CS 189/289

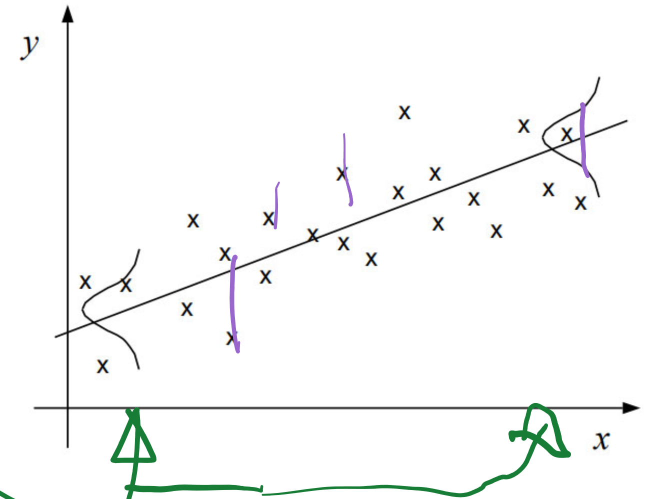
Today's lecture:

Linear regression (MLE + conditional Gaussians)

Assigned reading: 4-4.1.4

Regression

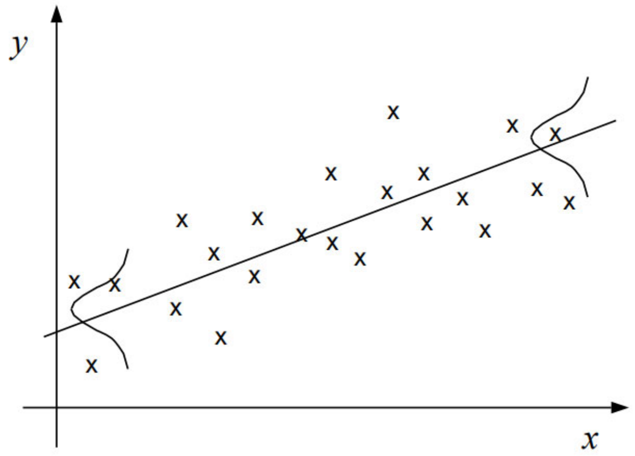
- *Supervised learning*: data pairs $D = \{(x_i, y_i)\}$, where, x_i may be discrete and/or continuous.
- *Regression:* label, y_i, is a real-valued, e.g., $y_i \in \mathbb{R}$.
- Formally, want $p(y|x)$, the conditional pdf.
- "Point" prediction is then $\hat{y} = E_Y[p(Y|X = x)]$.



error

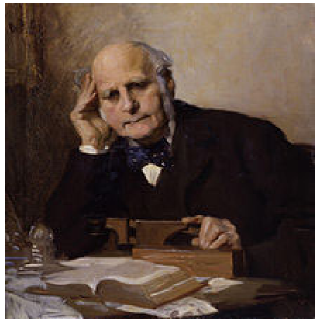
Regression examples

- Covid infection rates from zip code and vaccination rate, etc.
- How much a particular protein will bind to a drug target.
- A person’s blood pressure from their genetics.
- Tracking - object location in video at the next time-step.
- Housing prices, crime rates, stock prices, *etc*.
- Earliest regression: Legendre in 1805, and Gauss in 1809, both estimating orbits of bodies about the sun.

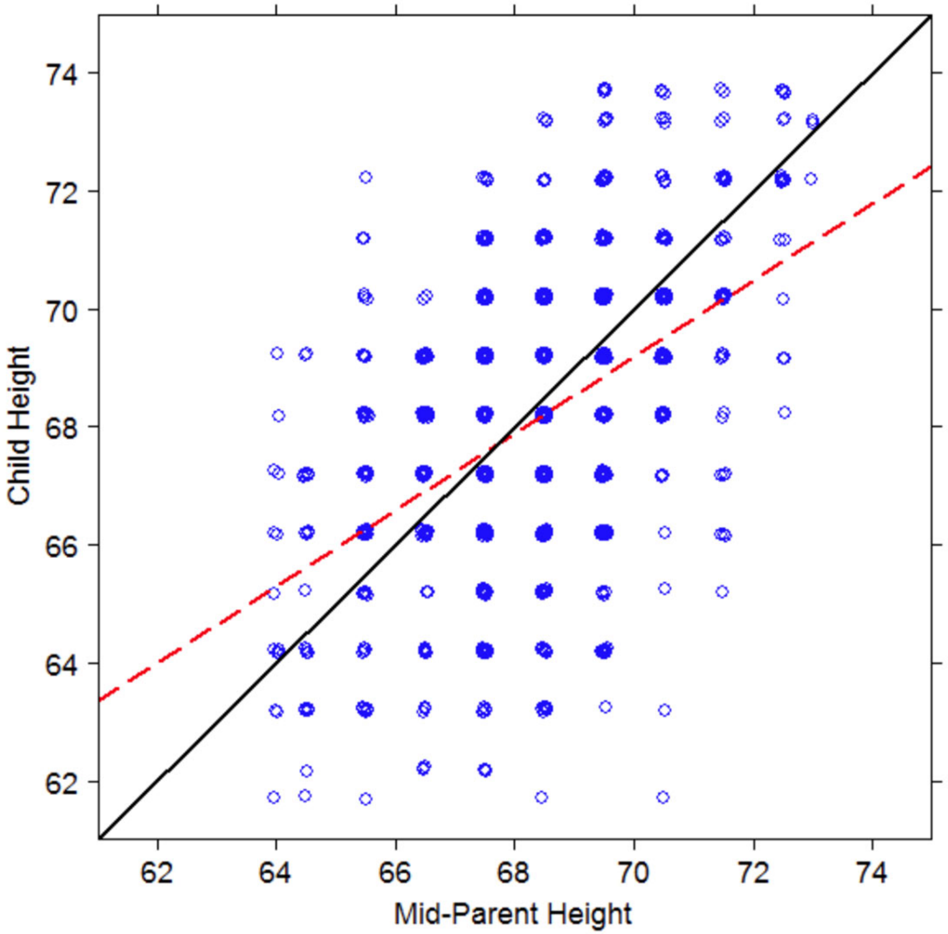


History of the term "Regression"

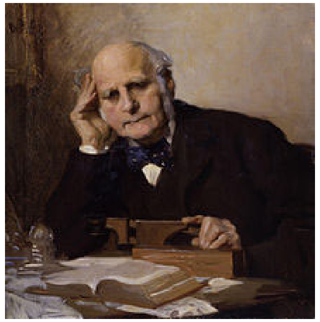
Sir Francis Galton (1822-1911) *"regression to the mean"*.



