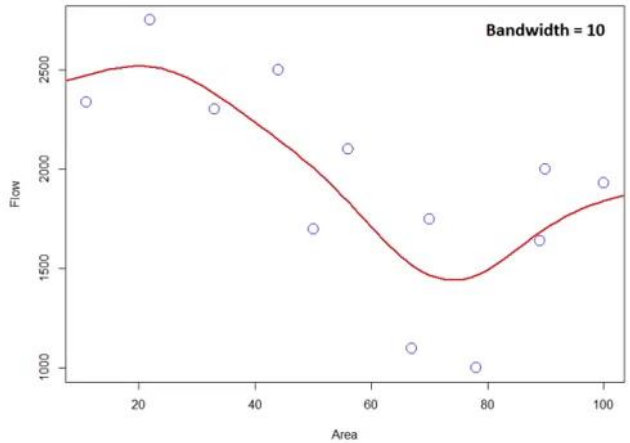
Example of Gaussian Kernal Regression

Gaussian $$\phi(r) = \exp(-r^2/c^2) \quad (4.4)$$

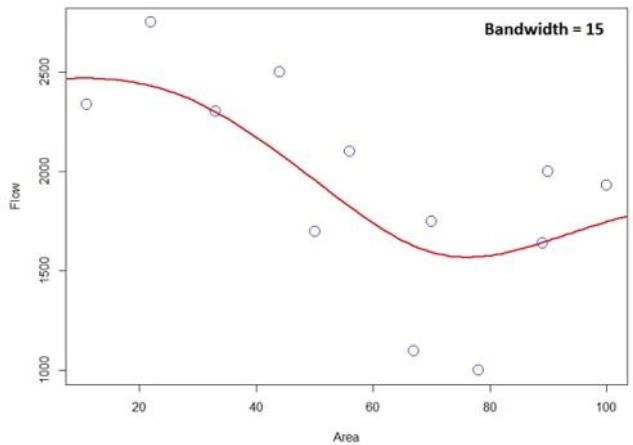




small c



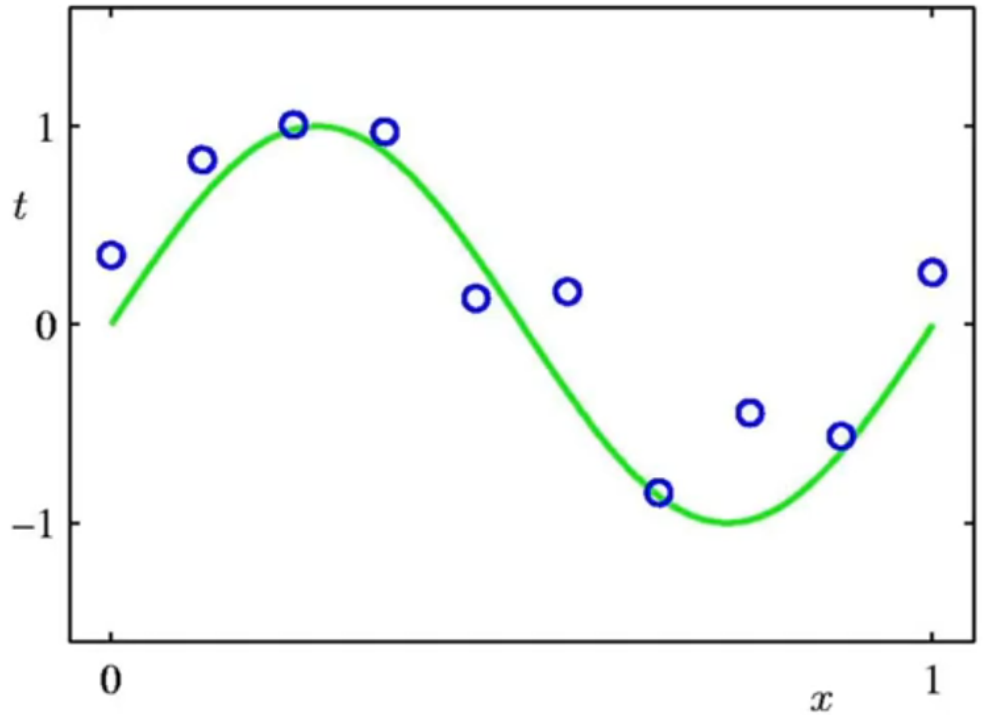
medium c



large c

Images from towardsdatascience.com

Polynomial Curve Fitting



Green: the ground truth curve

$$\sin(2\pi x)$$

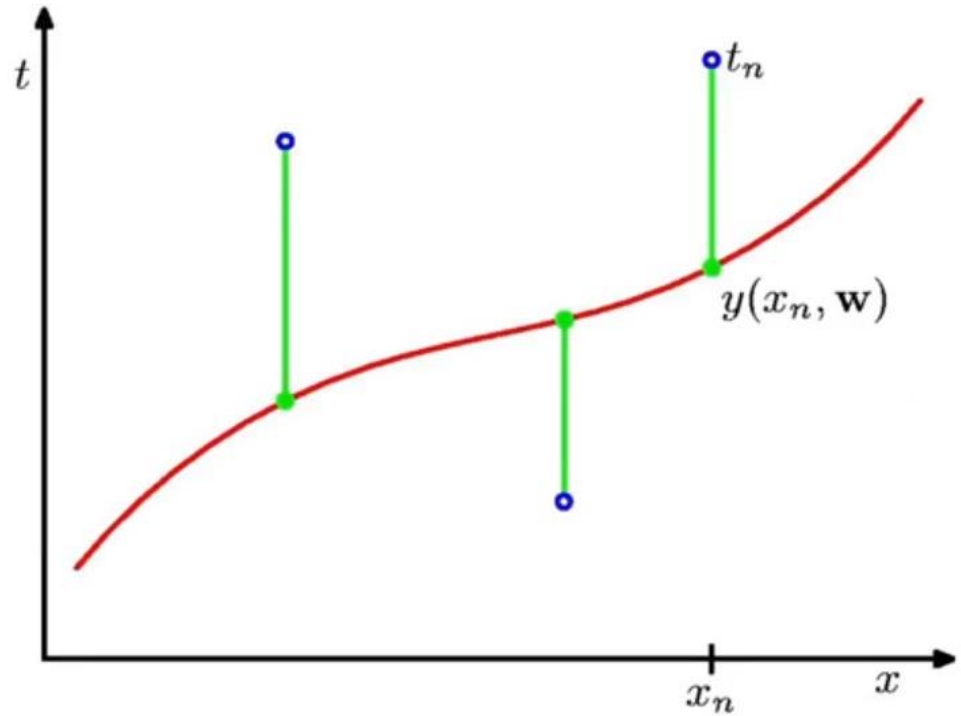
blue points: observation values with noise
number of observations: 10

We use a polynomial function to fit the data

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Images from *Pattern Recognition and Machine Learning* by C. Bishop

Sum-of-Squares Error Function



To determine the weight vector $\mathbf{w}$, we minimize the error function.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Images from *Pattern Recognition and Machine Learning* by C. Bishop

0^{th} Order Polynomial (M=0, N=10)

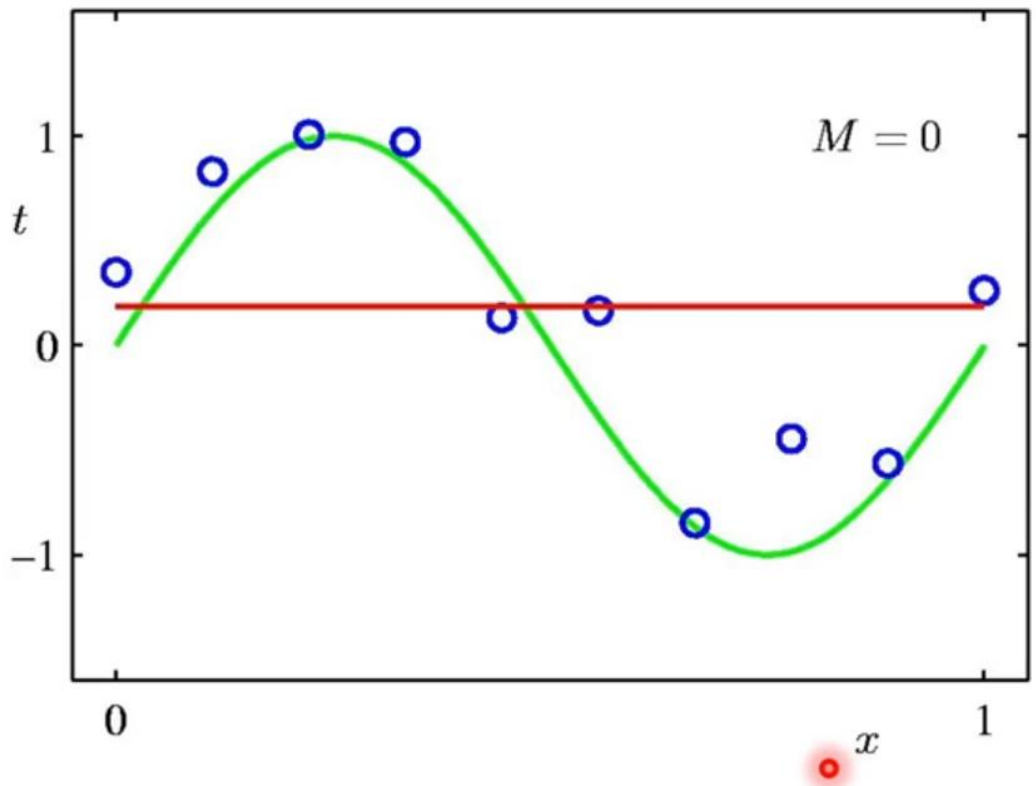
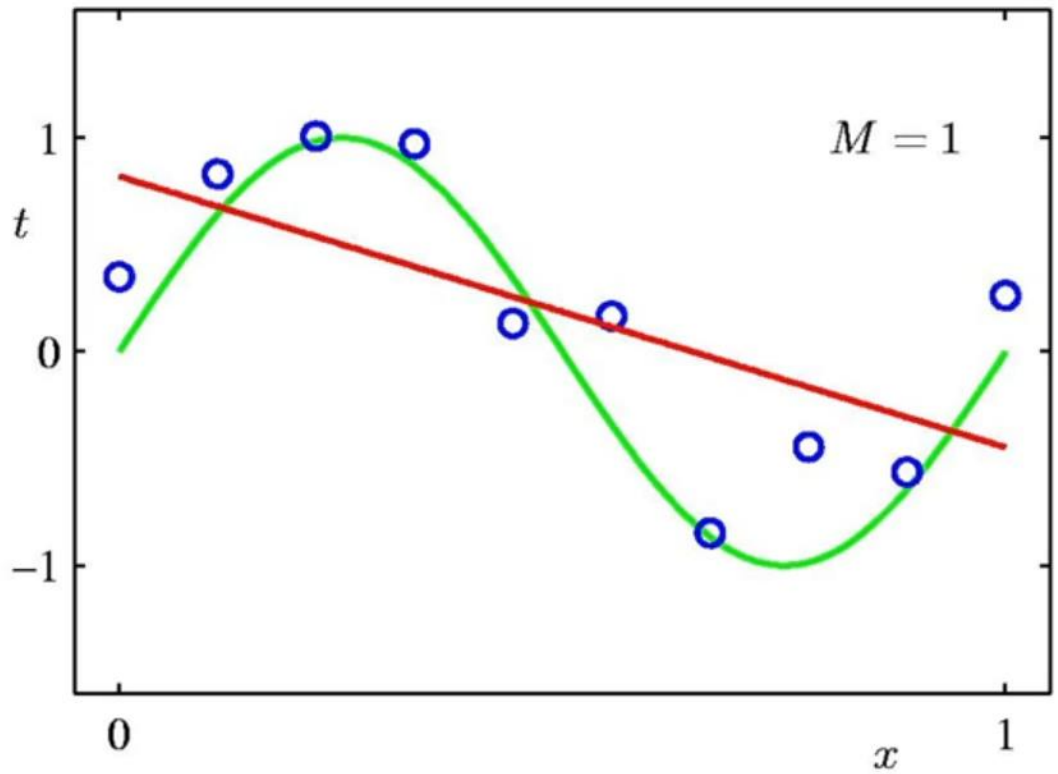


