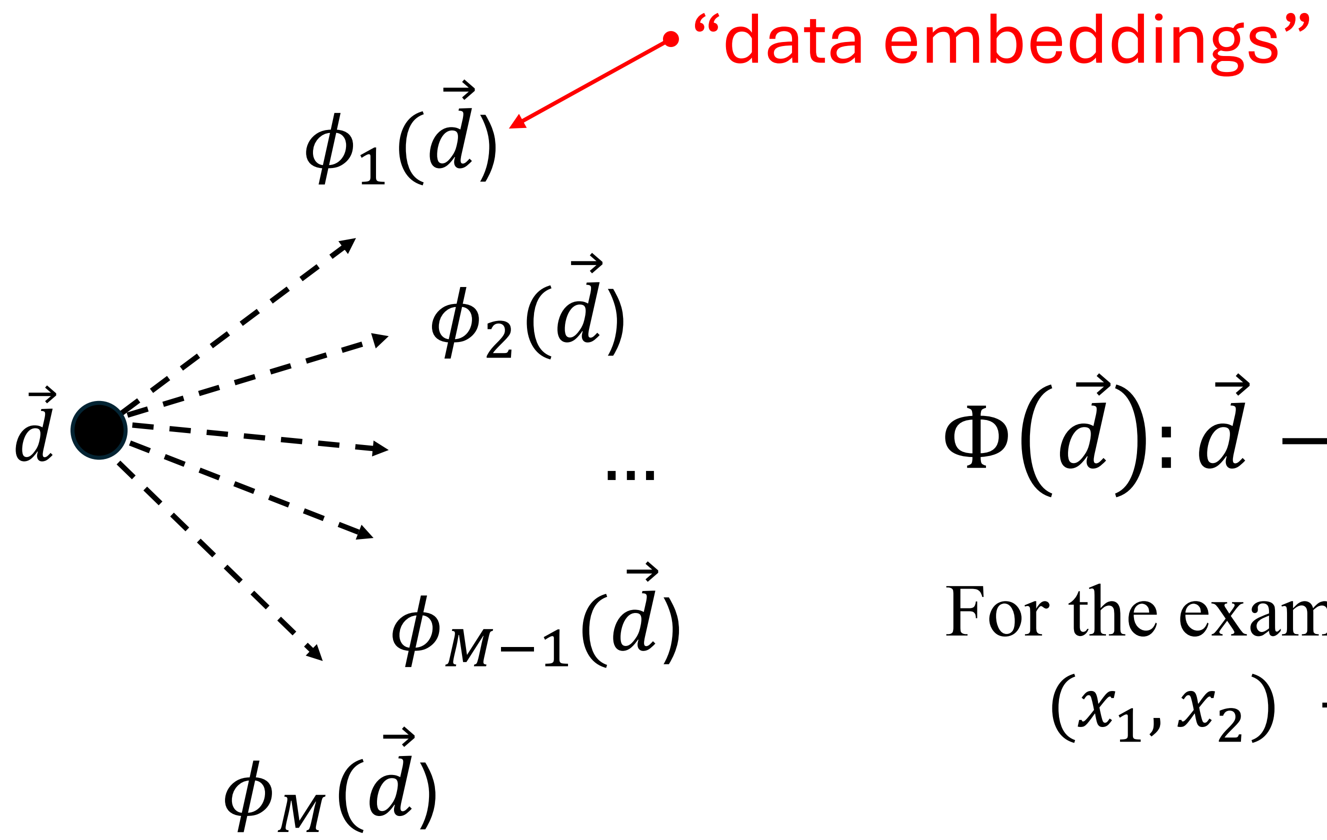
CS 461: Machine Learning Principles

Class 5: Sept. 19

Linear Regression and Data Pre-Processing

Instructor: Diana Kim

In the last class,
we talked about creating a feature space to accommodate the linear modeling for regression/ classification problems.

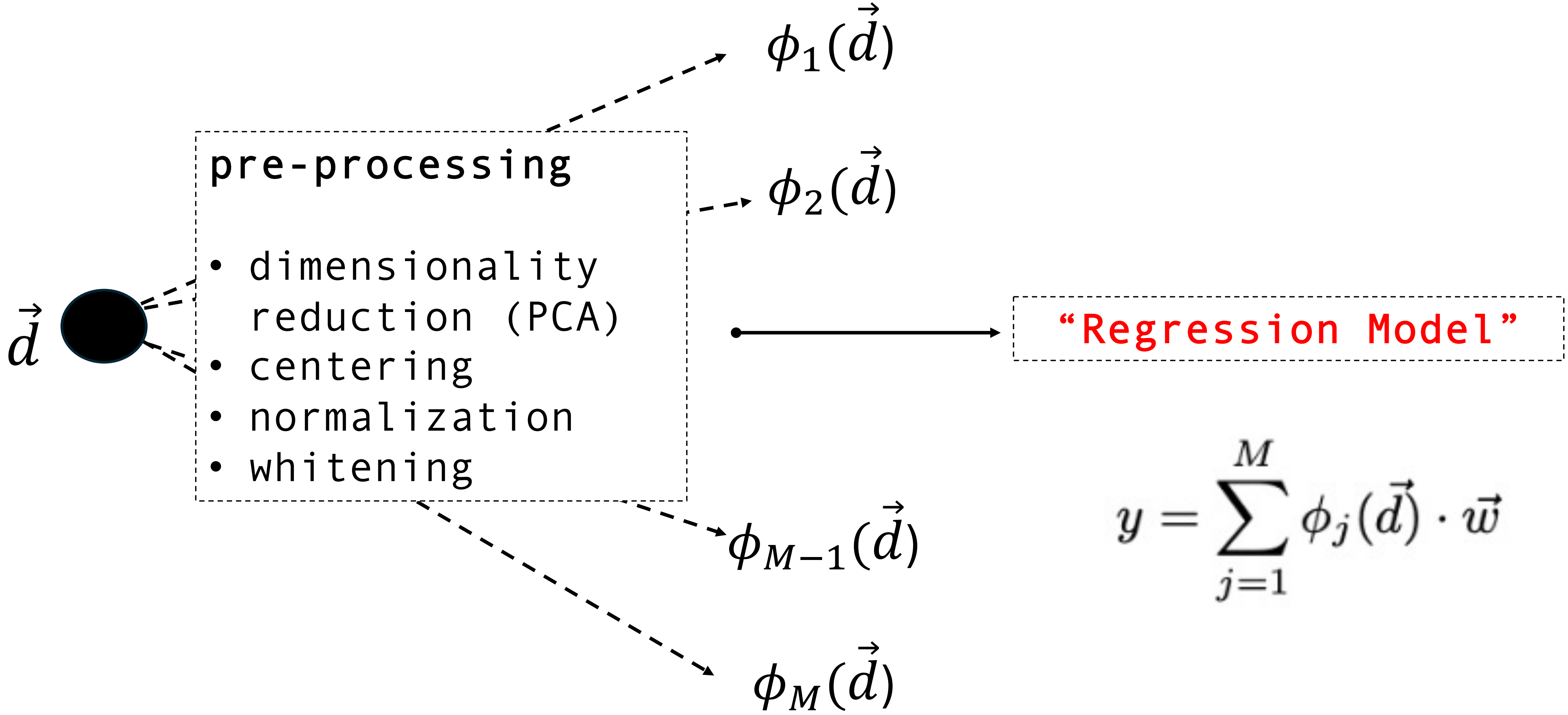


$$\Phi(\vec{d}): \vec{d} \rightarrow R^M$$

For the example of **XOR** problem,

$$(x_1, x_2) \rightarrow (x_1, x_2, x_1 \cdot x_2)$$

We may need the pre-processing steps before the algorithm
but let’s focus on “**regression algorithm**” first and cover pre-processing part.

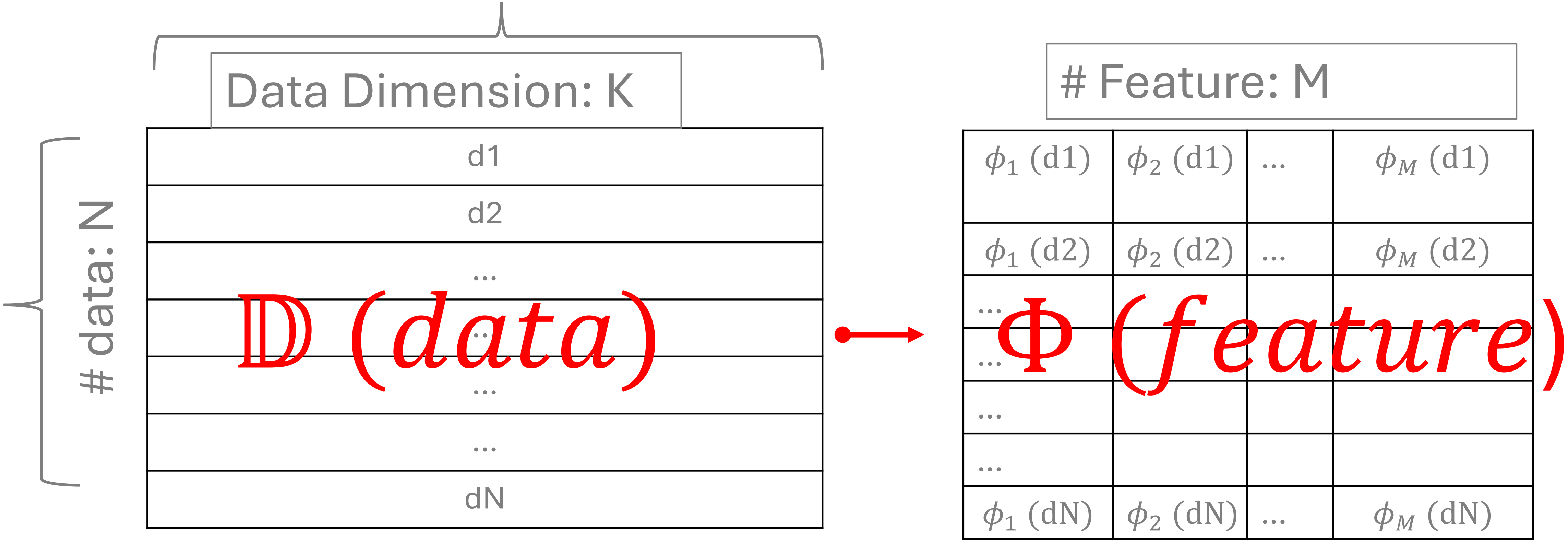


$$y = \sum_{j=1}^{M} \phi_j(\vec{d}) \cdot \vec{w}$$

Regression Problem

Regression Problem

- Suppose we defined a proper feature map $\phi(\vec{d})$ for data point $(\vec{d}, y)$. The data matrix D is transformed into Φ.