"It appeared from these experiments that the offspring did not tend to resemble their parents in size, but ***always to be more mediocre than they*** *– to be smaller than the parents, if the parents were large; to be larger than the parents, if the parents were small."*



History of the term "Regression"

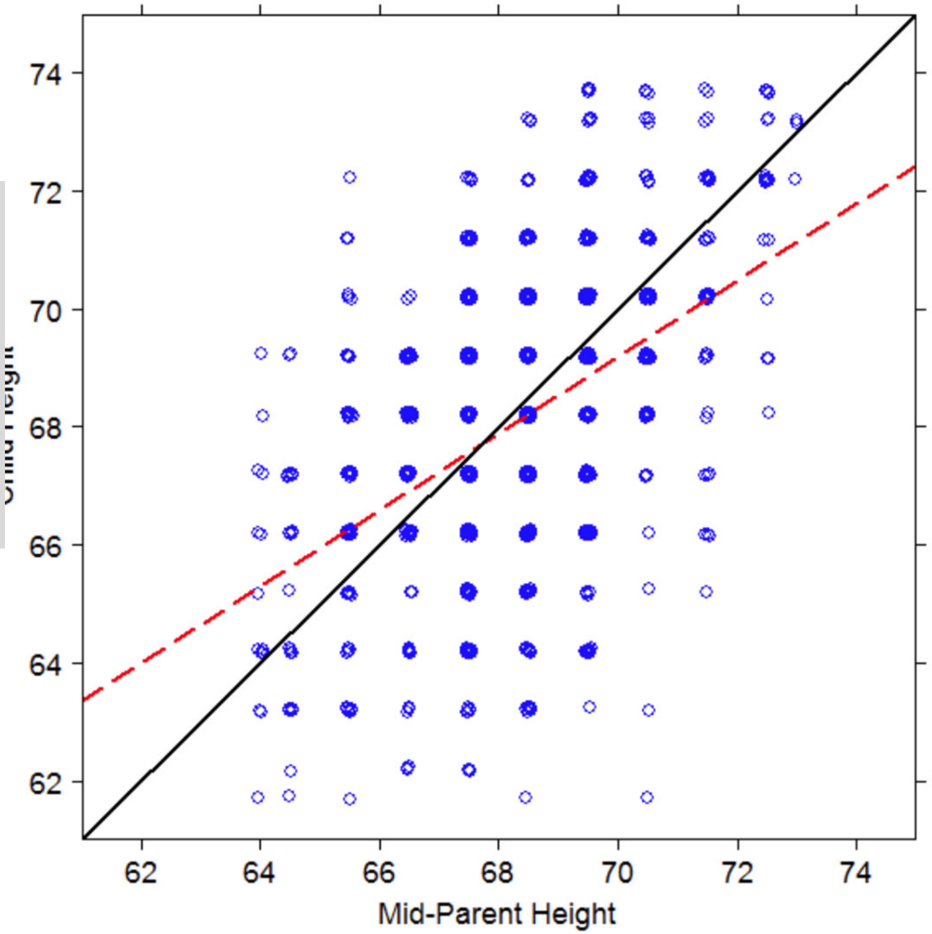


Sir Francis Galton (1822-1911) *"regression to the mean"*.

"It appeared from these experiments that the

Don't confuse this meaning of the term "regression" with that in ML/statistics.

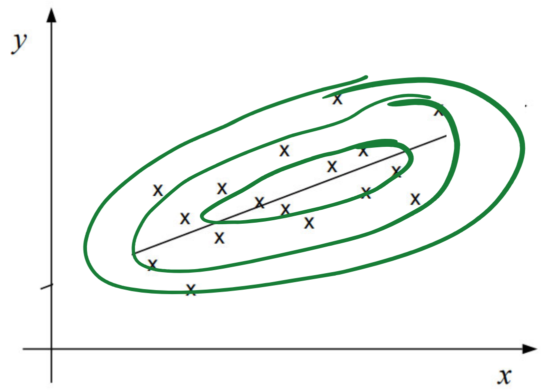
smaller than the parents, if the parents were large; to be larger than the parents, if the parents were small."



Possible Regression Tactics (goal: estimate $p(y|x)$)

- Our data are drawn from some distribution, $(X, Y) \sim p(x, y)$.
- What are possible strategies to estimate $p(y|x)$?

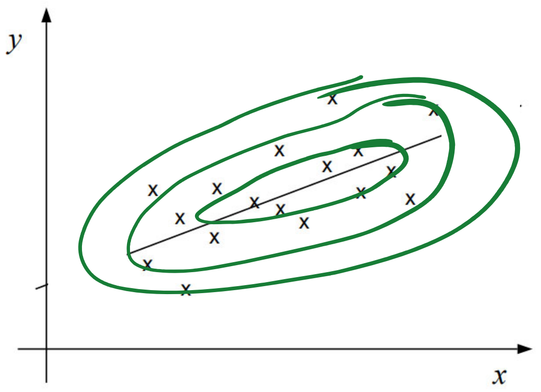
1. Estimate $p(x, y|\theta)$ e.g. MVG for RVs X, Y, and then use fitted model to compute $p(y|x, \hat{\theta})$



Possible Regression Tactics (goal: estimate $p(y|x)$)

- Our data are drawn from some distribution, $(X, Y) \sim p(x, y)$.
- What are possible strategies to estimate $p(y|x)$?