Figure 4.3

1st Order Polynomial (M=1 , N=10)



When M=0 or 1, we get large training error values because our model does not fit the ground truth model.

Figure 4.3

3rd Order Polynomial (M=3, N=10)

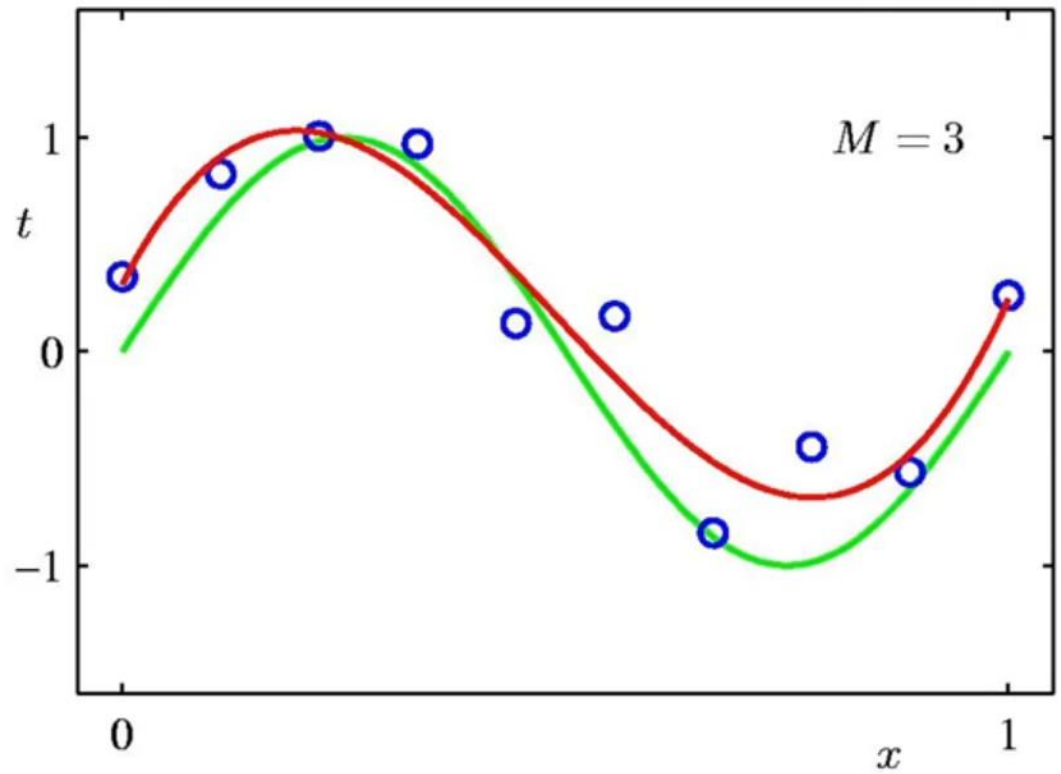
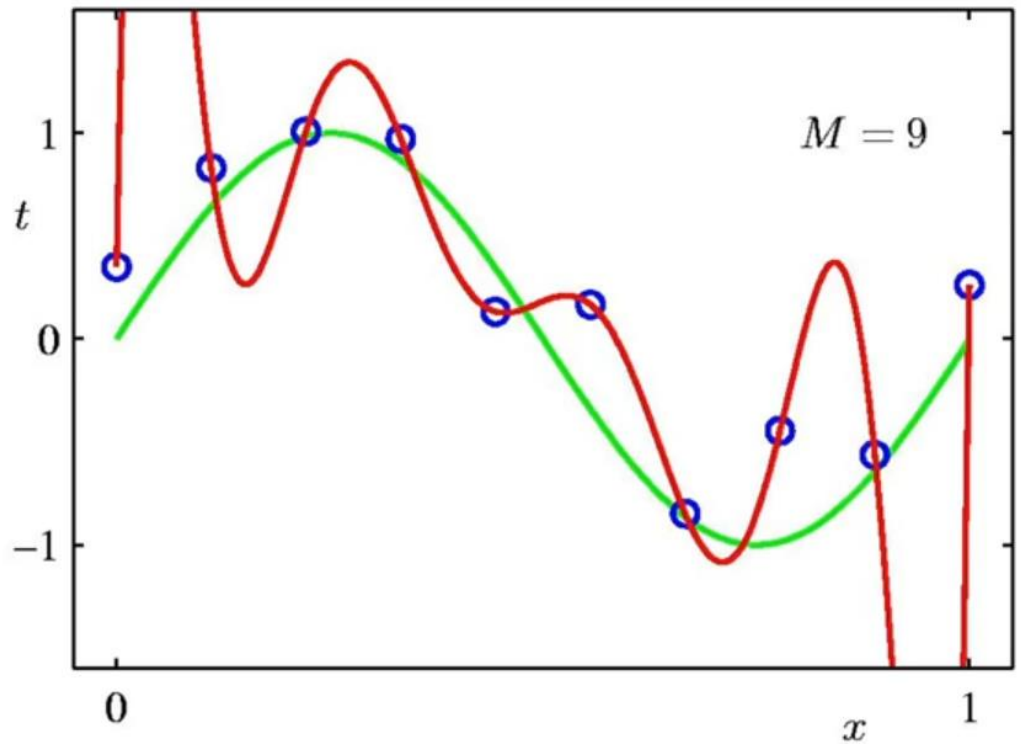


Figure 4.3

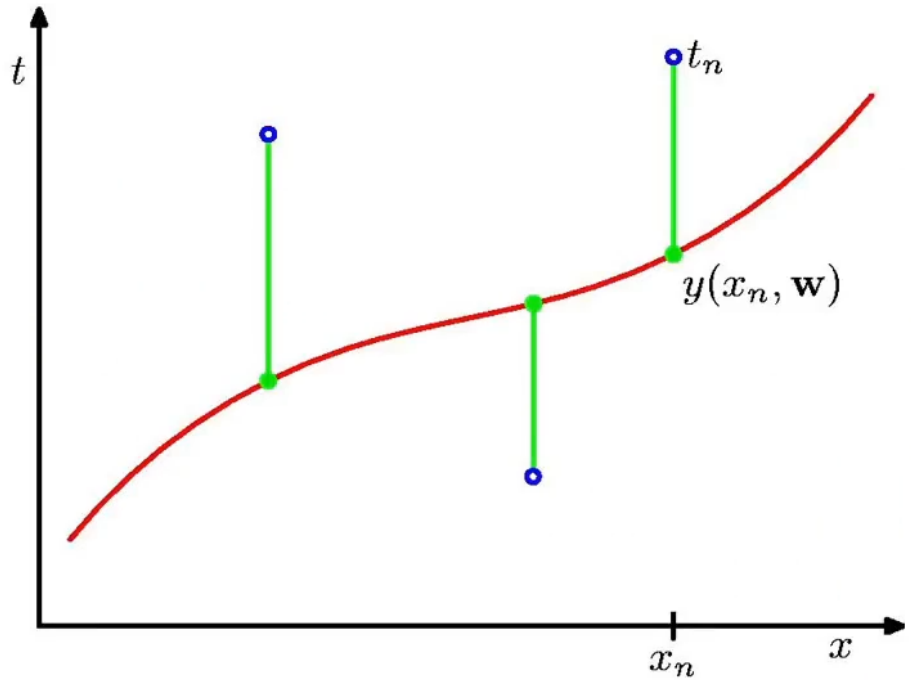
9th Order Polynomial (M=9, N=10)



When M = 9, our training error reduces to 0, but the learned model is highly unlike the ground truth model.

Figure 4.3

Sum-of-Squares Error Function



To determine the weight vector $\mathbf{w}$, we minimize the error function.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Images from *Pattern Recognition and Machine Learning* by C. Bishop

Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Those parameters are too large to make sense.

Overfitting and Underfitting

- If we split our dataset into two subsets: training and validation.
- We have a model containing hyperparameters (M, λ) and parameters $(w_0, w_1, \dots, w_M)$.
- Under various hyperparameter values, we use the training set to learn the optimized parameters $(w_0, w_1, \dots, w_M)$.