Regression Problem

- Suppose we defined a proper feature map $\phi(\vec{d})$ for data point $(\vec{d}, y)$.
  The data matrix D is transformed into Φ.
  We set the linear combination of $\phi(\vec{d})$ with $\vec{w}$ to predict the value y.

$$y = \Phi(\vec{d}) \cdot \vec{w} \quad + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

ε errors from :

+ imperfection feature space design

+ imperfection hypothesis space

+ error from measurement

Regression Problem

- Suppose we defined a proper feature map $\phi(\vec{d})$ for data point $(\vec{d}, y)$. then we can transform a data matrix D into Φ.
- We set the linear combination of $\phi(\vec{d})$ with $\vec{w}$ to predict the value y.

$$y = \Phi(\vec{d}) \cdot \vec{w} \quad + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- We have observed data. (design matrix Φ)

$$\begin{bmatrix} \phi_1\ (d1) & \phi_2\ (d1) & \ldots & \phi_M\ (d1) \\ \phi_1\ (d2) & \phi_2\ (d2) & \ldots & \phi_M\ (d2) \\ \ldots & & & \\ \ldots & & & \\ \ldots & & & \\ \ldots & & & \\ \phi_1\ (dN) & \phi_2\ (dN) & \ldots & \phi_M\ (dN) \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ \ldots \\ \\ W_M \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ \\ \\ \\ y_n \end{bmatrix}$$

Regression Problem

- Suppose we defined a proper feature map $\phi(\vec{d})$ for data point $(\vec{d}, y)$. then we can transform a data matrix D into Φ.
- We set the linear combination of $\phi(\vec{d})$ with $\vec{w}$ to predict the value y.

$$y = \Phi(\vec{d}) \cdot \vec{w} \quad + \varepsilon, \ \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- We have observed data.
- We want to estimate $\vec{w}$

$$\begin{bmatrix} \phi_1\,(\mathrm{d1}) & \phi_2\,(\mathrm{d1}) & \dots & \phi_M\,(\mathrm{d1}) \\ \phi_1\,(\mathrm{d2}) & \phi_2\,(\mathrm{d2}) & \dots & \phi_M\,(\mathrm{d2}) \\ \dots & & & \\ \dots & & & \\ \dots & & & \\ \dots & & & \\ \phi_1\,(\mathrm{dN}) & \phi_2\,(\mathrm{dN}) & \dots & \phi_M\,(\mathrm{dN}) \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ \dots \\ \\ W_M \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ \\ \\ \\ y_n \end{bmatrix}$$

Regression Algorithm
: how to find the $\vec{w}$?

Regression Problem: Estimation Problem

from observations

$$y = \Phi(\vec{d}) \cdot \vec{w} + \varepsilon, \ \ \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

want to estimate W!

We learned the two possible ways (MAP and MLE)!

- $w^* = argmax\, p(\vec{w}|y, \Phi) = \dfrac{p(y|\vec{w}, \Phi)p(\vec{w})}{p(\Phi)}$: (MAP)
- $\boldsymbol{w^* = argmax\, p(y|\vec{w}, \Phi)}$: **(MLE)**
- Q: the distribution of y~?

Regression Problem: Estimation Problem

Suppose we have a feature map $\phi(\vec{d})$ for a data point $\vec{d}$

we want to learn $\vec{w}$ whose the linear combination of $\phi(\vec{d})$ predicts the value y with error $\varepsilon \sim N(0, \sigma^2)$.

we have observations: data Φ ($\mathrm{N} \times M$) and $\vec{y}$

- $w^* = argmax\, p(\vec{w} \mid \vec{y}, \Phi) = \dfrac{p(y|\vec{w}, \Phi)p(\vec{w})}{p(\Phi)}$: (MAP)
- $w^* = argmax\, p(\vec{y}|\vec{w}, \Phi)$: (MLE)

  $= argmax\, \mathcal{N}_y(\Phi\left(\vec{d}\right) \cdot \vec{w}, \sigma^2 \mathrm{I})$ (when observations are i.i.d)

Regression Problem: Estimation Problem

we have observations: data Φ ($N \times M$) and $\vec{y}$

- $w^* = argmax\, p(\vec{y}|\vec{w}, \Phi)$: (MLE)

$$= argmax\, \mathcal{N}_y\left(\Phi(\vec{d}) \cdot \vec{w} \quad , \sigma^2 \mathrm{I}\right)$$

Ground Truth (data)

Prediction!

$$\arg\min_w ||\vec{y} - \Phi \cdot \vec{w}||^2$$

$$\arg\min_w ||\vec{y} - \Phi \cdot \vec{w}||^2$$

MLE becomes
Minimum Mean Square Error Problem

$$J(\vec{w}) = ||\vec{y} - \Phi \cdot \vec{w}||^2$$

$$J(\vec{w}) = (\vec{y}^t - \vec{w}^t \cdot \Phi^t) \cdot (\vec{y} - \Phi \cdot \vec{w})$$

$$\nabla J(\vec{w}) = -2 \cdot \Phi^t \cdot (\vec{y} - \Phi \cdot \vec{w}) = 0$$

$$\Phi^t \cdot \Phi \cdot \vec{w} = \Phi^t \cdot \vec{y}$$

Normal Equation

Regression Problem: Estimation Problem

$$\boxed{\Phi^t \cdot \Phi} \cdot \vec{w} = \Phi^t \cdot \vec{y}$$

The three possible cases:

- invertible (Rank M)
- invertible (Rank M) but close to singular (very small eigenvalues)
- non – invertible (Rank $<M$)