1. Estimate $p(x, y|\theta)$ e.g. MVG for RVs X, Y, and then use fitted model to compute $p(y|x, \hat{\theta}) = \frac{p(y,x|\hat{\theta})}{p(x|\hat{\theta})} = \frac{p(y,x|\hat{\theta})}{\int_y p(y, x|\hat{\theta}) dy}$.

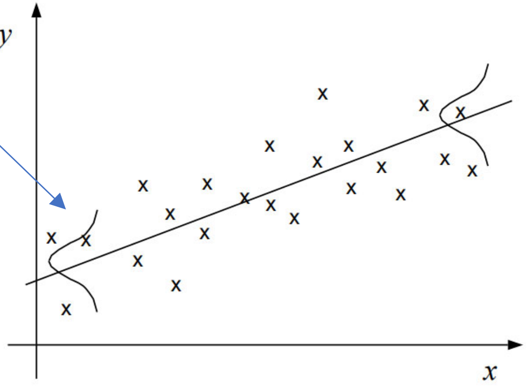


Possible Regression Tactics (goal: estimate $p(y|x)$)

- Our data are drawn from some distribution, $(X_i, Y_i) \sim p(x, y)$.
- What are possible strategies to estimate $p(y|x)$?

1. Estimate $p(x, y|\theta)$ e.g. MVG for RVs X, Y, and then use fitted model to compute $p(y|x, \hat{\theta}) = \frac{p(y,x|\hat{\theta})}{p(x|\hat{\theta})} = \frac{p(y,x|\hat{\theta})}{\int_y p(y, x|\hat{\theta})dy}$.
2. Consider the inputs to be fixed, and model only the output as a RV. That is, directly model the conditional $p(y|x, \hat{\theta})$.

"*generative*", vs. "*discriminative*"



Linear Regression

- Takes the discriminative approach.
- Predictions are a linear function of the parameters:

$$\hat{y} = E_Y[p(y|x)] = w^T x + w_0, \text{ for } w, x \in \mathbb{R}^d.$$

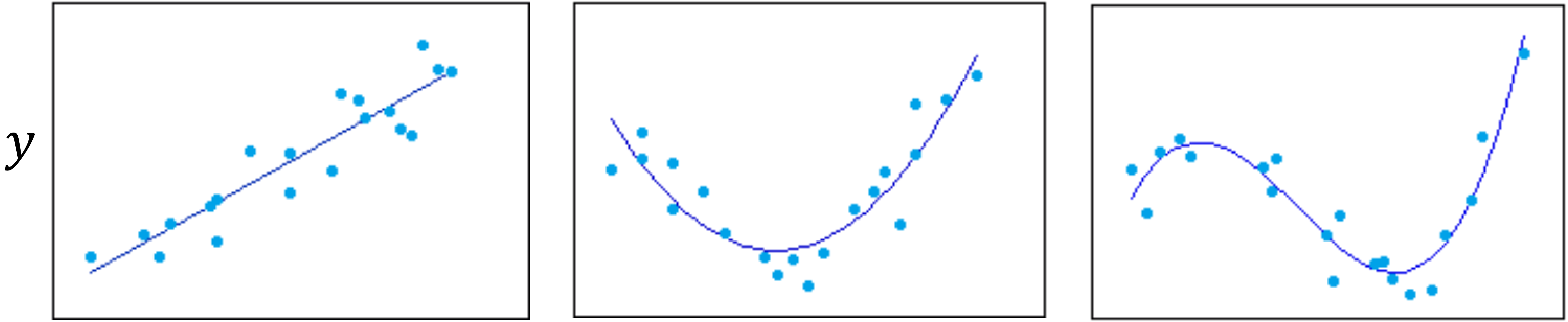
- w_0 is called the "offset"/"bias"/"intercept".

Book-keeping trick: instead of a bias, we can make an extra feature that is always 1, now use $x' = [x, 1]$ and $\hat{y} = w^T x'$.

How useful can a linear model be?!

Which of these curves could have been modelled by linear regression?

$$\hat{y} = E_Y[p(y|x)] = w^T x + w_0, \text{ for } w, x \in \mathbb{R}^d$$



https://statisticsbyjim.com/regression/curve-fitting-linear-nonlinear-regression/

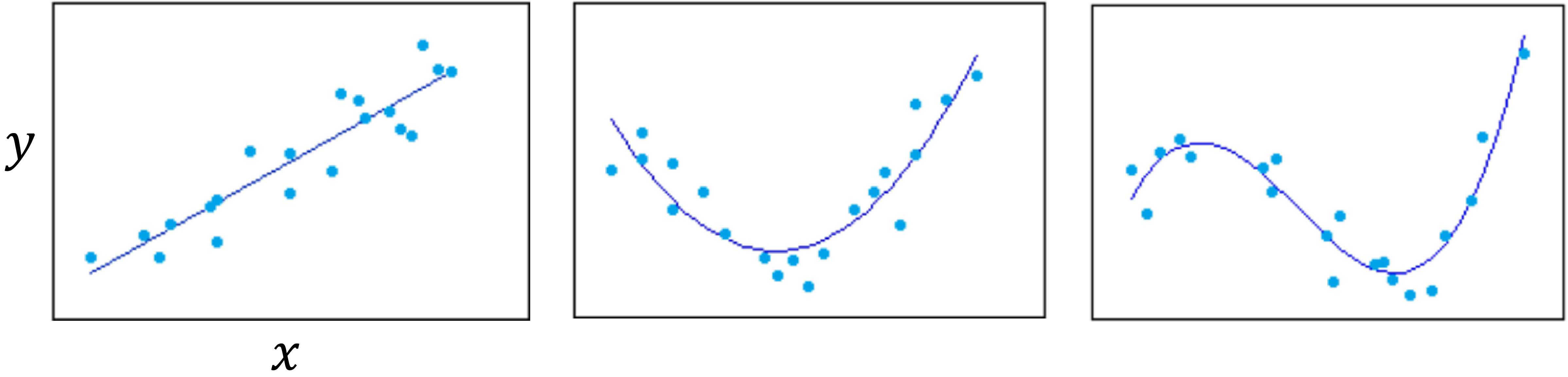
How useful can a linear model be?!