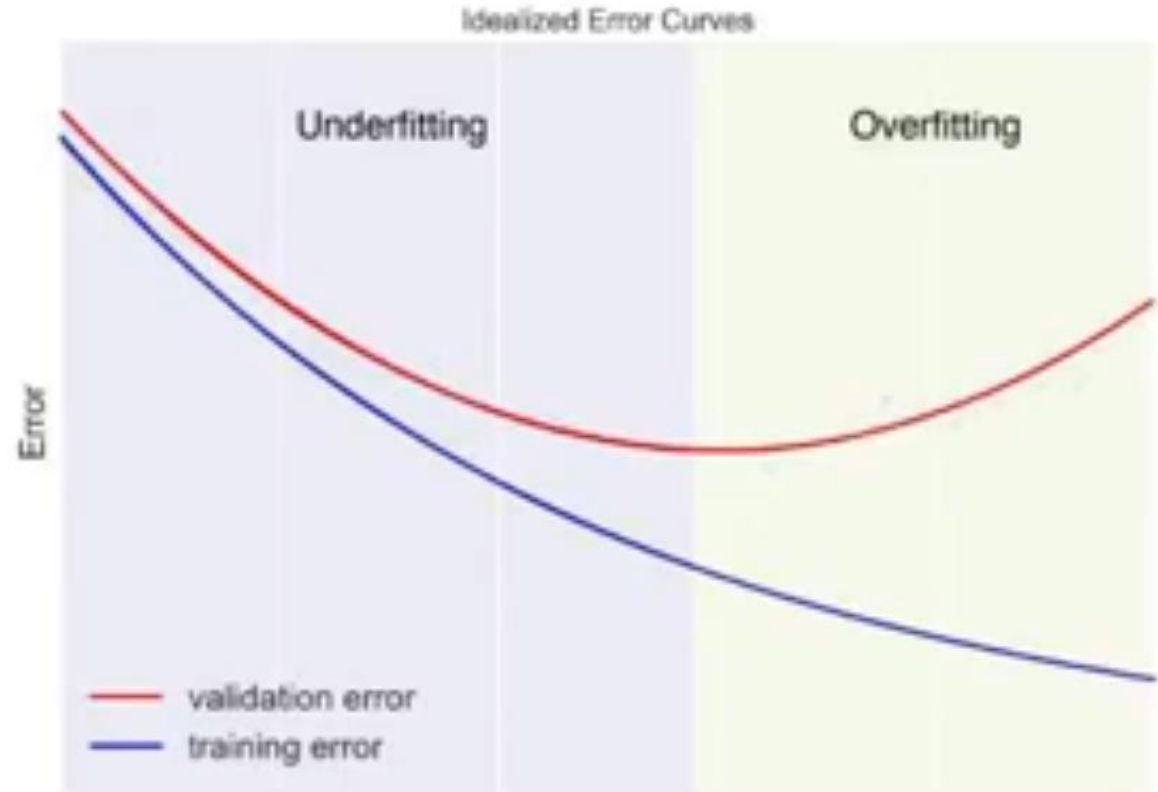


Figure 4.5

Cross-Validation

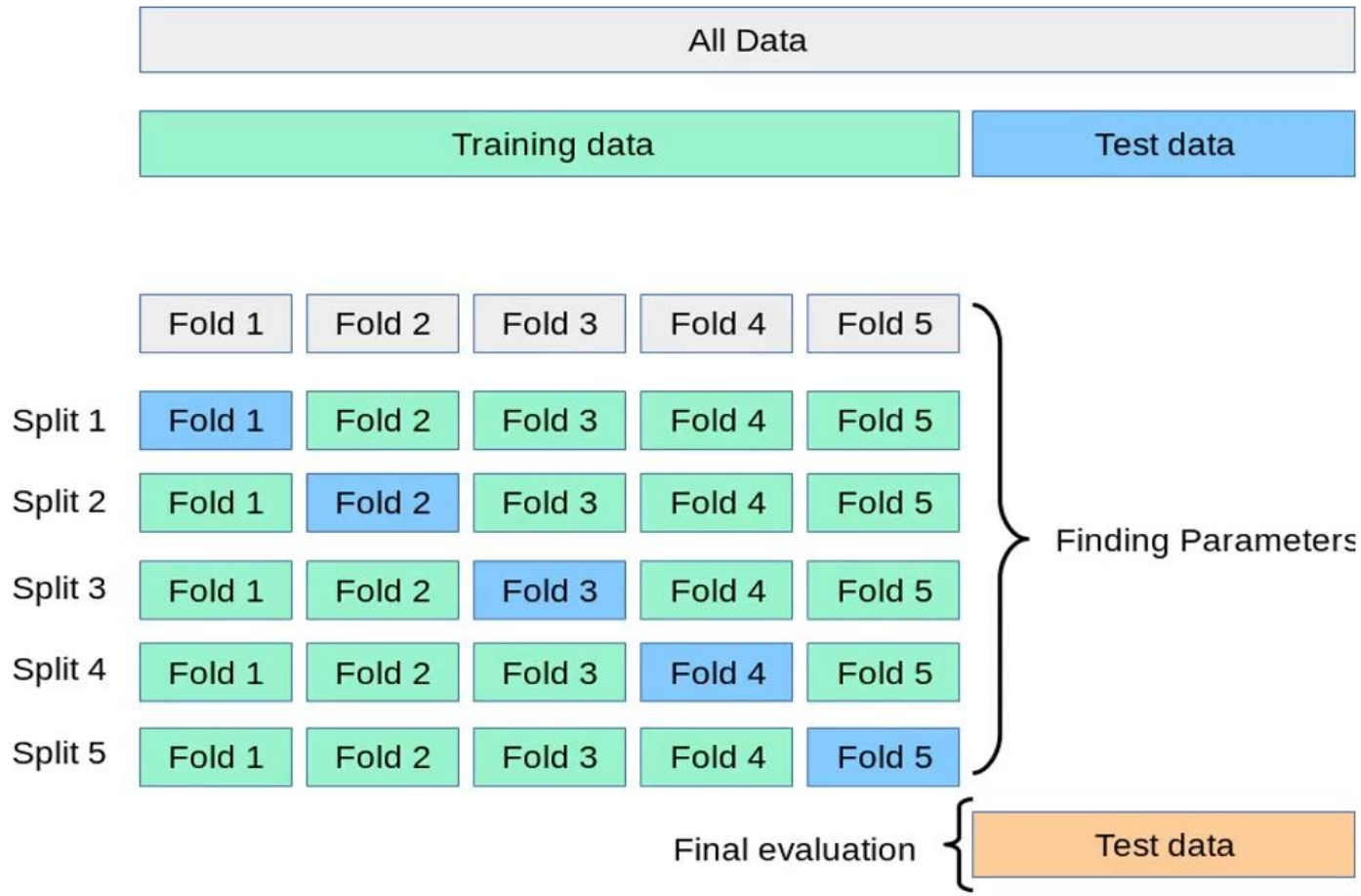
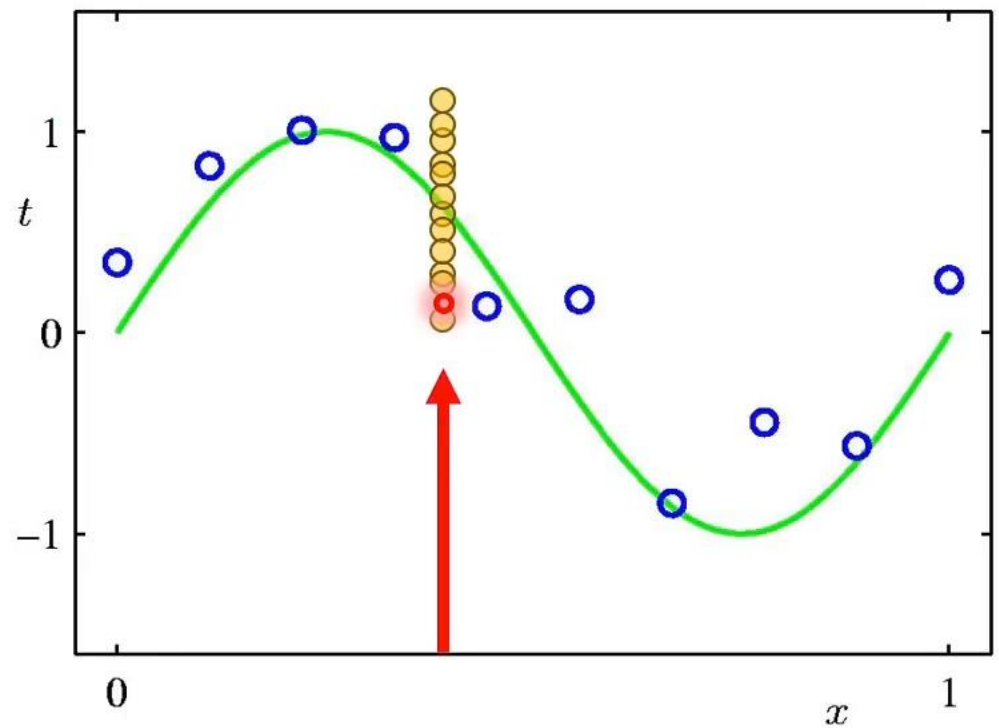


Figure from scikit-learn.org

Bias-Variance tradeoff

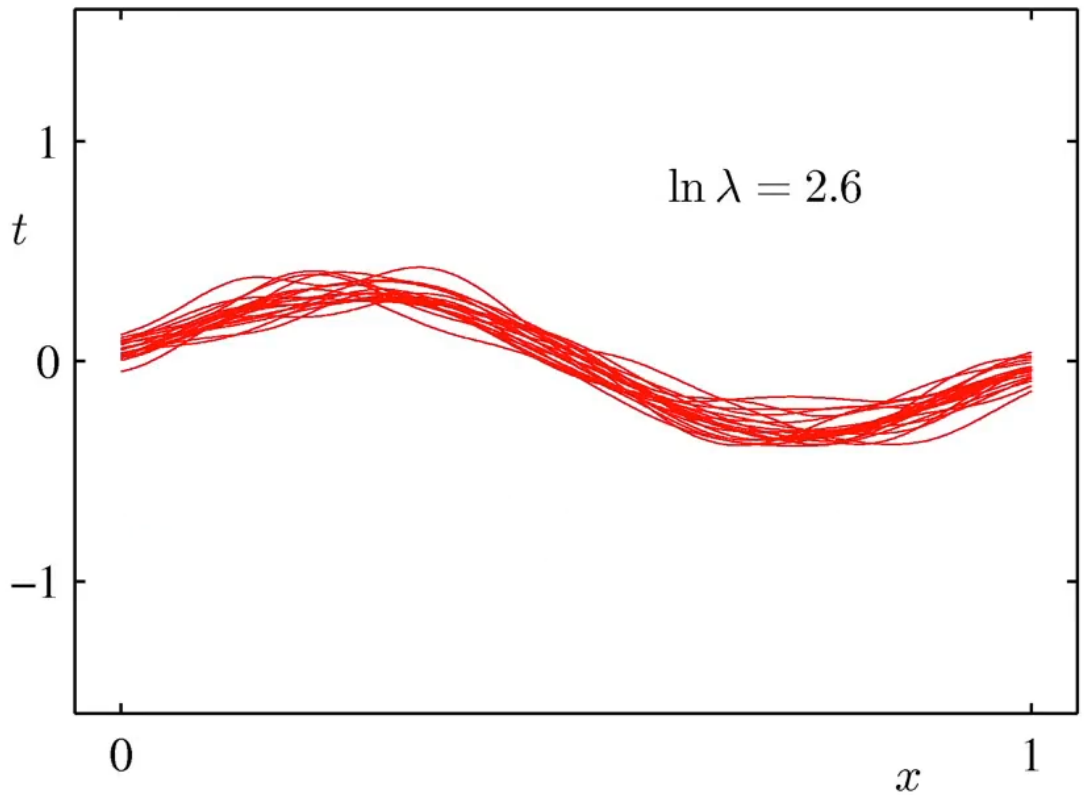
- $\mathbf{x}$: input vector
- t: observed value, which has a distribution based on $\mathbf{x}$
- $h(\mathbf{x})$: Expectation of t when input is $\mathbf{x}$