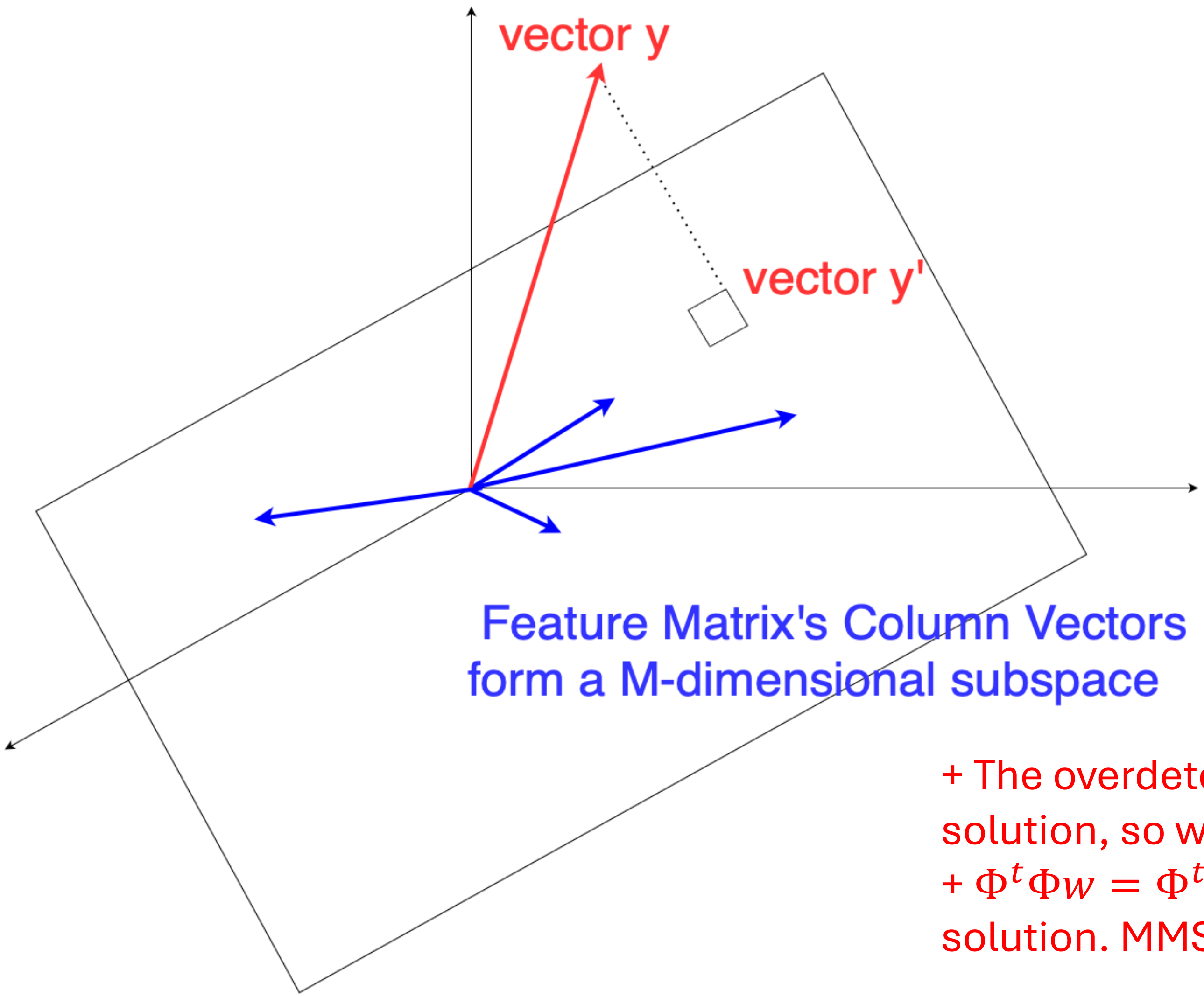
Note) close look into the bias term

$$J(w) = ||\vec{y} - \Phi \cdot \vec{w}||^2$$
$$= ||\vec{y} - \Phi' \cdot \vec{w'} - [b, b, ...b]^t||^2$$
$$\frac{\partial J}{\partial b} = -2 \cdot [1, 1, 1., , , 1] \cdot (\vec{y} - \Phi' \cdot \vec{w'} - [b, b, ...b]^t) = 0$$
$$= \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} y_i' - b \cdot N = 0$$
$$bN = \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} y_i'$$
$$b = \frac{1}{N} \sum_{i=1}^{N} y_i - \frac{1}{N} \sum_{i=1}^{N} y_i'$$

$$\begin{bmatrix} \phi_1\,(\mathrm{d1}) & \phi_2\,(\mathrm{d1}) & \ldots & \mathbf{1} \\ \phi_1\,(\mathrm{d2}) & \phi_2\,(\mathrm{d2}) & \ldots & \mathbf{1} \\ \ldots & & & \\ \ldots & & & \\ \ldots & & & \\ \ldots & & & \\ \phi_1\,(\mathrm{dN}) & \phi_2\,(\mathrm{dN}) & \ldots & \mathbf{1} \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ \ldots \\ \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ \\ \\ \\ y_n \end{bmatrix}$$

- $y' = prediction\ without\ bias?$

Geometrical Interpretation: MMSE Solution



+ The overdetermined system $\Phi w = y$ might not have a solution, so we find an approximated solution instead.
+ $\Phi^t \Phi w = \Phi^t y$ The approximated solution is the MMSE solution. MMSE can be derived from MLE.

Geometrical Interpretation MMSE Estimation

- solving overdetermined System $$\Phi(\vec{d}) \cdot \vec{w} = \vec{y}$$

- projection to the data space and find an approximated solution

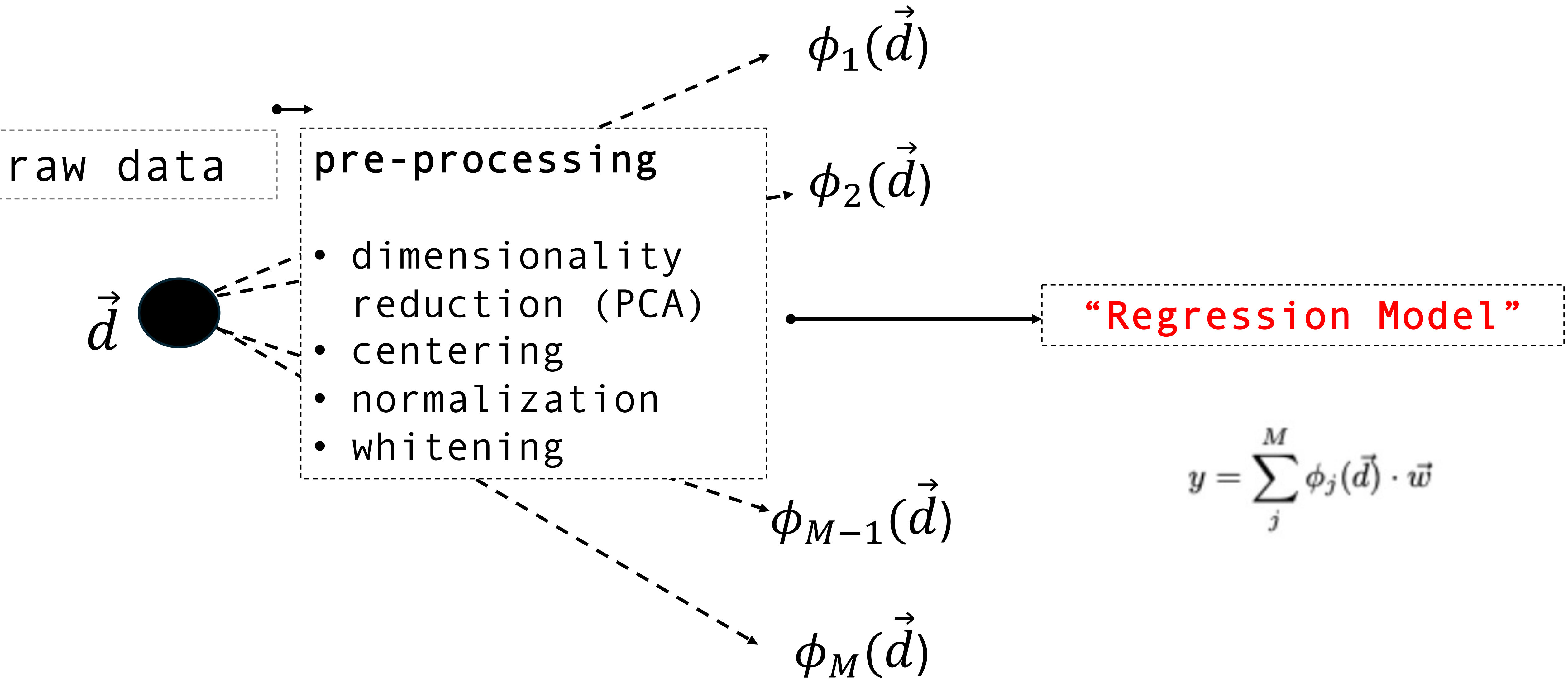
$$\Phi(\vec{d})^t \cdot \Phi(\vec{d}) \cdot \vec{w} = \Phi(\vec{d})^t \cdot \vec{y}$$

- when data space's rank is M : unique approximated solution

- when data space's rank is less than M: pseudo-inverse solution

We learned regression algorithm.
but we need some pre-processing steps for successful learning.

We need the pre-processing steps before the algorithm.



+ for regression,
we will focus on dimensionality reduction and whitening

Two Kinds of Raw Data in Regression Problems

- each feature dimension has semantic.

  "prediction of house market price"

  <u>single house/ townhome, square feet, garden size, public school scores</u>

- no semantic, a whole vector represents an image/ audio /texts

Both cases will need pre-processing steps, but

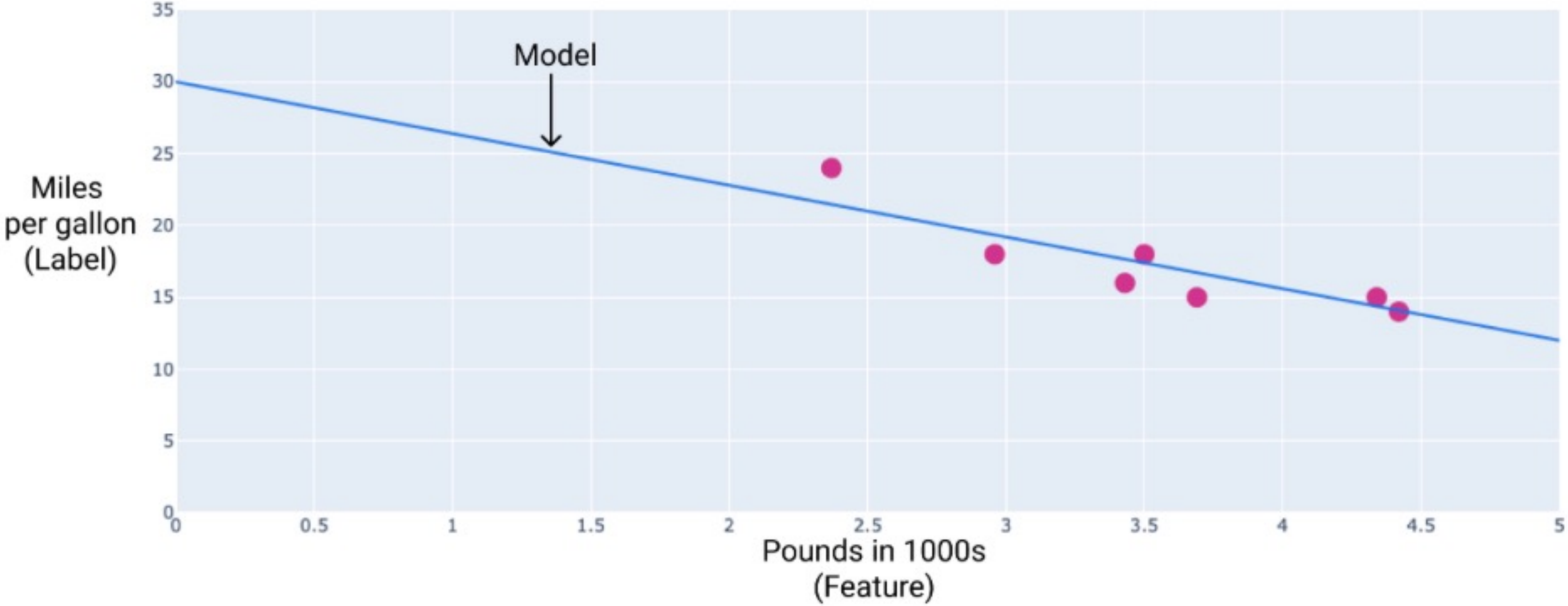
Semantic Data

[A car's fuel efficiency in miles per gallon]