$w, x \in \mathbb{R}^1$

$\hat{y} = w^T x$ — Linear

$\hat{y} = w^T [x, x^2]$ — Quadratic

$\hat{y} = w^T [x, x^2, x^3]$ — Cubic



For full generality, $x \in \mathbb{R}^D$ need the cross-terms and bias terms for arbitrary polynomial, e.g., quadratic $[1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$.

https://statisticsbyjim.com/regression/curve-fitting-linear-nonlinear-regression/

Basis expansion of raw input space

$$x \in \mathbb{R}^{d=2} = [x_1, x_2] \to [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2] \in \mathbb{R}^{k=6}$$

Polynomial expansion of order 2 (i.e. quadratic)

- Denote *basis expansion* of the (raw) input features: $\Phi(x): \mathbb{R}^d \to \mathbb{R}^k$.

For $d = 1$, some polynomial expansions are:

- A quadratic expansion ($k = 2$), $\Phi(x) = [1, x, x^2]$.
- A cubic expansion ($k = 3$), $\Phi(x) = [1, x, x^2, x^3]$.

Identity basis expansion, $\Phi(x) = x$, and $k = d$.

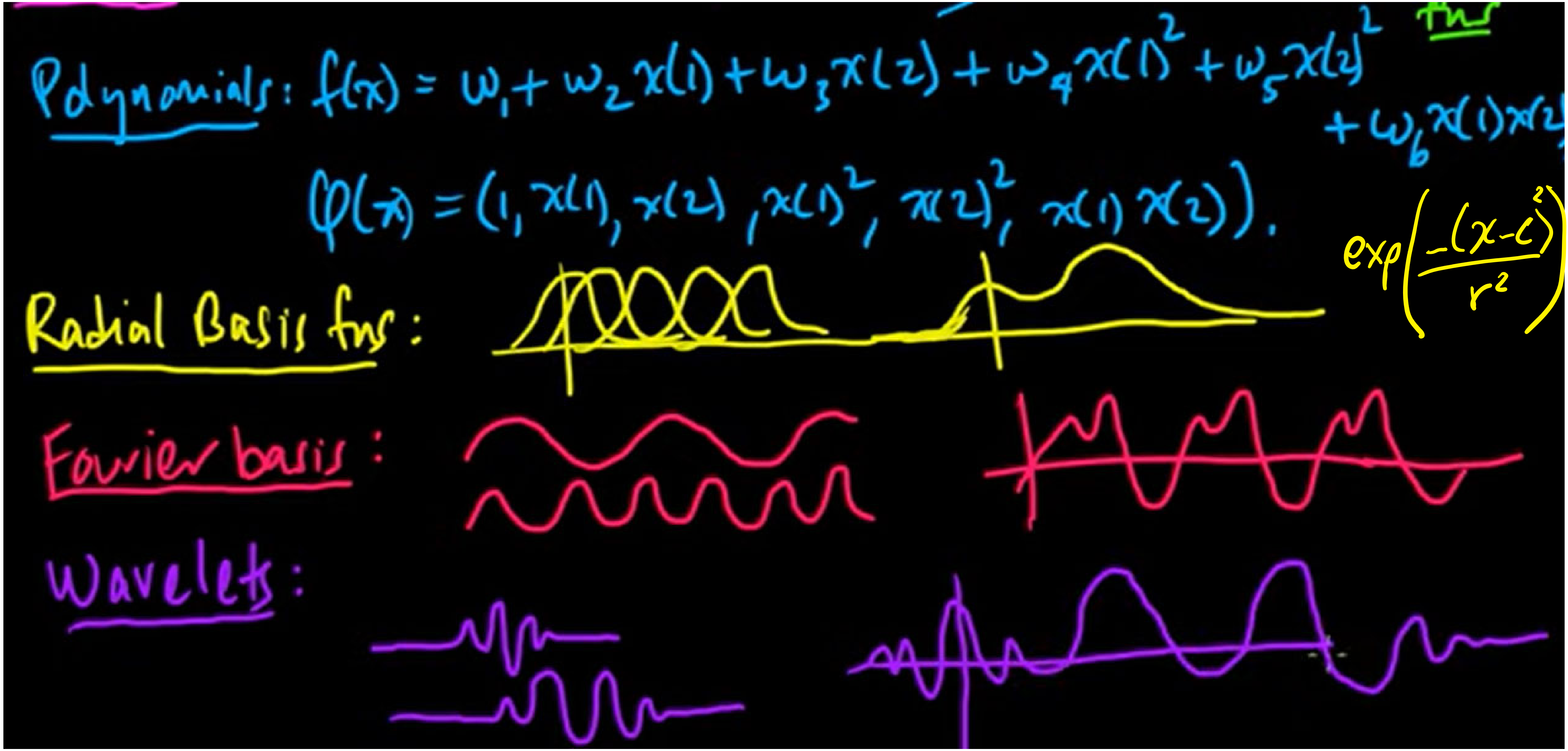
Basis expansion of raw input space

Basis functions are pre-determined, so just a notational change:

$$\hat{y} = E_Y[p(y|x)] = w^T\Phi(x), \text{ for } w \in \mathbb{R}^k, x \in \mathbb{R}^d$$

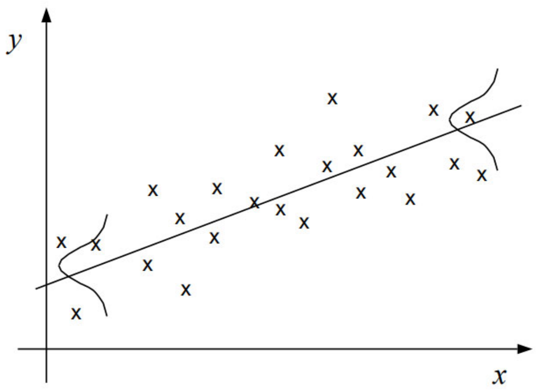
In this lecture, for simplicity of notation, we will assume that this expansion has already been done, and just write $\hat{y} = w^T x$.