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t$$ (3.36) Bishop's book



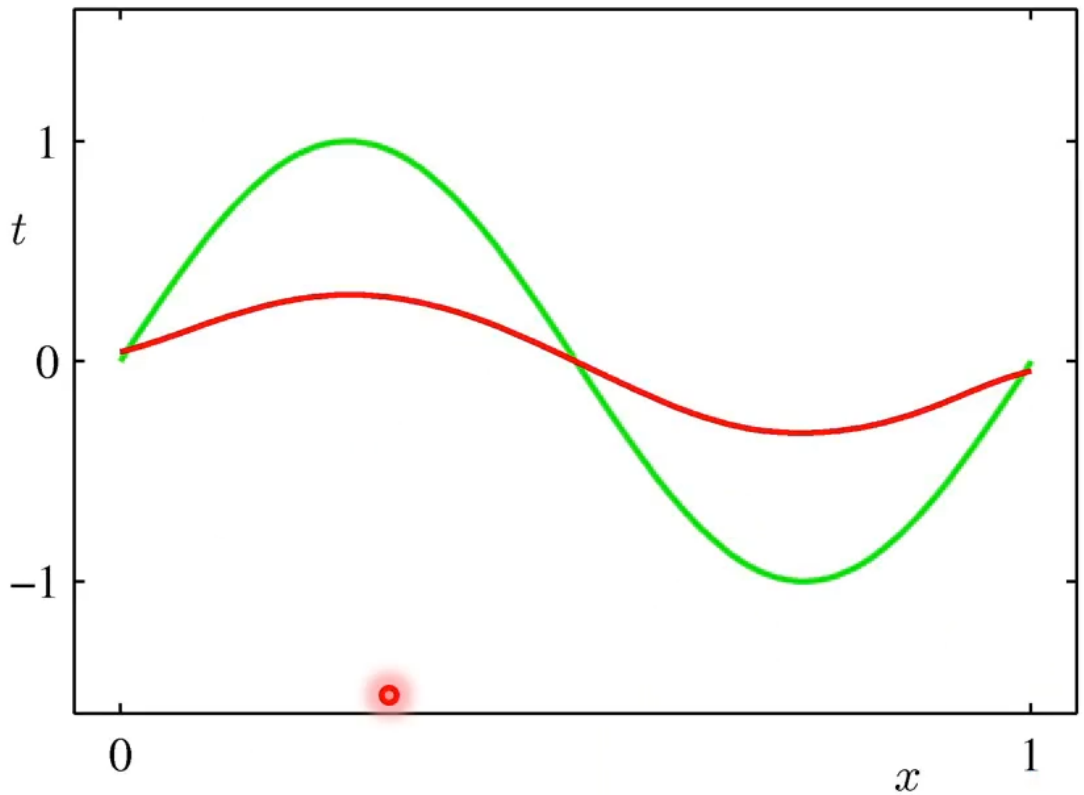
Gaussian basis function, 100 data sets, 25 points per set, $\lambda = e^{2.6}$

only 20 sets are shown for clarity



Low variance

mean of the 100 $y(\mathbf{x})$ functions



High bias

Figure 4.6 [53]

- Definition of an average loss

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x}))p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \qquad \text{(1.86) Bishop}$$

- If the loss function is a squared term $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$,the average loss becomes

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \qquad \text{(1.87) Bishop}$$

- We want to choose $y(\mathbf{x})$ to minimize $\mathbb{E}[L]$, which occurs when its derivative is 0

$$\frac{\delta \mathbb{E}[L]}{\delta y(\mathbf{x})} = 2\int \{y(\mathbf{x}) - t\}p(\mathbf{x}, t)\,\mathrm{d}t = 0 \qquad \text{(1.88) Bishop}$$

‘Old Faithful’ data set

- 272 measurements of the eruption of the Old Faithful geyser at Yellowstone National Park in the USA

image from tw.traveleredge.com

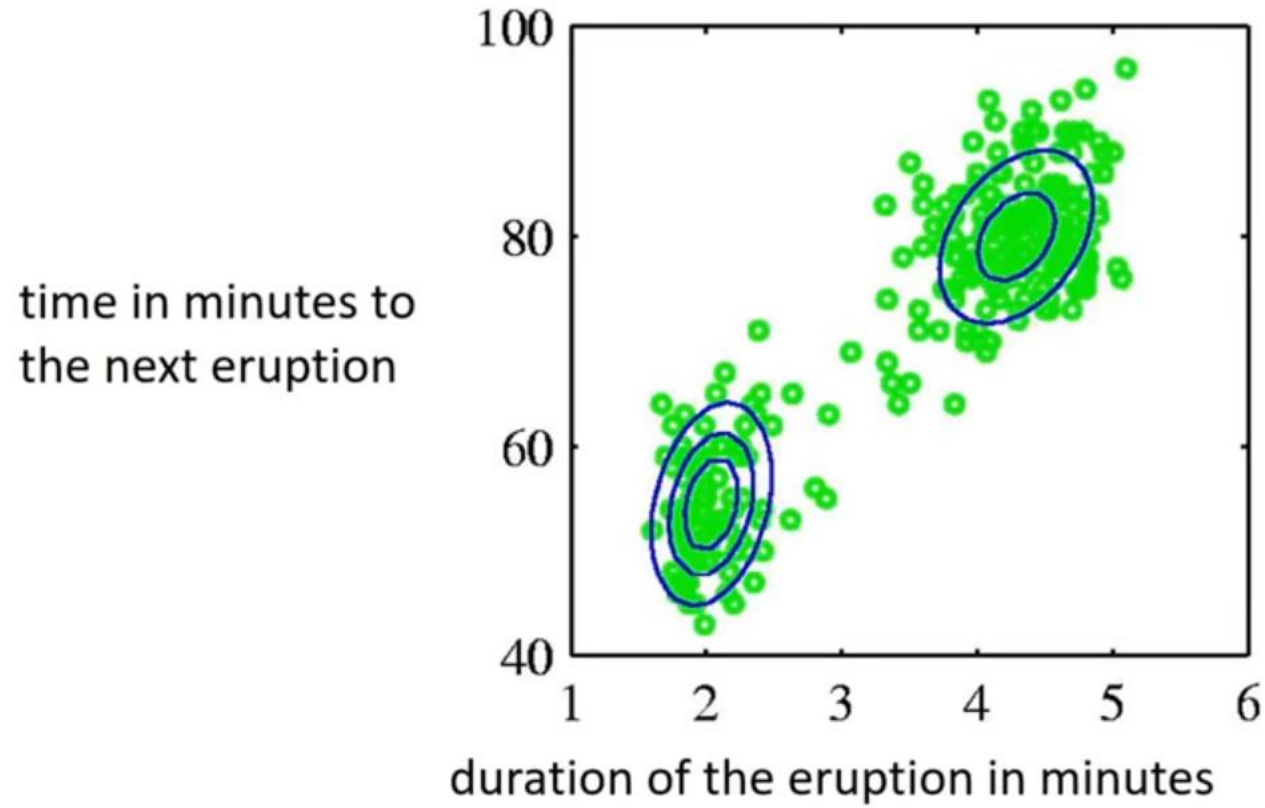


Figure 2.21
Bishop

Illustration

- If the loss function is a squared term $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$, the value of $y(\mathbf{x})$ should be the mean of t under the conditional distribution $p(t|\mathbf{x})$.

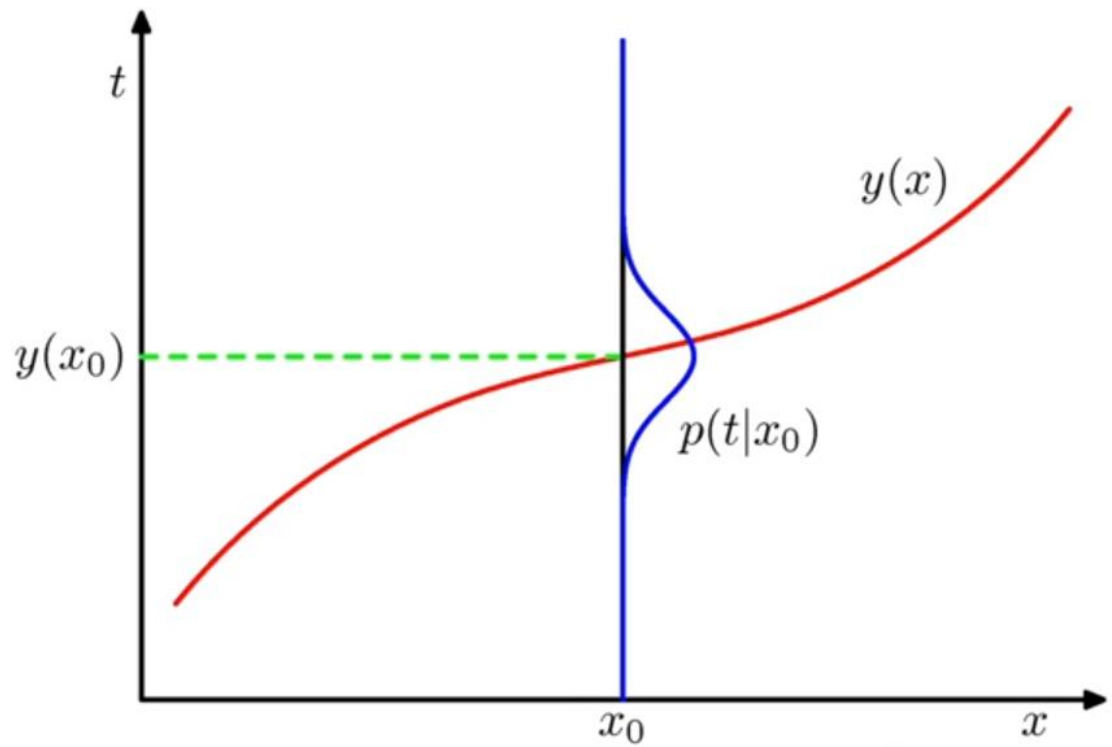


Figure 1.28
Bishop

3rd Order Polynomial (M=3, N=10)