Pounds in 1000s (features)	Miles Per Gallon (Label)
3.5	18
3.69	15
3.44	18
3.43	16
4.34	15
4.42	14
2.37	24

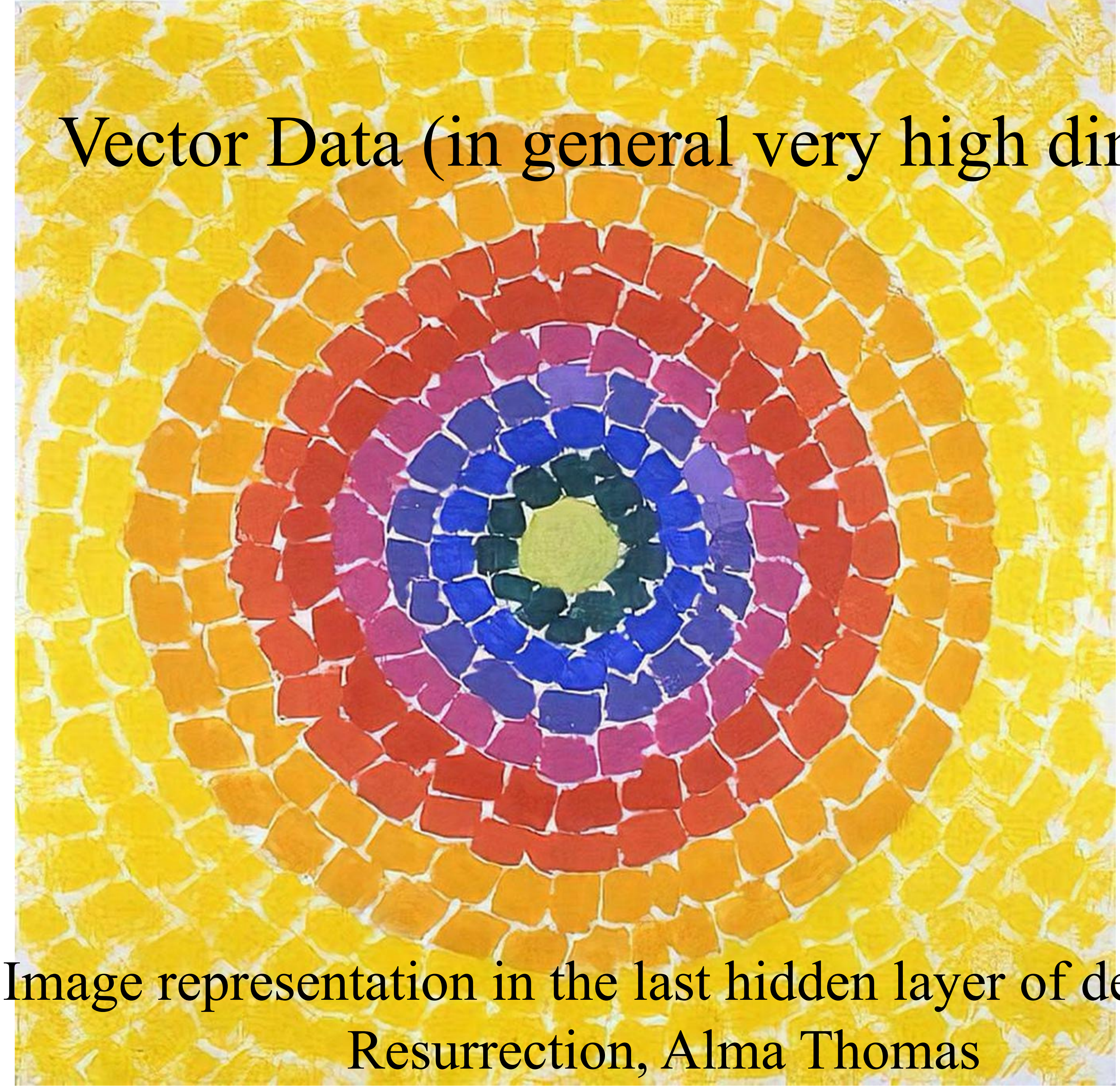
From https://developers.google.com/machine-learning/crash-course/linear-regression

Regression Model: Prediction of Miles/Gallon



From https://developers.google.com/machine-learning/crash-course/linear-regression

Vector Data (in general very high dimensional)