Many basis possible functions!



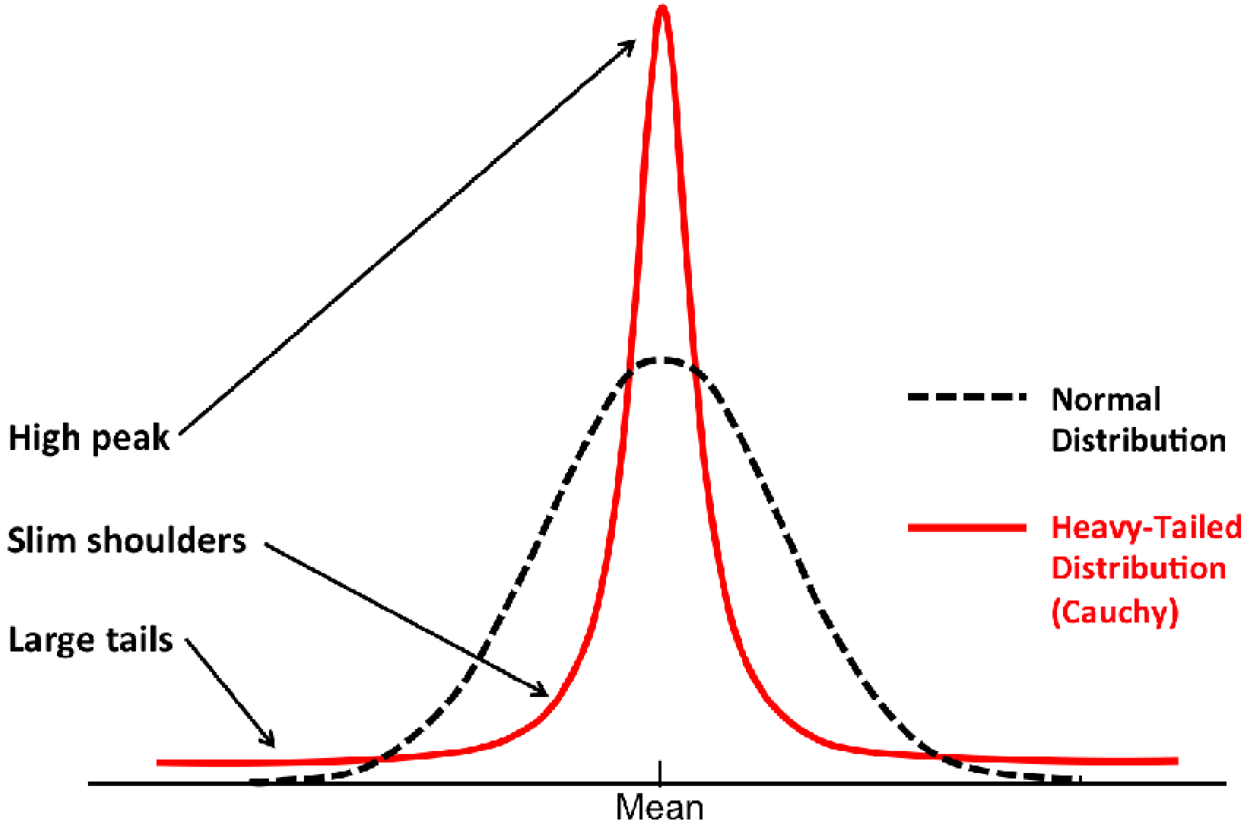
Specific form of linear regression

- So far we said linear regression is $\hat{y} = E_Y[p(y|x)] = E[p(y|x)] = w^T x$.
- But what do we use for $p(y|x)$?
- Standard linear regression uses a Gaussian $p(y|x) = N(y|w^T x, \sigma^2)$.
- Equivalent to $Y = w^T x + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$.
- Which is equivalent to $Y - w^T x = \epsilon \sim N(0, \sigma^2)$.
- Alternate forms give “heavier tails” to the distribution of the “residual”.



Aside: heavy-tailed distribution

- More of the mass lies further from the center of mass.
- Heavy-tailed noise better models outliers than a Gaussian.