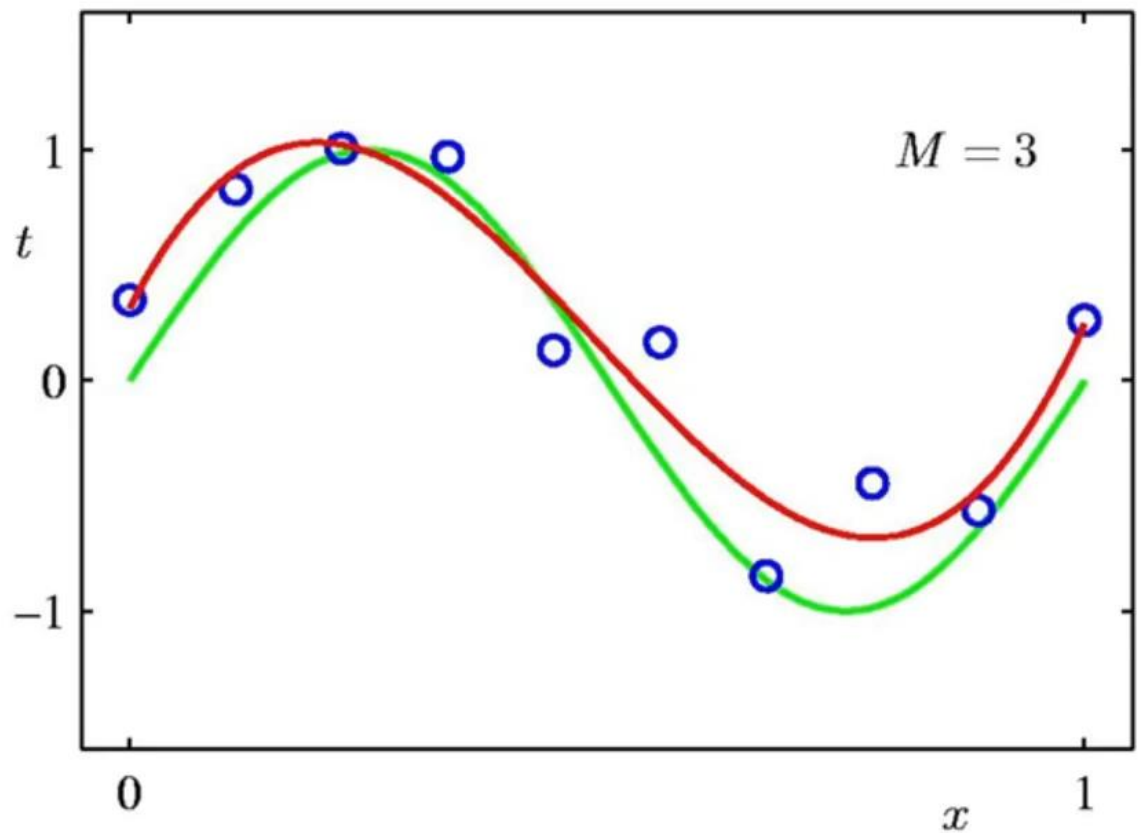
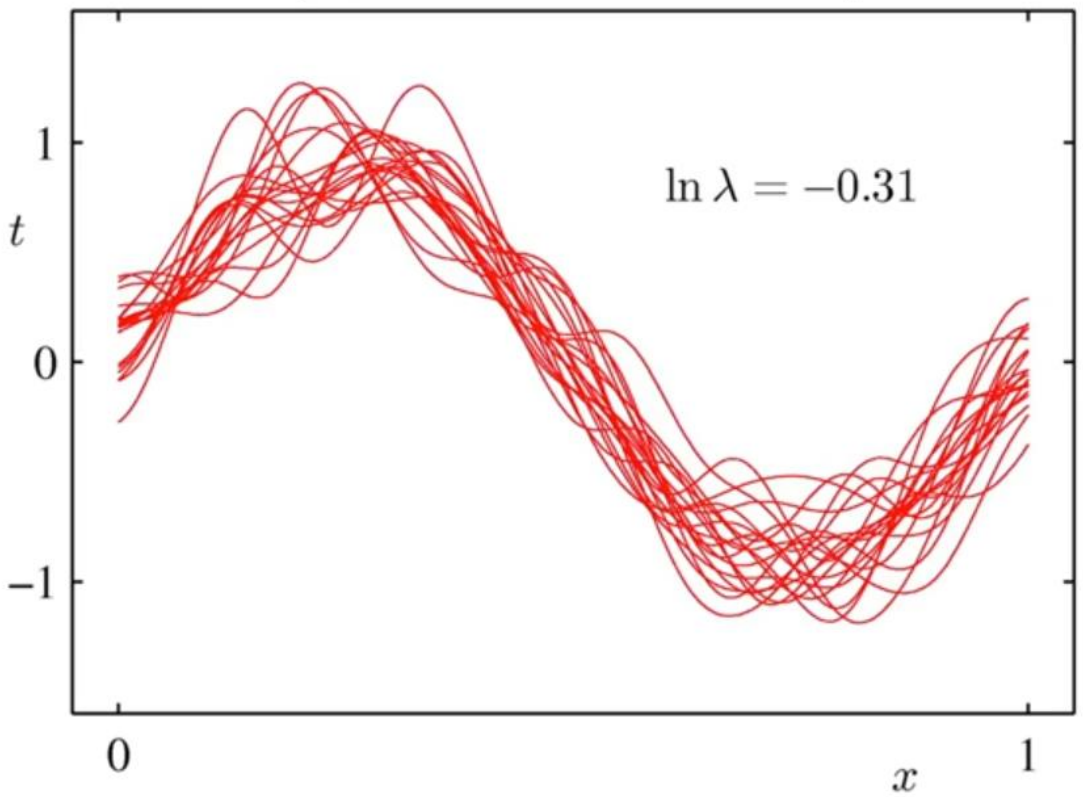


Figure 4.3

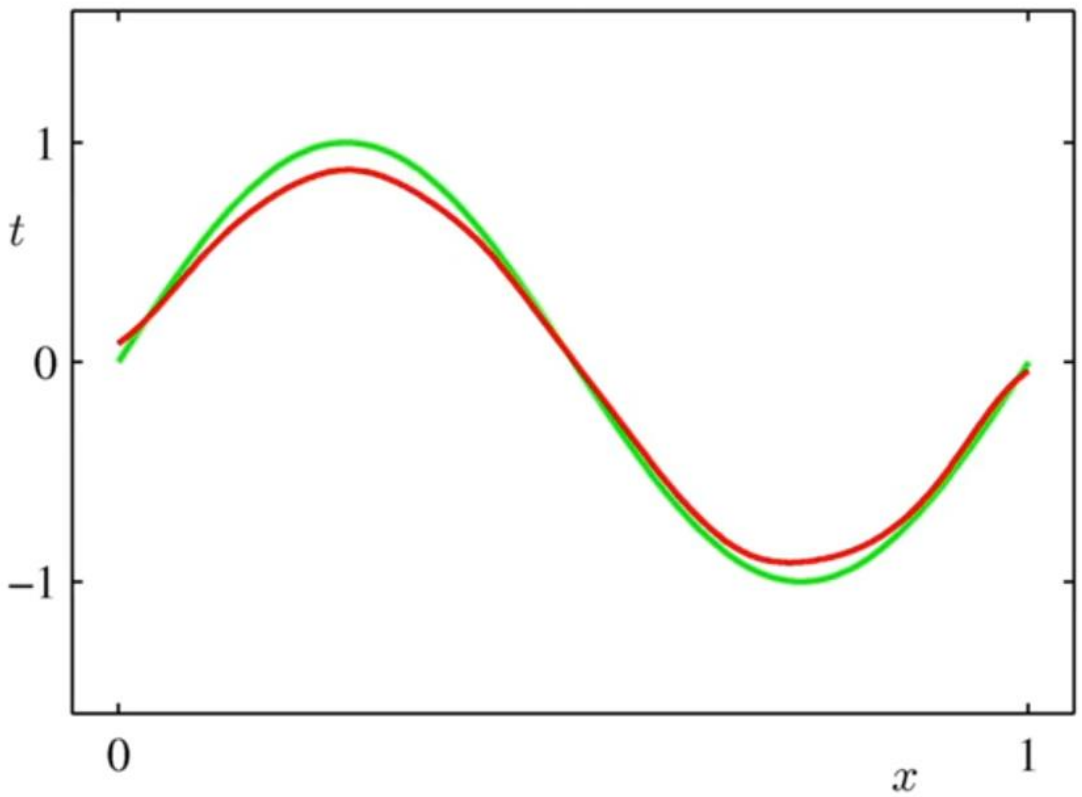
$$\lambda = e^{-0.31}$$

only 20 sets are shown for clarity



Medium variance

mean of the 100 $y(\mathbf{x})$ functions



Medium bias

Figure 3.5
Bishop

Regularizer

- Regularized data approximation problem

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \qquad (4.2)$$

$$E_W = \sum_k \|\mathbf{w}_k\|^p \qquad (4.9)$$

Given $\mathbf{d}_k, \mathbf{x}_k, \lambda$, minimize

$$E(\{w_k\}) = E_D + \lambda E_W \qquad (4.10)$$

$$= \sum_k \| \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k\|^2 + \lambda \sum_k \|\mathbf{w}_k\|^p \qquad (4.11)$$

P	Machine Learning	Neural Network
2	Ridge regression	Weight decay (encourage weights to decay to zero)
1	Lasso (least absolute shrinkage and selection operator)	

Image from stock.adobe.com

Robust Loss

- In (4.9) $p = 1$ and 2 are widely used. Any other choice? Barron proposes

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right) \quad \text{(1) Barron}$$

$$\lim_{\alpha \to 2} f(x, \alpha, c) = \frac{1}{2}(x/c)^2 \quad \text{L2 loss} \quad \text{(2) Barron}$$