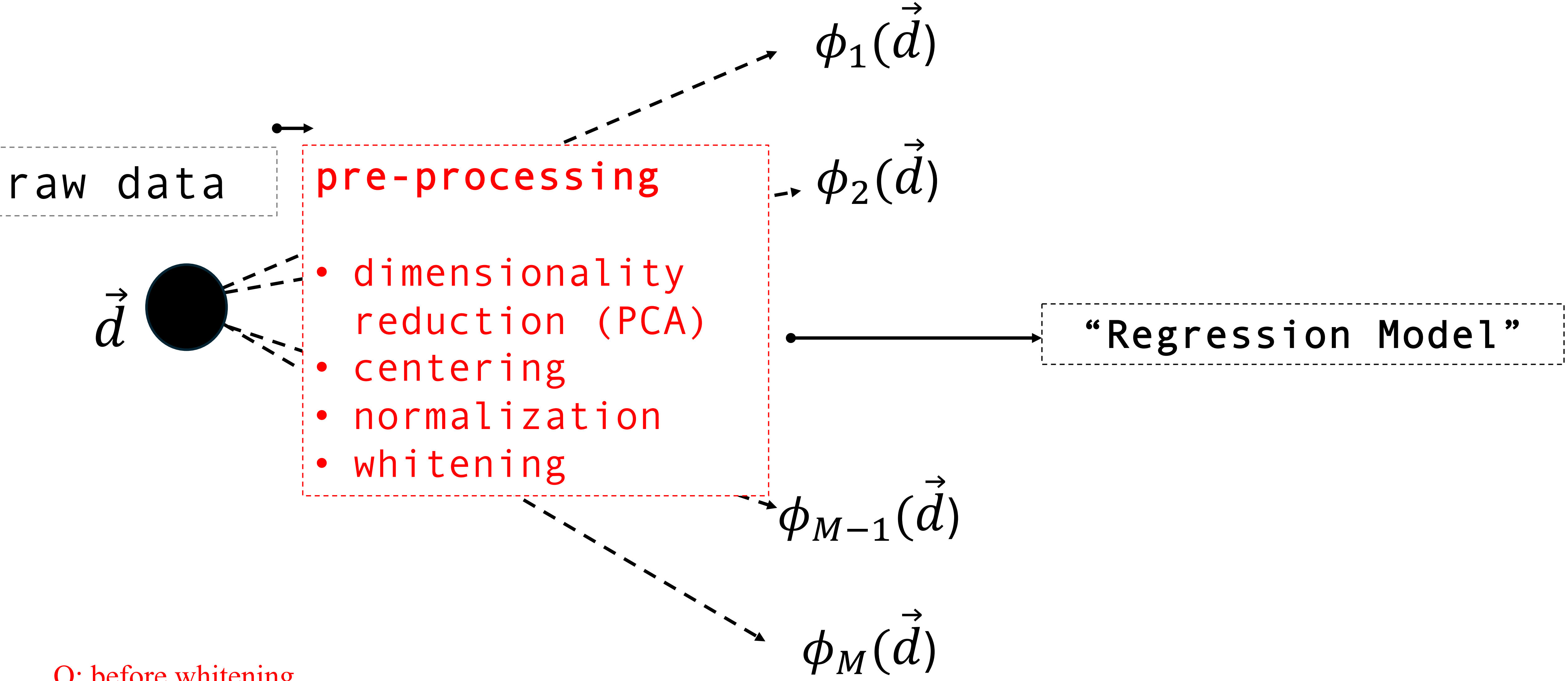


1.66116878e-02	1.43487025e-02	2.32550185e-02	1.19144898e-02
9.63837430e-02	8.31691474e-02	3.06324158e-02	2.04893108e-02
3.52624245e-02	5.56392968e-03	1.73700173e-04	1.50486574e-01
1.23705138e-02	5.22252880e-02	5.49408374e-03	4.67238687e-02
1.44064635e-01	5.96226342e-02	3.30738463e-02	7.86291659e-02
1.78842451e-02	8.52992311e-02	1.89456772e-02	1.03305764e-02
3.74222398e-02	8.35433230e-02	1.95080508e-02	3.26446295e-02
2.96458267e-02	1.08750183e-02	1.02128655e-01	4.72795311e-03
[illegible]	2.39002742e-02	1.02527821e-02	2.37334445e-02
[illegible]	2.45263092e-02	4.37902249e-02	2.01957620e-04
1.88012738e-02	3.28723639e-02	2.84272023e-02	9.55437776e-03
4.14613076e-02	1.39710018e-02	1.36978943e-02	1.87548213e-02
8.24719146e-02	4.43845317e-02	3.53335054e-03	7.63716456e-03
9.22968891e-03	3.64266671e-02	2.88719591e-02	1.30119538e-02
2.02936288e-02	1.33205568e-02	1.15464339e-02	8.20994973e-02
1.77106280e-02	8.52256175e-03	1.17111830e-02	1.45943370e-02
9.04859081e-02	4.89737056e-02	4.16163765e-02	1.04203597e-02
2.58007385e-02	4.69408296e-02	4.55647372e-02	1.24655450e-02
2.41902079e-02	2.42145211e-02	4.81210562e-04	8.90940428e-03
6.85443357e-02	4.78595048e-02	1.18027581e-02	1.17037995e-02
1.90981105e-02	1.00829145e-02	5.23220561e-03	3.84746492e-02
8.81355628e-02	3.36198583e-02	5.35092168e-02	4.87123579e-02
8.99140537e-03	5.39787523e-02	4.79393527e-02	2.99579669e-02
1.71675645e-02	3.76332477e-02	7.88647458e-02	3.79528590e-02
4.51750029e-03	9.38767791e-02	3.61216962e-02	1.98117495e-02
3.94294709e-02	1.14793226e-01	1.48017062e-02	6.40132884e-03
8.16167146e-03	6.94637448e-02	3.43800858e-02	8.04584008e-03
1.55022442e-01	2.59600277e-03	3.20520252e-02	1.00370266e-01
6.82575488e-03	3.21909995e-03	6.07831627e-02	5.22131985e-03
4.10482734e-02	5.29111736e-02	4.37461957e-02	4.37461808e-02
4.10865359e-02	9.59158130e-03	4.68185917e-02	7.04082549e-02
5.19240461e-03	9.47480425e-02	1.72703192e-02	1.32609099e-01
1.84857957e-02	2.34019849e-03	2.21508313e-02	3.19128227e-03
1.03731174e-02	7.90489465e-02	3.40001471e-02	2.08658073e-02
3.63909267e-03	2.93061193e-02	1.79619715e-02	3.92507110e-03
1.22312911e-01	4.27385271e-02	4.02529091e-02	6.87315594e-03
1.79619640e-02	1.44496362e-03	3.47868539e-04	2.03075245e-01
2.45202169e-01	1.26138151e-01	1.07999377e-01	1.46901429e-01
9.70007405e-02	1.03836969e-01	1.09804377e-01	1.04106106e-01
8.70869756e-02	8.81577432e-02	7.79228508e-02	9.59928930e-02
2.06121951e-01	2.38734394e-01	1.37491360e-01	7.11895898e-02
9.10348147e-02	1.08147562e-01	8.93435627e-02	8.45326930e-02
8.54639262e-02	7.94288218e-02	7.84831643e-02	6.98279142e-02
6.67123348e-02	7.02826679e-02	1.02719694e-01	1.04542613e-01
1.12103589e-01	8.02482218e-02	1.26211137e-01	1.22251317e-01
1.18328705e-01	9.65996012e-02	9.47735459e-02	8.21543038e-02
7.41177499e-02	1.03439212e-01	1.14290312e-01	1.15447372e-01
1.28355548e-01	1.06327742e-01	7.30694234e-02	6.20305464e-02
1.06132567e-01	7.94187784e-02	8.86070132e-02	8.47868249e-02
1.07920051e-01	8.36525112e-02	6.55624866e-02	7.80229717e-02
8.64467472e-02	8.55527893e-02	1.10759147e-01	1.32106051e-01
7.44441077e-02	5.27140088e-02	1.08958408e-01	8.06024447e-02
9.61078107e-02	9.56790447e-02	1.04670644e-01	8.01085979e-02
6.94930553e-02	7.93032944e-02	9.49410051e-02	7.71025643e-02
1.05781302e-01	1.46627113e-01	6.05126023e-02	4.13953587e-02
1.23981662e-01	1.08231083e-01	1.34232193e-01	1.18365087e-01
1.09341882e-01	8.85261148e-02	8.09772611e-02	8.30637813e-02
1.13967851e-01	8.66363198e-02	1.12511240e-01	1.40123367e-01
6.65690452e-02	4.43194956e-02	1.57082587e-01	1.04566351e-01
9.03969407e-02	1.14839226e-01	1.21048279e-01	9.68690664e-02
9.22555625e-02	1.10619672e-01	1.21558294e-01	8.79304186e-02
1.04587585e-01	1.42198473e-01	8.80973265e-02	4.42507043e-02

Image representation in the last hidden layer of deep-CNN

Resurrection, Alma Thomas

We need the pre-processing steps for the raw data.

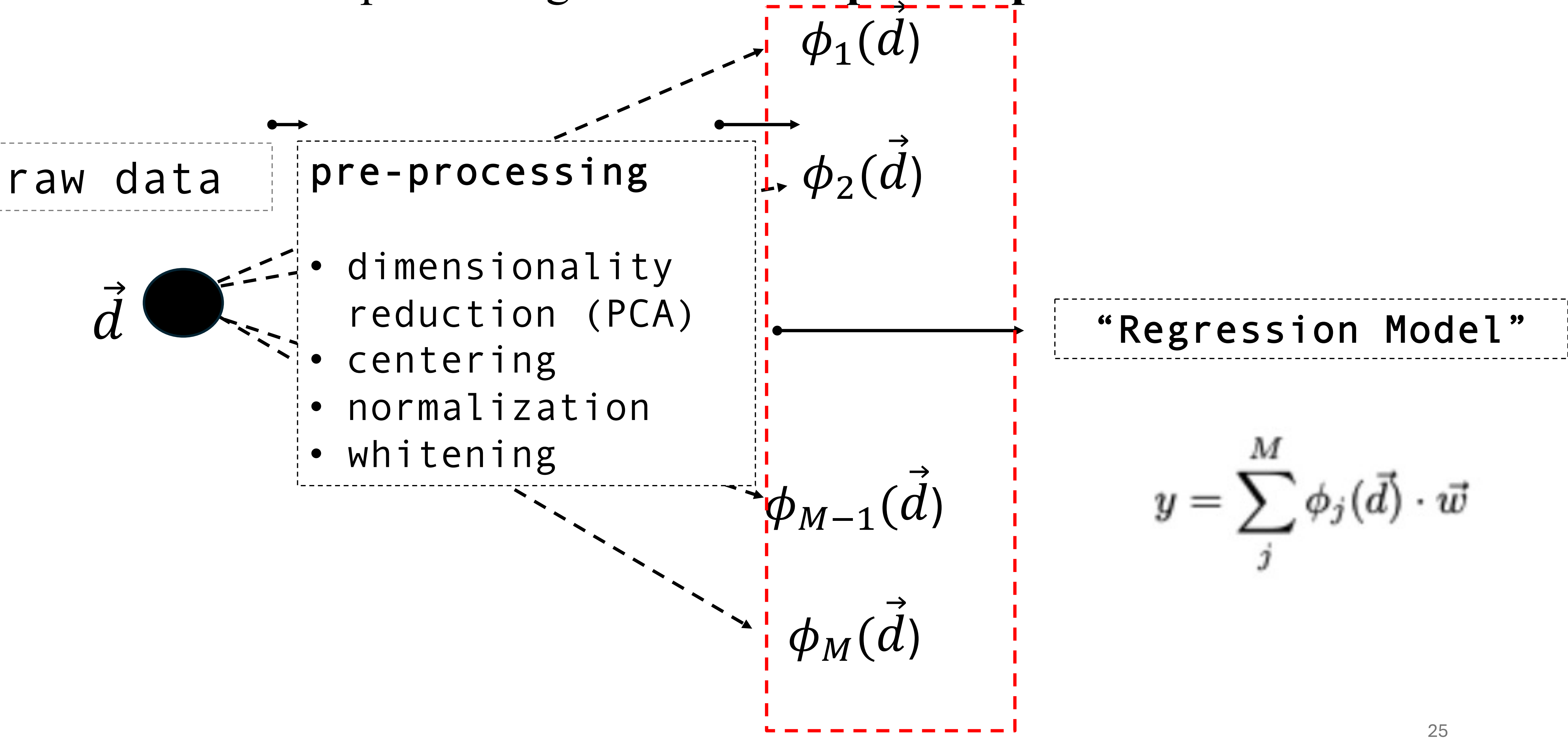


Q: before whitening,
We must do PCA. Why? Otherwise, some error dimension can be magnified by whitening (1/small eigenvalue)

Centering, Normalization, Standardization, Whitening

- Centering: $\vec{x} - E[\vec{X}]$
- Normalization: $\dfrac{\vec{x}}{||x||}$
- Standardization: $\dfrac{x_i - E[X_i]}{VAR[X_i]}$ (element-wise operation)
- Whitening : $Y = AX + \vec{b}$ when $Y \sim N(0, I)$