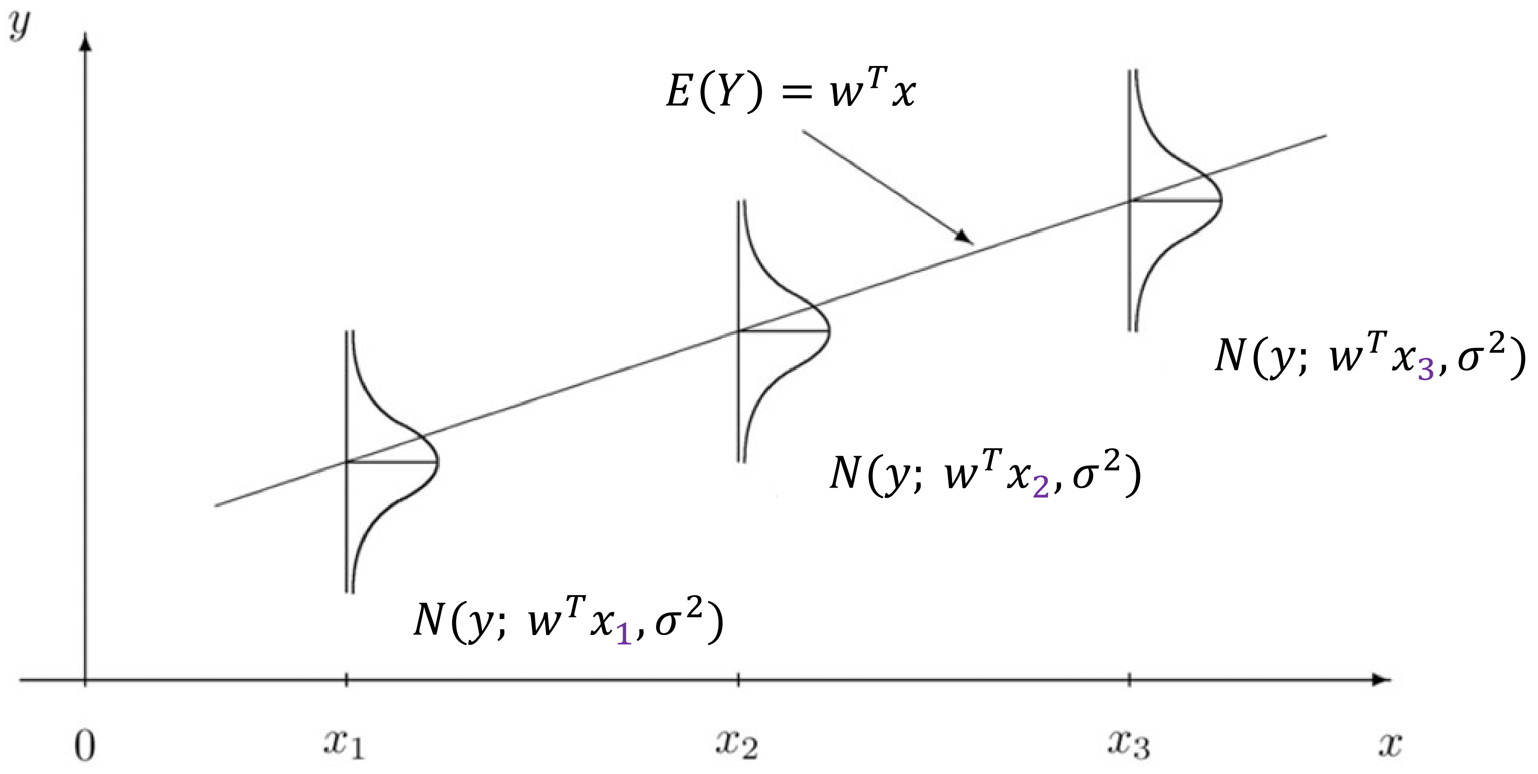
$$Y - w^T x = \epsilon \sim N(0, \sigma^2)$$

$$Y - w^T x = \epsilon \sim Cauchy(0, \sigma^2)$$

https://www.semanticscholar.org/paper/The-Normal-Distribution-.-.-.-or-Another-Brandenburger-Weston/b0a227e36d655c0dceaeb774225e0b4fb6cda8ed/figure/2

Gaussian linear regression, $p(y|x) = N(y|w^T x, \sigma^2)$

For every value $X = x$, the target variable, Y, takes on a Gaussian distribution with the same variance, σ^2:



https://sdcastillo.github.io/PA-R-Study-Manual/introduction-to-modeling.html

"Training" a Gaussian linear regression model

How will we fit the regression model, $p(y|x) = N\left(y \middle| w^T x, \sigma^2\right)$?

MLE: $\theta_{MLE} = \left(w_{MLE}, \sigma^2_{MLE}\right) = \underset{(w,\sigma^2)}{\arg\max} \log p\left(D = \{(x_i, y_i)_{i=1}^n\} \middle| \theta\right)$

$$= \underset{(w,\sigma^2)}{\arg\max} \textstyle\sum_{i=1}^n log\, p(y_i|x_i, \theta)$$

$$= \underset{(w,\sigma^2)}{\arg\max} \textstyle\sum_{i=1}^n log\, N(y_i|w^T x_i, \sigma^2)$$

$$= \underset{(w,\sigma^2)}{\arg\max} \textstyle\sum_{i=1}^n log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^n \left(y_i - w^T x_i\right)^2$$

$$= \underset{(w,\sigma^2)}{\arg\max}\, n\, log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \qquad - \frac{1}{2\sigma^2} \sum_{i=1}^n \left(y_i - w^T x_i\right)^2$$

"Training" a Gaussian linear regression model

$$p(y|x) = N\left(y \middle| w^T x, \sigma^2\right)$$

Note, if we ignore $\boldsymbol{\sigma^2_{MLE}}$

$$\left(w_{MLE}, \cancel{\sigma^2_{MLE}}\right) = \arg\max_{(w,\cancel{\sigma^2})} n \, log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - w^T x_i)^2$$

... then estimating $\boldsymbol{w}$ above is the same as

$$= \arg\min_{w} \sum_{i=1}^{n}\left(y_i - w^T x_i\right)^2$$

$$= \arg\min_{w} \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

"least squares" loss function!

i.e. least squares arises naturally from conditional Gaussian MLE

"Training" a Gaussian linear regression model

$$\mathrm{w}_{\mathrm{MLE}} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2$$

Lets re-write this loss in "vectorized" form:

First, define:

$$\begin{pmatrix} y_1 - w^T x_1 \\ \vdots \\ y_n - w^T x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} x_1^T w \\ \vdots \\ x_n^T w \end{pmatrix} = y - \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} w = y - Aw$$

$$w^T x_i = x_i^T w$$

Scalar

"Training" a Gaussian linear regression model

$A \in \mathbb{R}^{n \times d}$

$$w_{\text{MLE}} = \arg\min_{w} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

Lets re-write this loss in "vectorized" form:

First, define:

$$\begin{pmatrix} y_1 - w^T x_1 \\ \vdots \\ y_n - w^T x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} x_1^T w \\ \vdots \\ x_n^T w \end{pmatrix} = y - \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} w = y - Aw$$

$$\underbrace{w^T x_i}_{\text{scalar}} = \underbrace{x_i^T w}_{\text{scalar}}$$

A is called the "design matrix"

"Training" a Gaussian linear regression model

$$w_{\text{MLE}} = \arg\min_{w} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

$A \in \mathbb{R}^{n \times d}$

Lets re-write this loss in "vectorized" form:

First, define:

$$\begin{pmatrix} y_1 - w^T x_1 \\ \vdots \\ y_n - w^T x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} x_1^T w \\ \vdots \\ x_n^T w \end{pmatrix} = y - \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} w = y - Aw$$

$$\underbrace{w^T x_i}_{\text{scalar}} = \underbrace{x_i^T w}_{\text{scalar}}$$

A is called the "design matrix"

Then, we can re-write the loss as

$$\arg\min_{w} (y - Aw)^T (y - Aw)$$

$$= \arg\min_{w} \|y - Aw\|_2^{\,2}$$

"Training" a Gaussian linear regression model

So want to minimize

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$$

A

$$\mathcal{L} = \|y - Aw\|_2{}^2 \qquad (y \in \mathbb{R}^n, A \in \mathbb{R}^{n\times d}, w \in \mathbb{R}^{d\times 1})$$

$$= (y - Aw)^T(y - Aw)$$

$$= \left(y^T - (Aw)^T\right)(y - Aw)$$

$$= y^Ty - w^TA^Ty - y^TAw + w^TA^TAw$$

$$= y^Ty - 2w^TA^Ty + w^TA^TAw$$

To minimize, we want to set $\frac{\partial \mathcal{L}}{\partial w} = \mathbf{0}$.

Most easily achieved by using the rules of vector calculus, so lets do a quick refresher.

$$p(y|x) = N\left(y \middle| w^Tx, \sigma^2\right)$$

Refresher on vector calculus

Some "rules" for taking gradients with respect to vectors.

- e.g., for vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{d\times 1}$, such that $\boldsymbol{a}^T\boldsymbol{b} \in \mathbb{R}$,

$$\frac{\partial(a^T b)}{\partial a_j} = \frac{\partial(a_1 b_1 + a_2 b_2 + \cdots a_d b_d)}{\partial a_j} = b_j \in \mathbb{R}$$

Thus,

$$\frac{\partial(a^T b)}{\partial a} = \frac{\partial(a_1 b_1 + a_2 b_2 + \cdots a_d b_d)}{\partial a} = b \in \mathbb{R}^{d\times 1}$$

(not true for $\boldsymbol{a}\boldsymbol{b}^T$ which is a matrix, be careful!)

$$= \frac{\partial(b^T a)}{\partial a}$$

Useful cheat sheet: https://cs.nyu.edu/~roweis/notes/matrixid.pdf

Refresher on vector calculus

For vector $x \in \mathbb{R}^{d\times 1}$, and matrix $\Sigma \in \mathbb{R}^{d\times d}$

$$\frac{\partial x^T \Sigma x}{\partial x} = \left(\Sigma + \Sigma^T\right) x$$

Thus, if Σ is symmetric such that $\Sigma = \Sigma^T$ then

$$\frac{\partial x^T \Sigma x}{\partial x} = 2\Sigma x$$

(similar to the scalar version: $\frac{\partial (ax^2)}{\partial x} = 2ax$)

"Training" a Gaussian linear regression model

So want to minimize

$(y \in \mathbb{R}^n, A \in \mathbb{R}^{n\times d}, w \in \mathbb{R}^{d\times 1})$

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A \in \mathbb{R}^{n\times d}$$

$$\mathcal{L} = (y - Aw)^T(y - Aw)$$
$$= \left(y^T - (Aw)^T\right)(y - Aw)$$
$$= y^T y - w^T A^T y - y^T A w + w^T A^T A w$$
$$= y^T y - 2 w^T A^T y + w^T A^T A w$$

To minimize, we want to set $\frac{\partial \mathcal{L}}{\partial w} = \mathbf{0}$.

"Training" a Gaussian linear regression model

So want to minimize

$(y \in \mathbb{R}^n, A \in \mathbb{R}^{n \times d}, w \in \mathbb{R}^{d \times 1})$

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A \in \mathbb{R}^{n \times d}$$

$$\begin{aligned} \mathcal{L} &= (y - Aw)^T (y - Aw) \\ &= \left(y^T - (Aw)^T\right)(y - Aw) \\ &= y^T y - w^T A^T y - y^T A w + w^T A^T A w \\ &= y^T y - 2 w^T A^T y + w^T A^T A w \end{aligned}$$

To minimize, we want to set $\frac{\partial \mathcal{L}}{\partial w} = \mathbf{0}$.

$$\nabla_w \mathcal{L} = \begin{bmatrix} \partial \mathcal{L} / \partial w_1 \\ \partial \mathcal{L} / \omega_2 \\ \vdots \\ \partial \mathcal{L} / \omega_d \end{bmatrix} = \left[-2A^T y + 2A^T A w \right]$$

$$= \Big(\underbrace{-A^T y + A^T A w}_{d \times 1} \Big) \cdot 2$$

"Training" a Gaussian linear regression model

"left pseudo-inverse" of A

$(A^TA)^{-1}A^T * A = I$

$A^+ A = I$

$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A \in \mathbb{R}^{n \times d}$

So want to minimize

$$
\begin{aligned}
\mathcal{L} &= (y - Aw)^T(y - Aw) \\
&= \left(y^T - (Aw)^T\right)(y - Aw) \\
&= y^Ty - w^TA^Ty - y^TAw + w^TA^TAw \\
&= y^Ty - 2w^TA^Ty + w^TA^TAw
\end{aligned}
$$

To minimize, we want to set $\frac{\partial \mathcal{L}}{\partial w} = 0$.

$$
\nabla_w \mathcal{L} = \begin{bmatrix} \partial\mathcal{L}/\partial w_1 \\ \partial\mathcal{L}/\partial w_2 \\ \vdots \\ \partial\mathcal{L}/\partial w_d \end{bmatrix} = \left[-2A^Ty + 2A^TAw\right] = \underbrace{(-A^Ty + A^TAw)}_{d\times 1} \cdot 2
$$

set to 0

rectangular so no A^{-1}!

$$-A^Ty + A^TAw = 0$$

$$A^TAw = A^Ty$$

$$\boxed{w = (A^TA)^{-1}A^Ty}$$

"Training" a Gaussian linear regression model

"left pseudo-inverse" of A

$(A^T A)^{-1} A^T * A = I$

- A^+ is the Moore-Penrose inverse.
- Can be used even when A is not full rank.
- *i.e.*, when A has dependent feature vectors.
- Will yield w_{MLE} with min. 2-norm, $\| w_{MLE} \|_2$

$A^+ A = I$

$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A \in \mathbb{R}^{n \times d}$

rectangular so no A^{-1}!