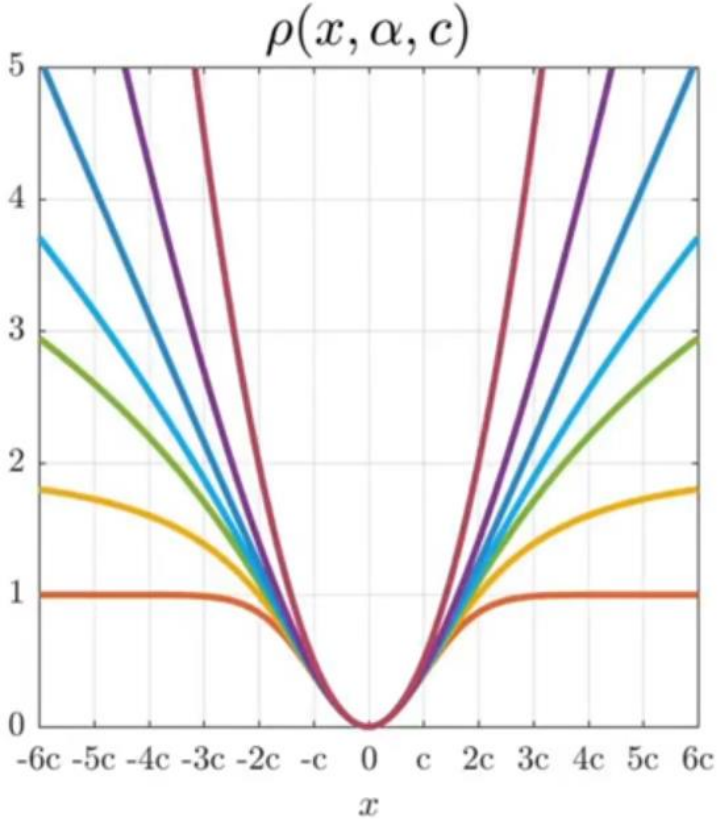
$$f(x, 1, c) = \sqrt{(x/c)^2 + 1} - 1 \quad \text{L1 loss} \quad \text{(3) Barron}$$

$$\lim_{\alpha \to 0} f(x, \alpha, c) = \log\left(\frac{1}{2}(x/c)^2 + 1\right) \quad \text{Cauchy loss} \quad \text{(5) Barron}$$

$$f(x, -2, c) = \frac{2(x/c)^2}{(x/c)^2 + 4} \quad \text{Geman-McClure loss} \quad \text{(6) Barron}$$

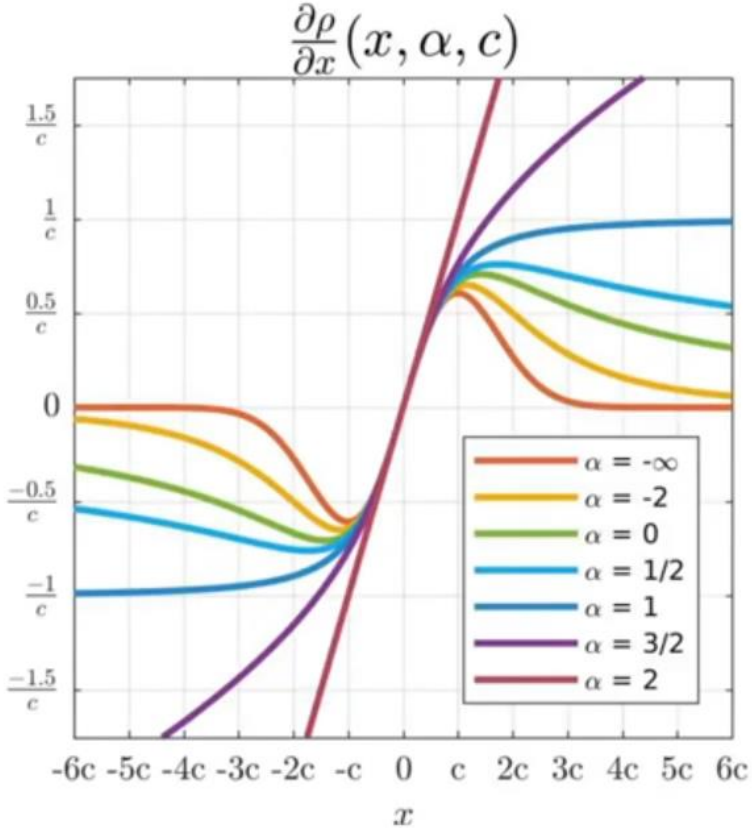
$$\lim_{\alpha \to -\infty} f(x, \alpha, c) = 1 - \exp\left(-\frac{1}{2}(x/c)^2\right) \quad \text{Welsch loss} \quad \text{(7) Barron}$$

BARRON, Jonathan. A General and Adaptive Robust Loss Function. CVPR 2019.



$$\rho\left(x,\alpha,c\right)=\begin{cases}\frac{1}{2}\left(x/c\right)^2 & \text{if } \alpha=2\\ \log\left(\frac{1}{2}\left(x/c\right)^2+1\right) & \text{if } \alpha=0\\ 1-\exp\left(-\frac{1}{2}\left(x/c\right)^2\right) & \text{if } \alpha=-\infty\\ \frac{|\alpha-2|}{\alpha}\left(\left(\frac{\left(x/c\right)^2}{|\alpha-2|}+1\right)^{\alpha/2}-1\right) & \text{otherwise}\end{cases}$$

(8) Barron



$$\frac{\partial \rho}{\partial x}\left(x,\alpha,c\right)=\begin{cases}\frac{x}{c^2} & \text{if } \alpha=2\\ \frac{2x}{x^2+2c^2} & \text{if } \alpha=0\\ \frac{x}{c^2}\exp\left(-\frac{1}{2}\left(x/c\right)^2\right) & \text{if } \alpha=-\infty\\ \frac{x}{c^2}\left(\frac{\left(x/c\right)^2}{|\alpha-2|}+1\right)^{\left(\alpha/2-1\right)} & \text{otherwise}\end{cases}$$

(9) Barron

Figure 4.7 [57]

Comparison with Student's t-distribution

	Parameters	Normal (Gaussian)	Cauchy
Student's t-distribution	μ, λ, ν	$\nu \to \infty$	$\nu = 1$
Barron's distribution	μ, α, c	$\alpha = 2$	$\alpha = 0$

Barron's experiment: create a variational autoencoder

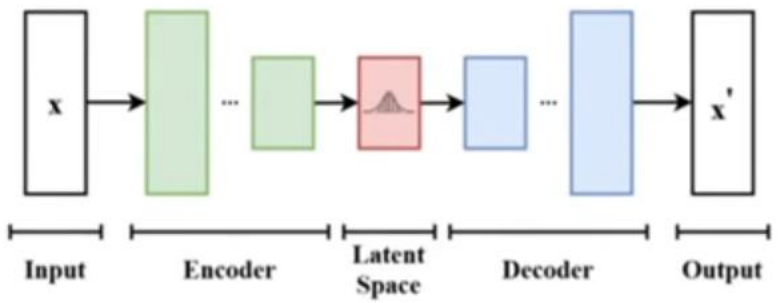


Figure from wikipedia

Large-scale CelebFaces Attributes (CelebA) Dataset

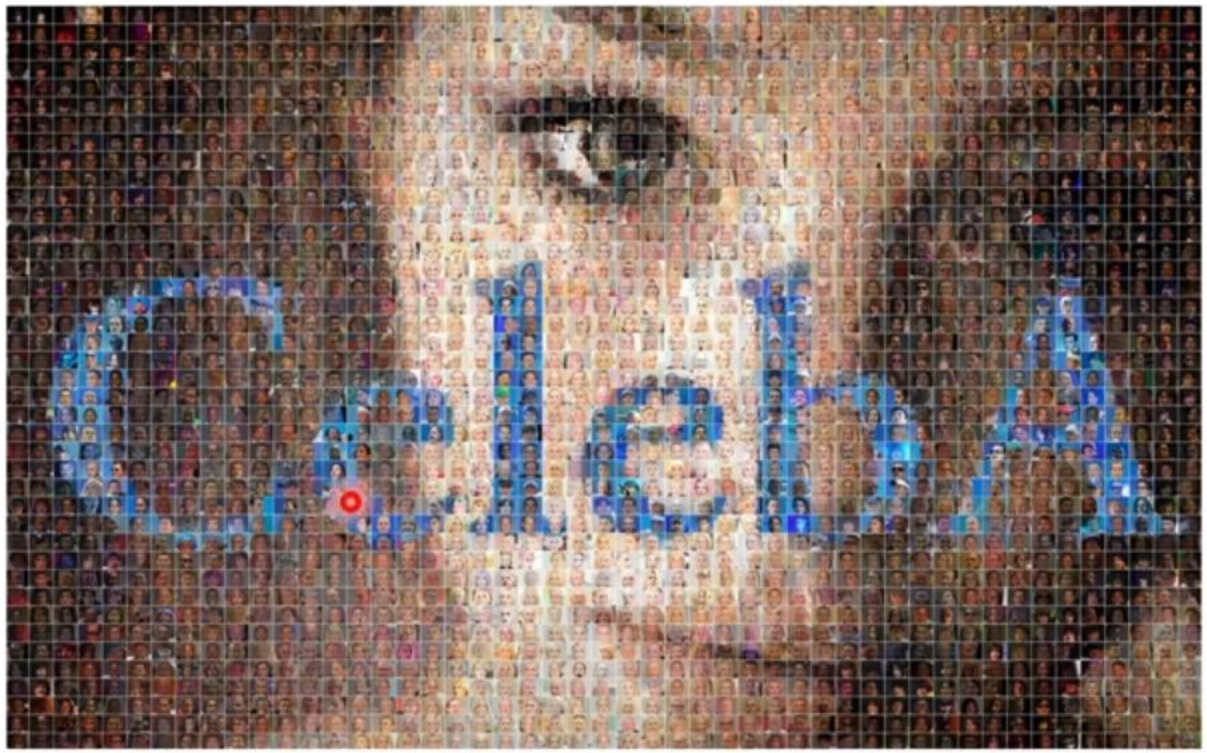


Image from mmlab.ie.cuhk.edu.hk

10,177 number of identities,
202,599 number of face images, and
5 landmark locations,
40 binary attributes annotations per image.