Raw data ⟶ Pre-processing ⟶ **Feature Space Expansion with Basis**



$$y = \sum_{j}^{M} \phi_j(\vec{d}) \cdot \vec{w}$$

Basis Function (1) : Polynomial Expansion

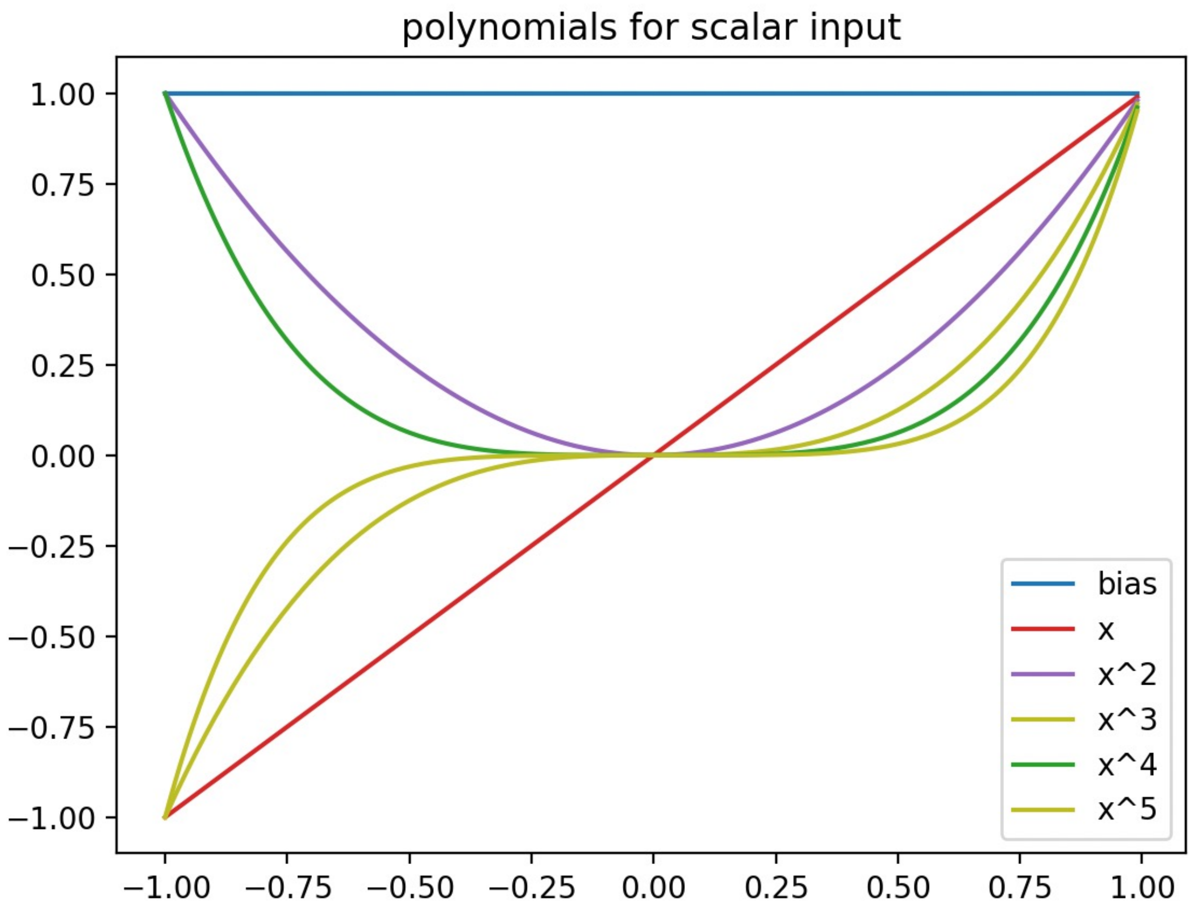
+ Pre-processed data:
 (reduced dimensions and whitened)

$\vec{d} = (x_1, x_2)$

$$(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

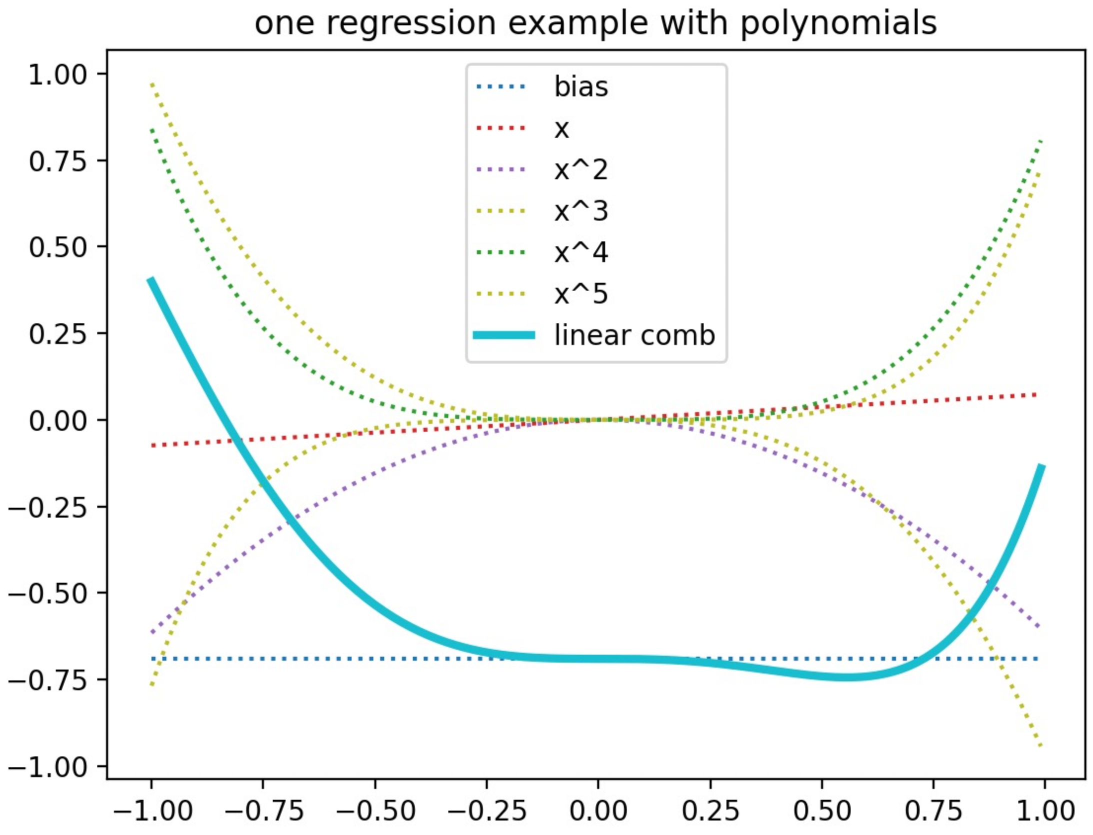
	1	x_2	${x_2}^2$
1	1	x_2	${x_2}^2$
x_1	x_1	x_1x_2	×
${x_1}^2$	${x_1}^2$	×	×

Basis Function (1) : Polynomial Expansion



- This example shows the case when input is scalar.
- Q: what if input is a 2D vector? How would you draw the plot?

Basis Function (1) : Polynomial Expansion



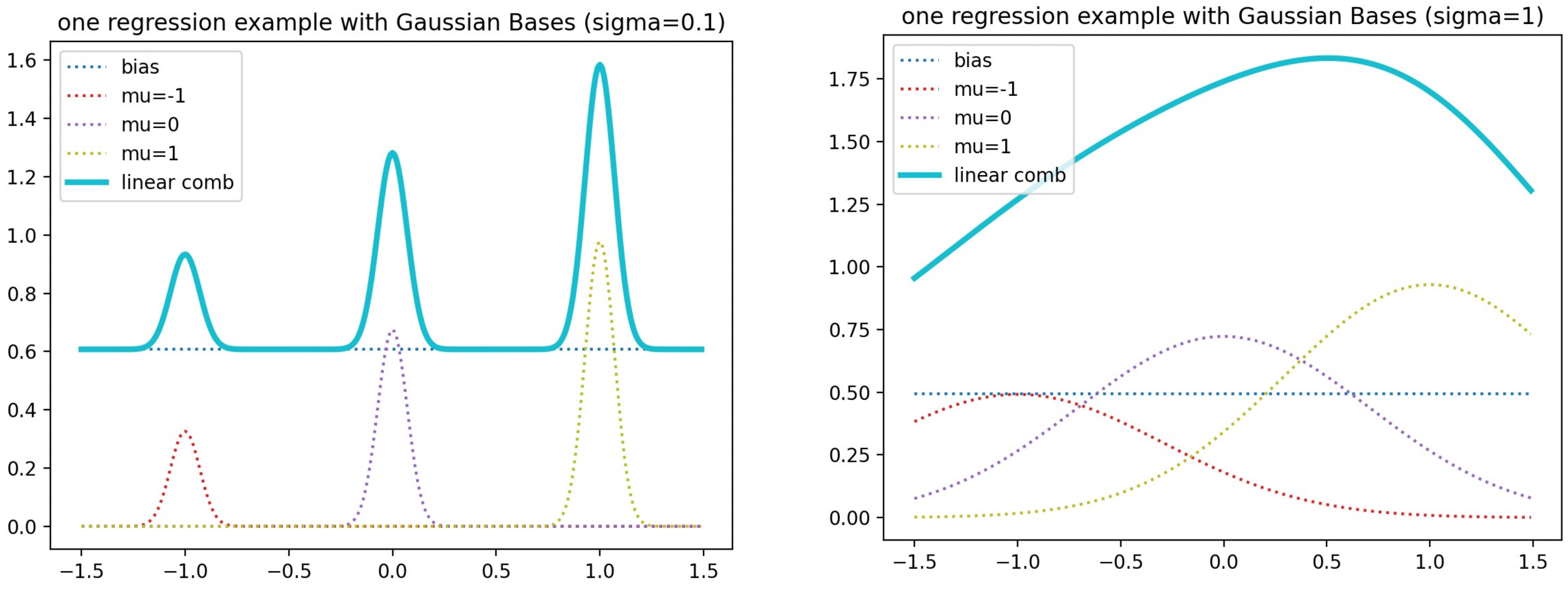
- This example shows the case when we set $\vec{w}$ *with an arbitrary parameter*. This shows a possible regression function.

Basis Function (2) : Gaussian Basis function

$$\phi_j = \exp\{-\frac{(x-\mu_j)^2}{2\sigma^2}\}$$

Q: the magnitude of σ^2?
(small: local and spiky vs. large: global and smooth)
Q: the locations of μ_j? (dense / sparse)

Basis Function (2) : Gaussian Basis function



The magnitude of sigma determines the influence over other neighboring Gaussian functions.

Summary: Train and Test Procedures for Regression Problem

- A set of hypothetical basis functions (polynomial / Gaussian)
- We have data points ((d_1, y_1), (d_2, y_2), (d_3, y_3), …, ((d_N, y_N))

Train

- Train data PCA for dim-reduction (high-dimensional data) and whitening. (save the PCA blocks computed with training set)
- Normal equation to estimate W

Test

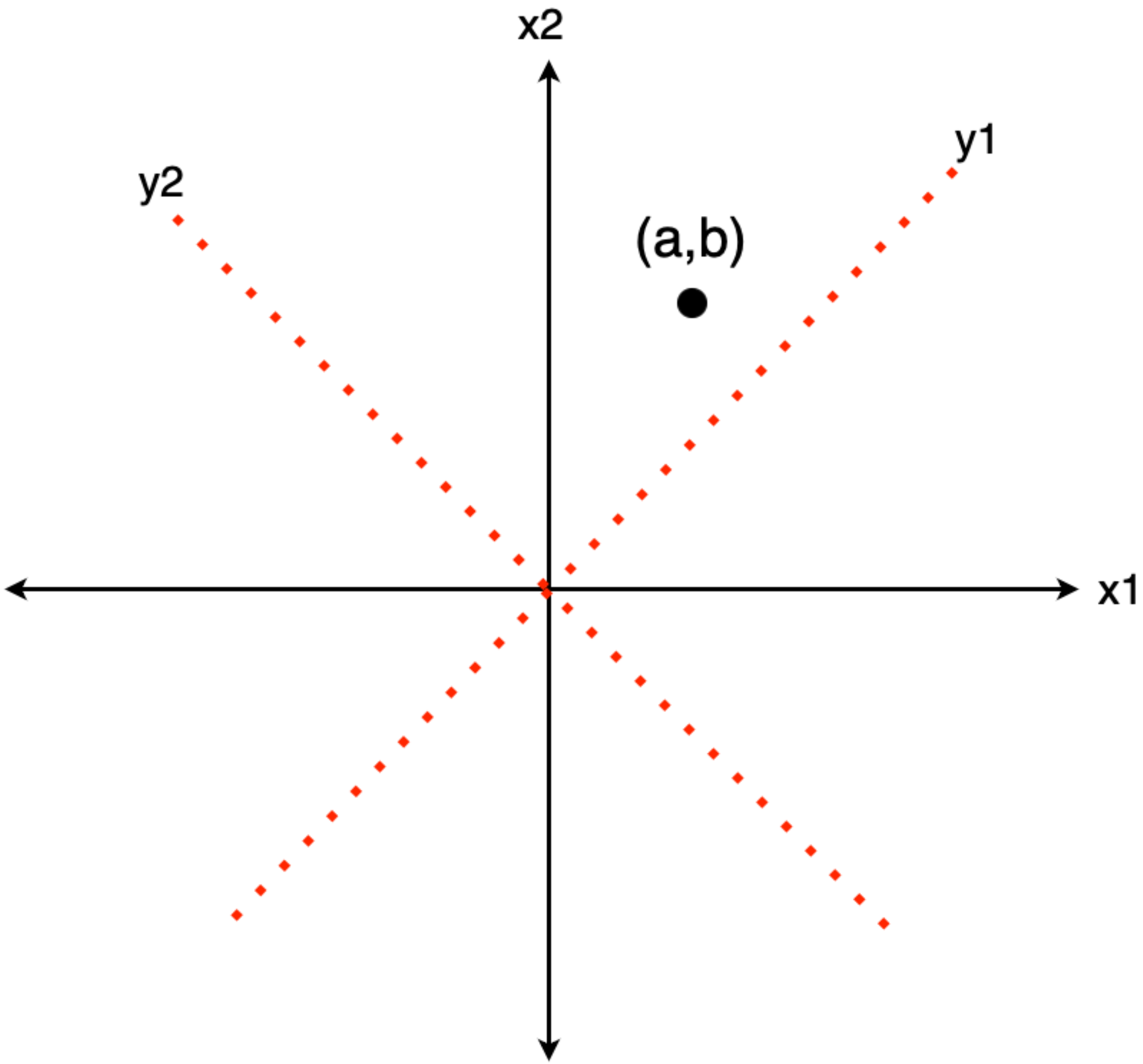
- Test data PCA for dim-reduction and whitening with the same block used in training!!!! (important)
- Compute error between the groudturth y and prediction y': $||y - y'||^2$

Dimensionality Reduction: PCA
- how can we reduce high-dimensional images into 2D space?

Principal Component Analysis (PCA)

We want to make data (N-D) moves around within a subspace (M-D), (N>M)

Vector $\vec{x} = (a, b)$ can be represented by different orthonormal bases.



- X_n

When $x_n \in R^D$ and u_i where $i = 1, 2, ..., D$ are are abnormal basis,

$$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i \quad \text{and} \quad \alpha_{ni} =< x_n, u_i >$$

- $\widetilde{X_n}$

We want to approximate x_n on the subspace of the first M basis,

$$\tilde{x_n} = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

Q: What is the optimal values for Z_{ni} and b_i minimizing J = $\| X_n - \widetilde{X_n} \|^2$

We want to minimize the averaged square error between x_n and $\tilde{x_n}$

$$\arg\min_{(z_{ni},b_i)} J = \frac{1}{N}\sum_{n=1}^{N}(< x_n, u_i > \cdot u_i^t - \sum_{i=1}^{M} z_{ni}u_i^t - \sum_{i=M+1}^{D} b_i u_i^t)\cdot$$

$$(< x_n, u_i > \cdot u_i - \sum_{i=1}^{M} z_{ni}u_i - \sum_{i=M+1}^{D} b_i u_i)$$

- Respect to z_{nk}

$$\frac{\partial J}{\partial z_{nk}} = (-2u_k^t)\cdot(< x_n, u_i > \cdot u_i - \sum_{i=1}^{M} z_{ni}u_i - \sum_{i=M+1}^{D} b_i u_i)$$

$$= -2 < x_n, u_k > +2z_{nk} = 0$$

- Respect to b_r

$$\frac{\partial J}{\partial b_r} = \frac{1}{N}\sum_{n=1}^{N}(-2u_r^t)\cdot(< x_n, u_i > \cdot u_i - \sum_{i=1}^{M} z_{ni}u_i - \sum_{i=M+1}^{D} b_i u_i)$$

$$= \frac{1}{N}\sum_{n=1}^{N}(-2 < x_n, u_r > +2b_r)$$

- Rewrite J

$$J = ||x_n - \tilde{x_n}||^2 = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x_n - \bar{x})^t u_i)^2$$

$$= \sum_{i=M+1}^{D} u_i^t \cdot \boxed{\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^t} \cdot u_i$$

$$= \sum_{i=M+1}^{D} u_i^t \Sigma u_i$$

Estimation of COV (X,X)

- Lagrangian Function for the Constraint ${u_i}^t u_i = 1$

Lagrangian function for the constraint $||u_i|| = 1$

$$J(\lambda) = \sum_{i=M+1}^{D} u_i^t \Sigma u_i + \lambda(1 - u_i^t u_i)$$

$$\frac{\partial J}{\partial u_i} = \Sigma u_i - \lambda^* u_i = 0$$

Q: What the optimal solution indicate about u_i?

- Go back to J

$$J = ||x_n - \tilde{x_n}||^2 = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x_n - \bar{x})^t u_i)^2$$

$$= \sum_{i=M+1}^{D} u_i^t \cdot \boxed{\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^t} \cdot u_i$$

$$= \sum_{i=M+1}^{D} u_i^t \Sigma u_i$$

Estimation of COV (X,X)

Q: To minimize J?

Now, we are ready to define $\widetilde{X_n}$ $(PCA\ Approximation)$

$$
\begin{aligned}
\tilde{x_n} &= \sum_{i=1}^{M}(x_n^t u_i)u_i + \sum_{i=M+1}^{D}(\bar{x}^t u_i)u_i \\
&= \bar{x} - \bar{x} + \sum_{i=1}^{M}(x_n^t u_i)u_i + \sum_{i=M+1}^{D}(\bar{x}^t u_i)u_i \\
&= \bar{x} - \sum_{i=1}^{M}(\bar{x}^t u_i)u_i + \sum_{i=1}^{M}(x_n^t u_i)u_i \\
&= \bar{x} + \sum_{i=1}^{M}((x_n^t - \bar{x}^t)u_i)u_i \\
&= \bar{x} + U_M U_M^t (x_n - \bar{x})
\end{aligned}
$$

$\widetilde{X_n}$ is not full dimension.

Depending on how we select U_M, we can define different approximations.

- Variance of $\widetilde{x_n}$

$$\tilde{x_n} - \bar{x} = u_j^t(x_n - \bar{x})u_j$$

$$\frac{1}{N}(\tilde{x_n} - \bar{x})^t(\tilde{x_n} - \bar{x})^t = \frac{1}{N}u_j^t(x_n - \bar{x})u_j u_j^t(x_n - \bar{x})^t u_j$$

$$var(\tilde{x_n}) = \lambda_i$$

Different PCA Approximation for M = 1, M = 10, M =50, M = 250

Figure 12.5 An original example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.