Results

Table 1
Barron

	Normal	Cauchy	t-dist.	Ours
Pixels + RGB	8,662	9,602	10,177	10,240
DCT + YUV	31,837	31,295	32,804	32,806
Wavelets + YUV	31,505	35,779	36,373	36,316

Validation set evidence lower bound (ELBO), higher is better.

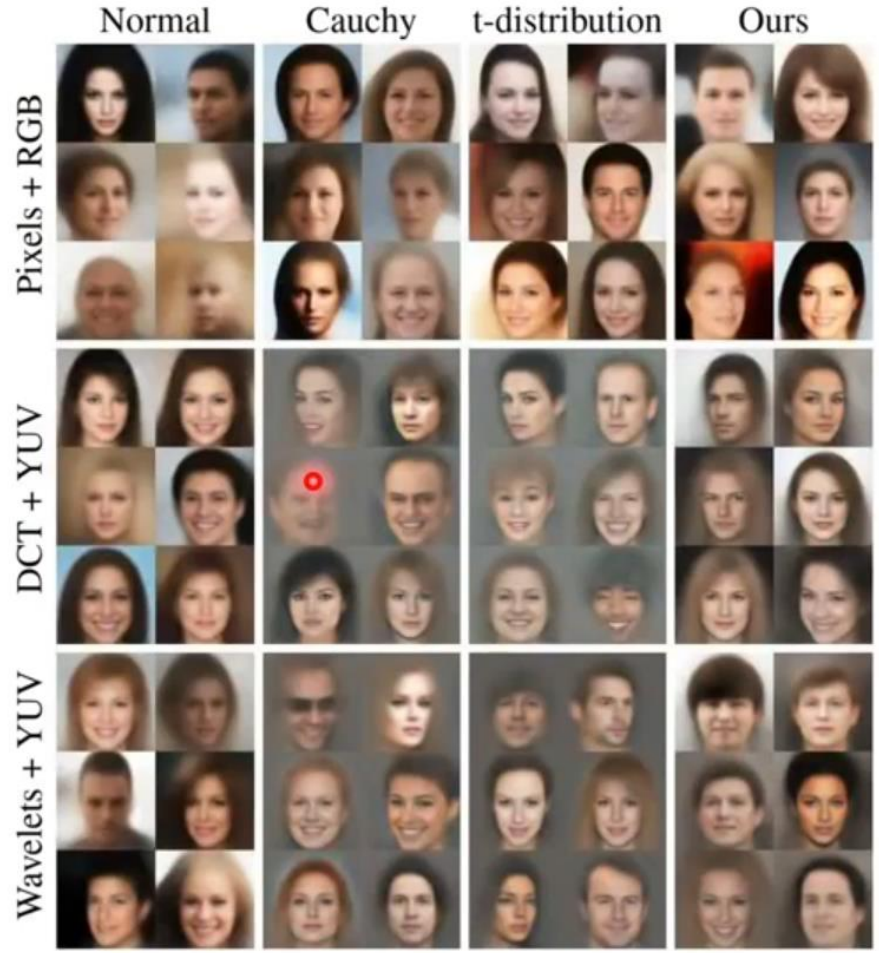


Figure 3
Barron

Colorization task
Scattered data interpolation problem

(a)

Input grayscale image, and color scribbles

(b)

Output result

(c)

Ground truth image

Figure 4.10

YUV Color Space

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{aligned} Y &\in [0, 255] \\ U &\in [0, 255] \\ V &\in [0, 255] \end{aligned}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.00093 & 1.401687 \\ 1 & -0.3437 & -0.71417 \\ 1 & 1.77216 & 0.00099 \end{bmatrix} \begin{bmatrix} Y \\ U - 128 \\ V - 128 \end{bmatrix}$$

Equations from wikipedia

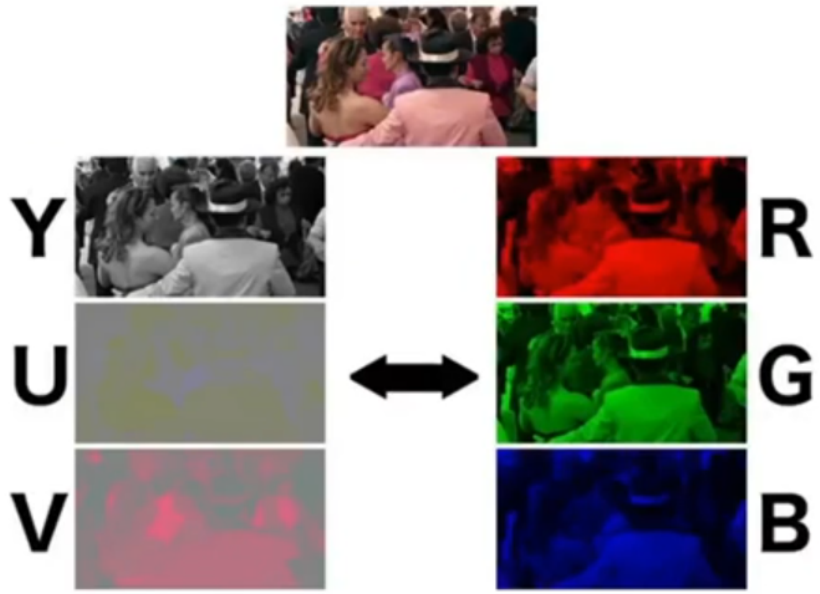


image from www.retrosix.wiki

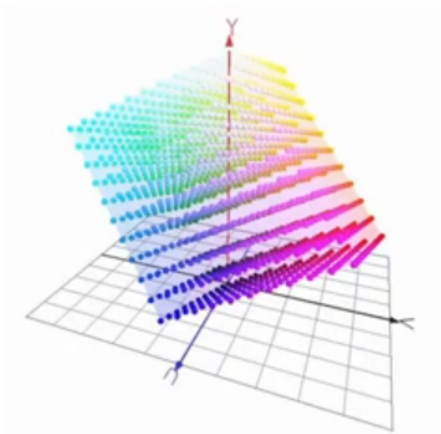


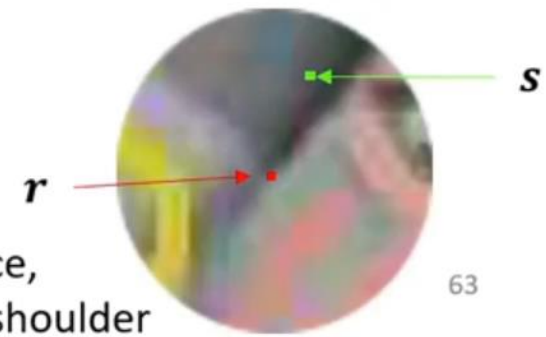
image from Philippe. “Color Space Transformations.” (2006).

Colorization using Optimization (Levin et al. ACM TOG 2004)

- Let the input grayscale image be Y, and the color images be U, V.
- Let $\boldsymbol{r}$, $\boldsymbol{s}$ denote pixels on the image.
- The given scribbles assign some values of U, V.
- The remaining U value are assigned by minimizing

$$J(U) = \sum_{\mathbf{r}} \left(U(\mathbf{r}) - \sum_{\mathbf{s} \in N(\mathbf{r})} w_{\mathbf{rs}} U(\mathbf{s}) \right)^2 \quad (1) \text{ Levin}$$

$N(\boldsymbol{r})$ means a small neighborhood of $\boldsymbol{r}$
$w_{\boldsymbol{rs}}$ is a weight coefficient



corner among face,
pillow, and right shoulder

Weight Coefficient

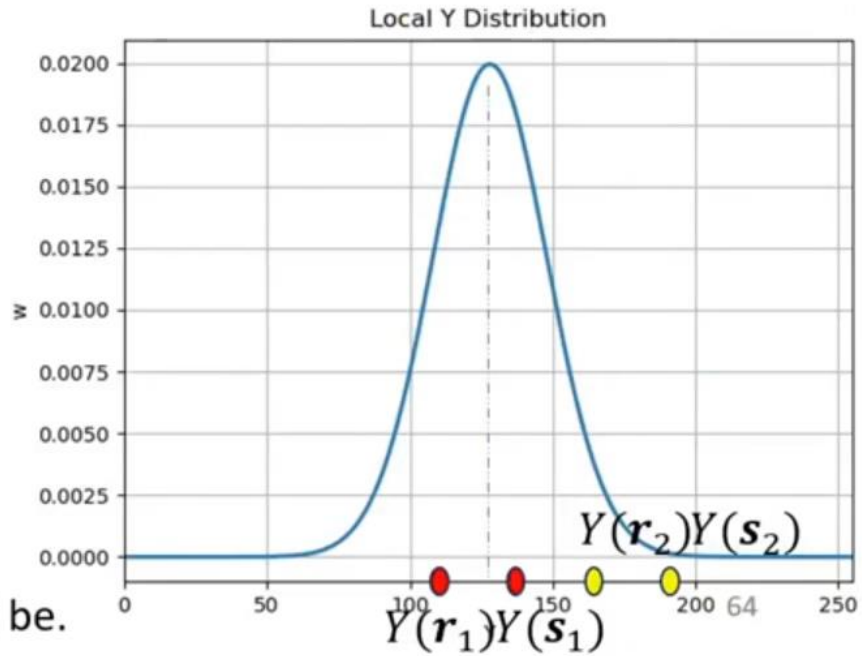
- $Y(\boldsymbol{r})$ and $Y(\boldsymbol{s})$ are similar, $U(\boldsymbol{r})$ and $U(\boldsymbol{s})$ should be similar in terms of their
  - raw values

$$w_{\mathbf{rs}} \propto e^{-(Y(\mathbf{r})-Y(\mathbf{s}))^2/2\sigma_{\mathbf{r}}^2} \qquad \text{(2) Levin}$$

  - and normalized values

$$w_{\mathbf{rs}} \propto 1 + \frac{1}{\sigma_{\mathbf{r}}^2}(Y(\mathbf{r}) - \mu_{\mathbf{r}})(Y(\mathbf{s}) - \mu_{\mathbf{r}}) \qquad \text{(3) Levin}$$

Consider this case: their difference is the same
$|Y(\boldsymbol{r}_1) - Y(\boldsymbol{s}_1)| = |Y(\boldsymbol{r}_2) - Y(\boldsymbol{s}_2)|$,
but $Y(\boldsymbol{r}_1)\ Y(\boldsymbol{s}_1)$ are on the opposite side of the distribution and $Y(\boldsymbol{r}_2)\ Y(\boldsymbol{s}_2)$ are on the same side. U$(\boldsymbol{r}_2)\ U(\boldsymbol{s}_2)$ should be more similar. The further they are from the $\mu_{\boldsymbol{r}}$, the more similar they should be.