From Bishop Chap. 12

$$\widetilde{X_n} = \bar{x} + U_M U_M^t (x_n - \bar{x})$$

Visualization of Mean and Eigenvectors
The image can be represented by sum of mean and the linear combinations of eigenvectors

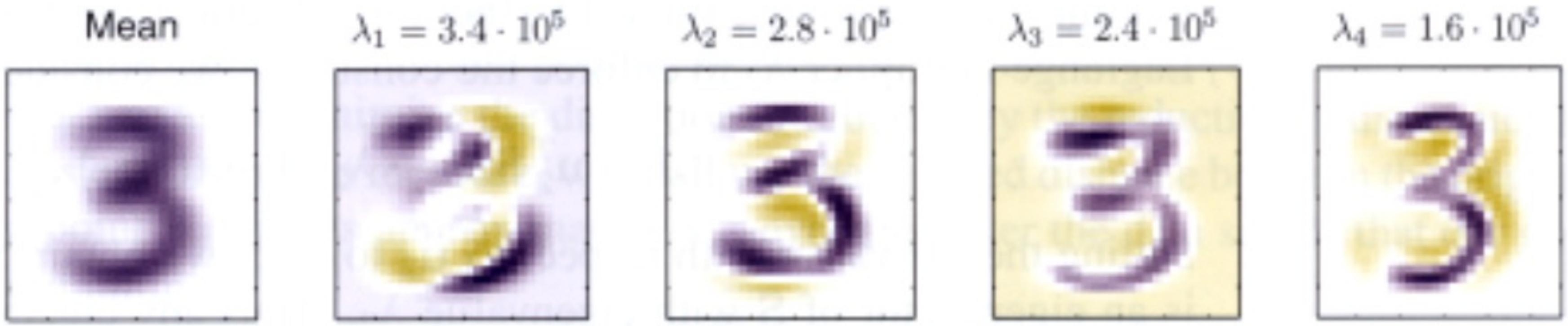


Figure 12.3 The mean vector $\overline{\mathbf{x}}$ along with the first four PCA eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_4$ for the off-line digits data set, together with the corresponding eigenvalues.

From Bishop Chap. 12

$$\widetilde{X}_n = \bar{x} + \underbrace{U_M \underbrace{U_M^t(x_n - \bar{x})}_{\text{A vector}}}_{\text{The linear combination of eigenvectors}}$$

PCA Applications

- Compression (small variance dimension does not help in learning)

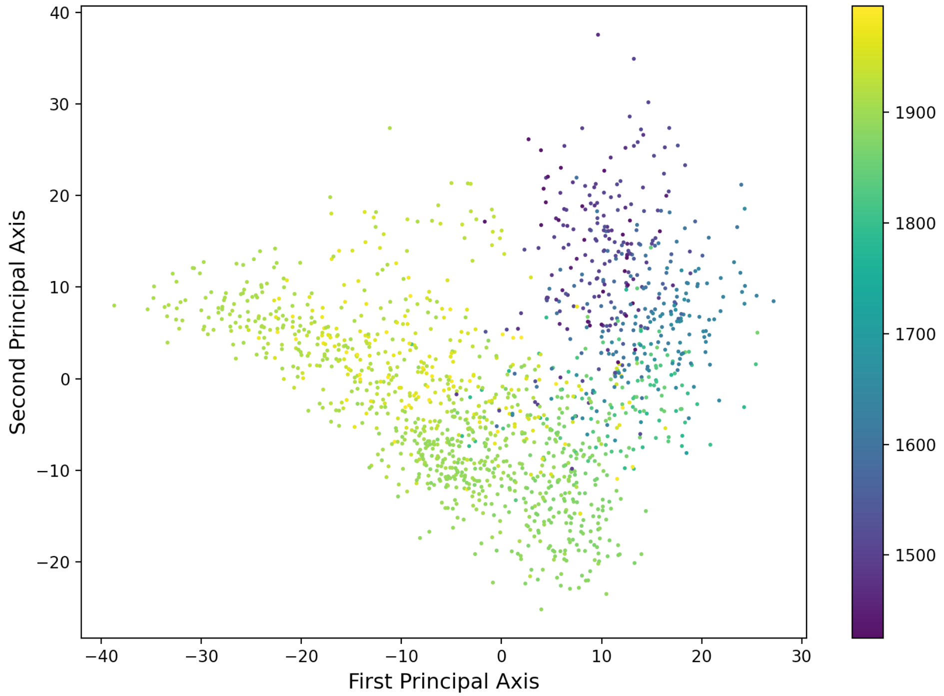
$$\widetilde{X_n} = \bar{x} + U_M U_M^t (x_n - \bar{x})$$

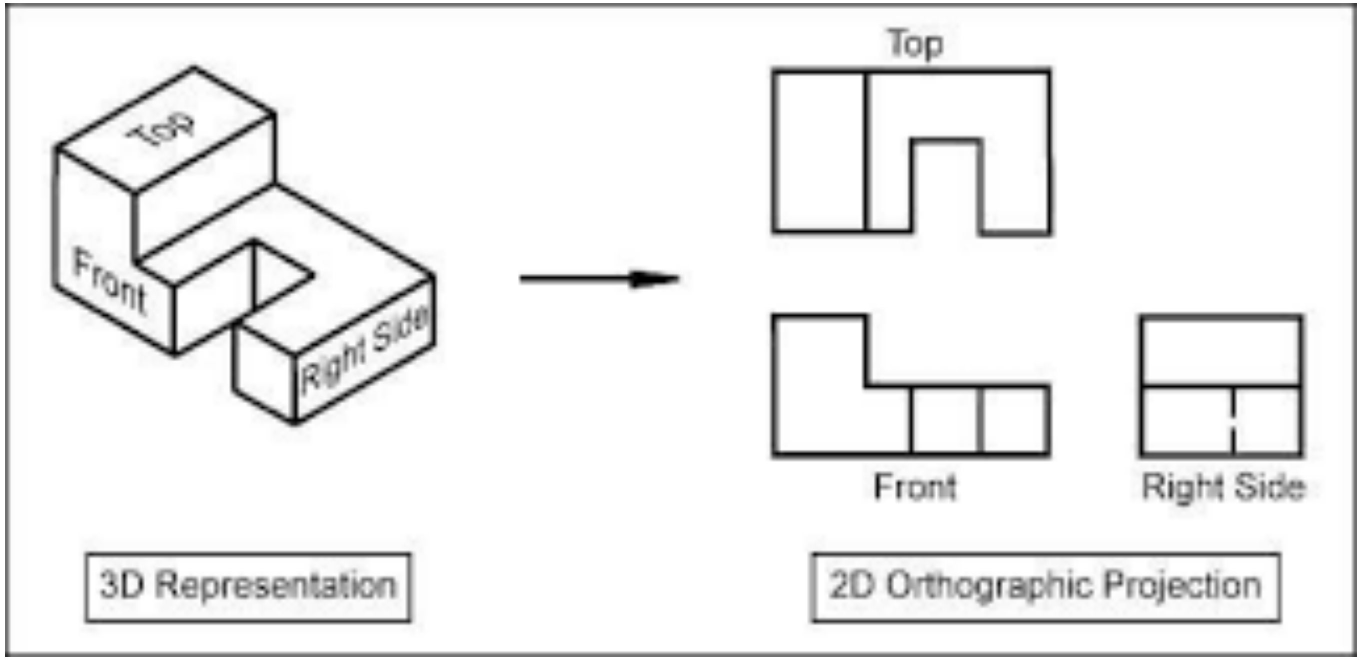
- Whitening (Rotation)

$$\tilde{x_n} = \Lambda^{-\frac{1}{2}} U_M^t (x_n - \bar{x})$$

PCA Applications

- Visualization (1) (the projection of high dimensional data to 3D or 2D)





The last hidden layer embedding of a Deep-CNN Style Classifier is projected to the top principal axes (the eigenvectors corresponding to the first and second largest eigenvalues). The samples are color-coded by year of made.

PCA Applications

- Visualization (2) (projection of high dimensional data to 3D or 2D)

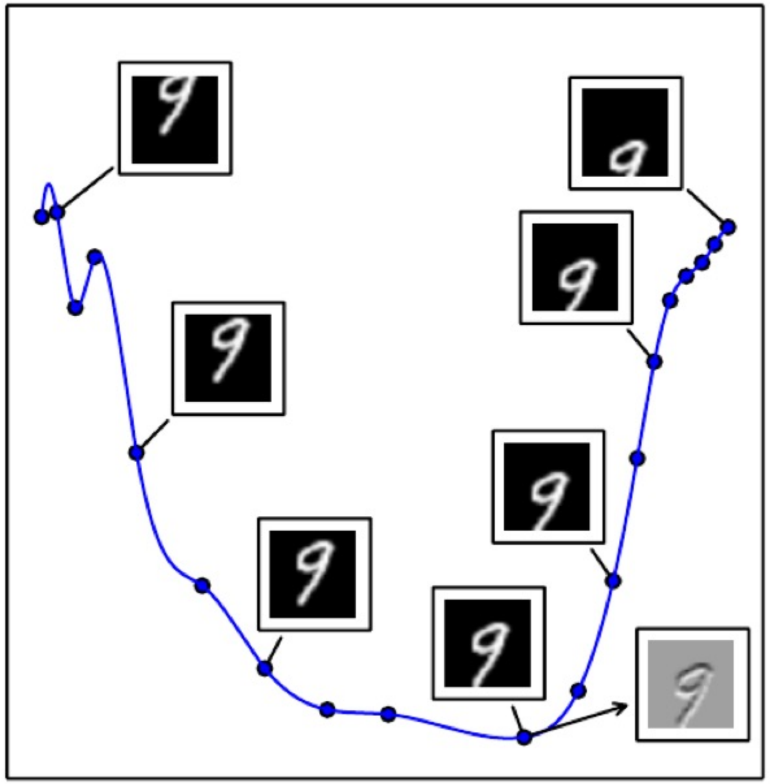


Figure 14. 6 from Deep Learning by Ian Goodfellow

One dimensional manifold that traces out a curved path for vertical shift of digit “9”. The manifold in the high dimensional space is projected into 2D.