Optimization

- View the variables of the U image as a very long vector $\mathbf{x}$, and the U values of those scribbles as a given vector $\mathbf{b}$.

$$J(U) = \sum_{\mathbf{r}} \left(U(\mathbf{r}) - \sum_{\mathbf{s} \in N(\mathbf{r})} w_{\mathbf{rs}} U(\mathbf{s}) \right)^2 \qquad (1) \text{ Levin}$$

becomes $\underset{\mathbf{x}}{\operatorname{argmin}} (\mathbf{A}\mathbf{x} - \mathbf{y})^2$ (4.28)

where $\mathbf{A}$ is a sparse matrix calculated from Y using $(2)(3)$ Levin.

Additional Results

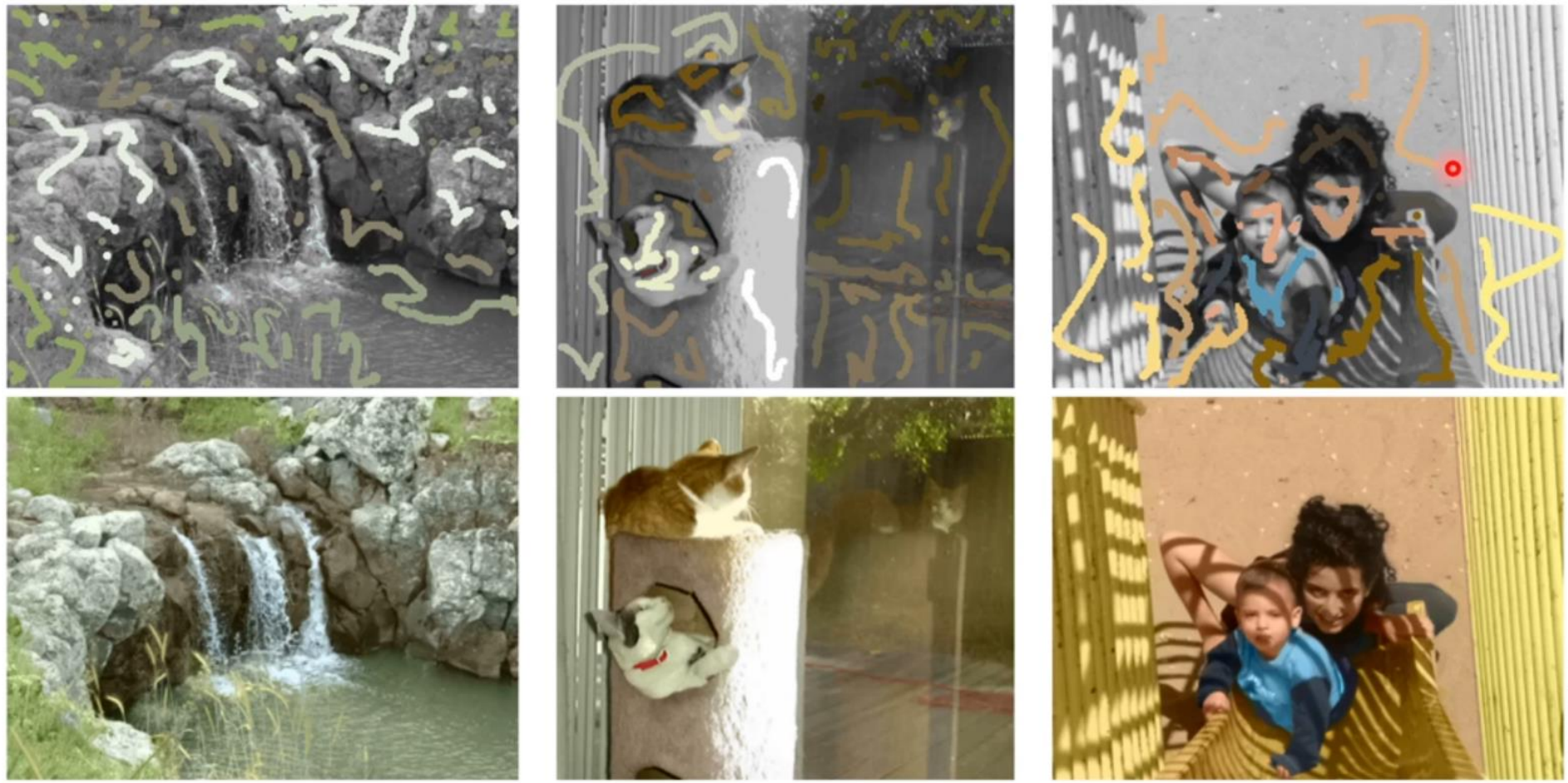


Figure 2
Levin

Interactive colorization

(a1) initial scribbles
(a2) artifacts
(b1) additional scribbles
(b2) result
(c1) additional scribbles
(c2) artifacts

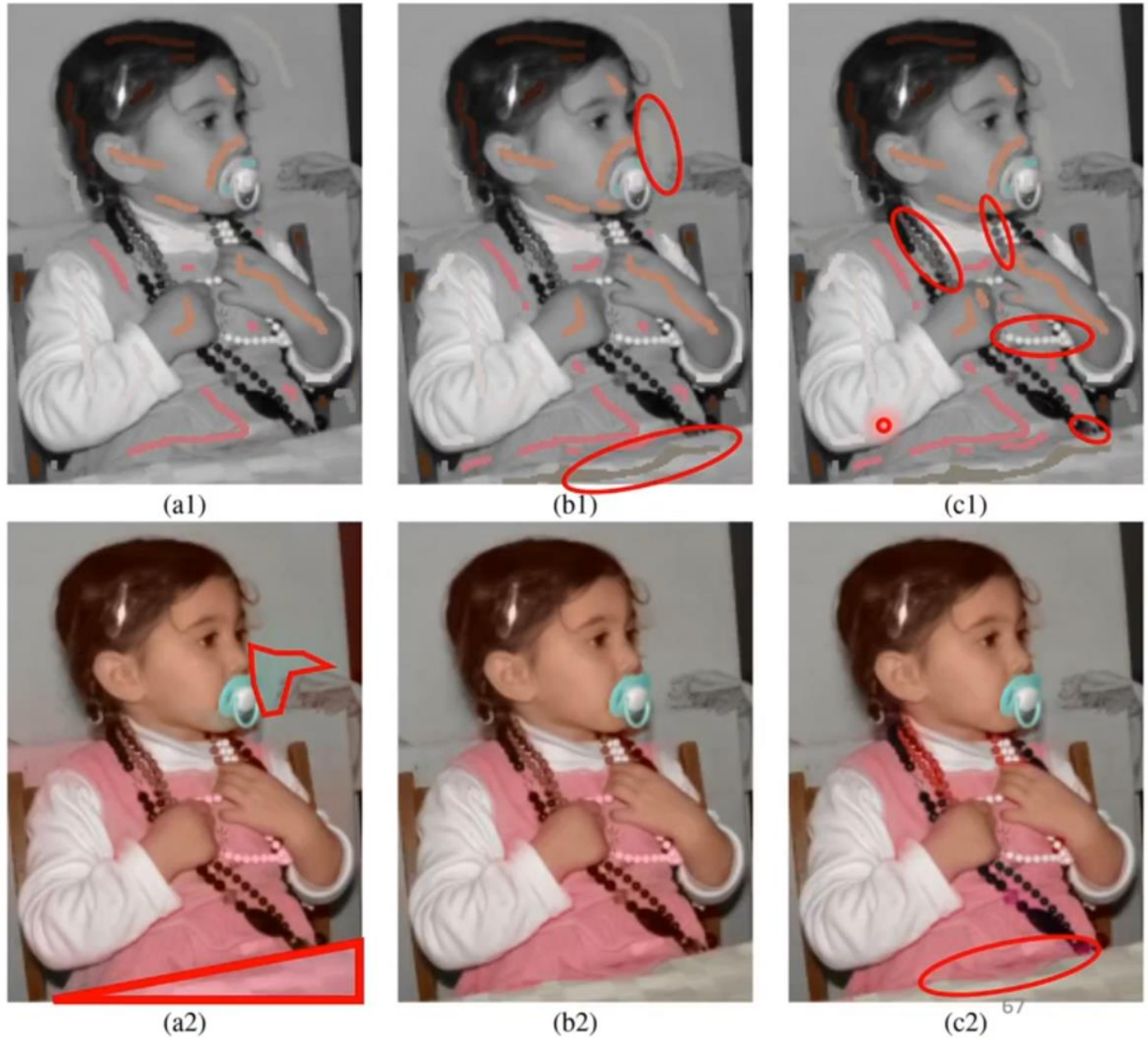


Figure 3
Levin

Recolor

Figure 4
Levin

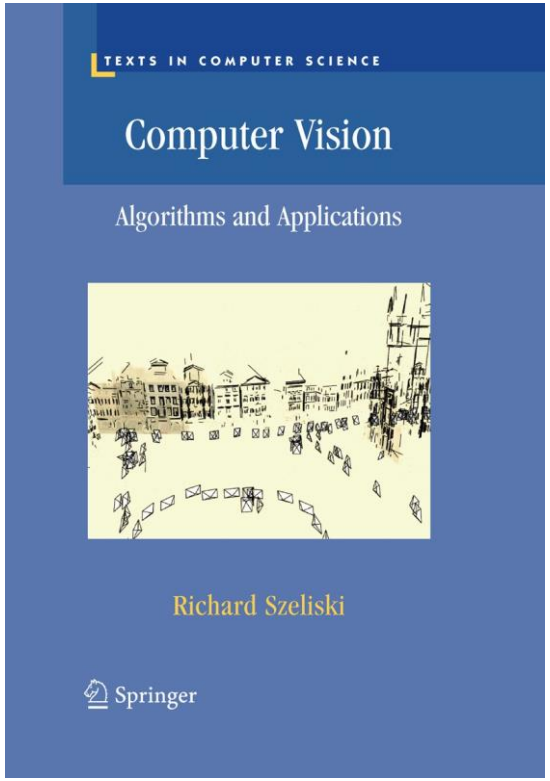
(a)
Input image

(b)
Recolored region and scribbles

(c)
Result

Questions?

- Reference



Pattern Recognition and Machine Learning by Christopher M. Bishop (2006)

- Full free PDF book downloadable from the author's official website (Link)