Related to spectral decomposition from last class:

Inverses and square roots

- If $A = \Phi D \Phi^T$

Then $A^{-1} = \Phi D^{-1} \Phi^T$

where $D^{-1} = \begin{bmatrix} \lambda_1^{-1} & & & \\ & \lambda_2^{-1} & & \\ & & \ddots & \\ & & & \lambda_n^{-1} \end{bmatrix}$

Take reciprocal of only non-zero values, leave the rest as zeros.

$-A^T y + A^T A w = 0$

$A^T A w = A^T y$

$\boxed{w = (A^T A)^{-1} A^T y}$

"Training" a Gaussian linear regression model

- We still need to check if the critical point, $w = (A^T A)^{-1} A^T y$ is minimum of the squared error loss.
- Recall $\nabla_w \mathcal{L} = -2A^T y + 2A^T A w$
- So Hessian matrix ($\nabla_w^2 \mathcal{L}$) is $2A^T A$. When is $A^T A$ PD?
- When the features in data set are independent (when it has full rank).

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A \in \mathbb{R}^{n \times d}$$

σ^2 from MLE is just the mean squared residual, $\sigma^2 = \frac{1}{N}\sum_i (y_i - w^T x)^2$.

$$\nabla_w \mathcal{L} = \begin{bmatrix} \partial\mathcal{L}/\partial w_1 \\ \partial\mathcal{L}/\omega_2 \\ \vdots \\ \partial\mathcal{L}/\omega_d \end{bmatrix} = \left[-2A^T y + 2A^T A w\right] = \underbrace{-A^T y + A^T A w}_{d \times 1}$$

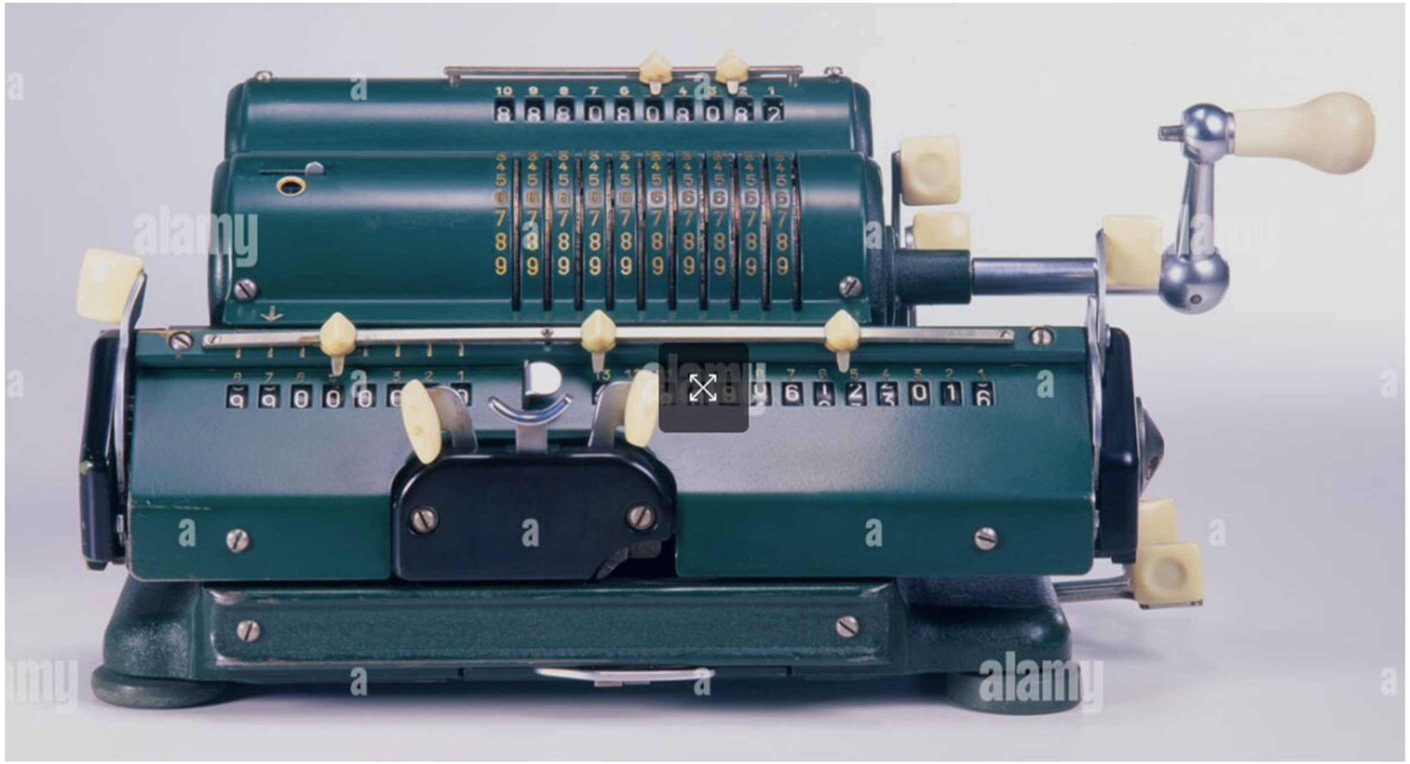
$$-A^T y + A^T A w = 0$$

$$A^T A w = A^T y$$

$$\boxed{w = (A^T A)^{-1} A^T y}$$

Regression in 1950s

Electromechanical desk "calculators" were used, and it could take up to 24 hours to receive the result from one regression on a small data set.



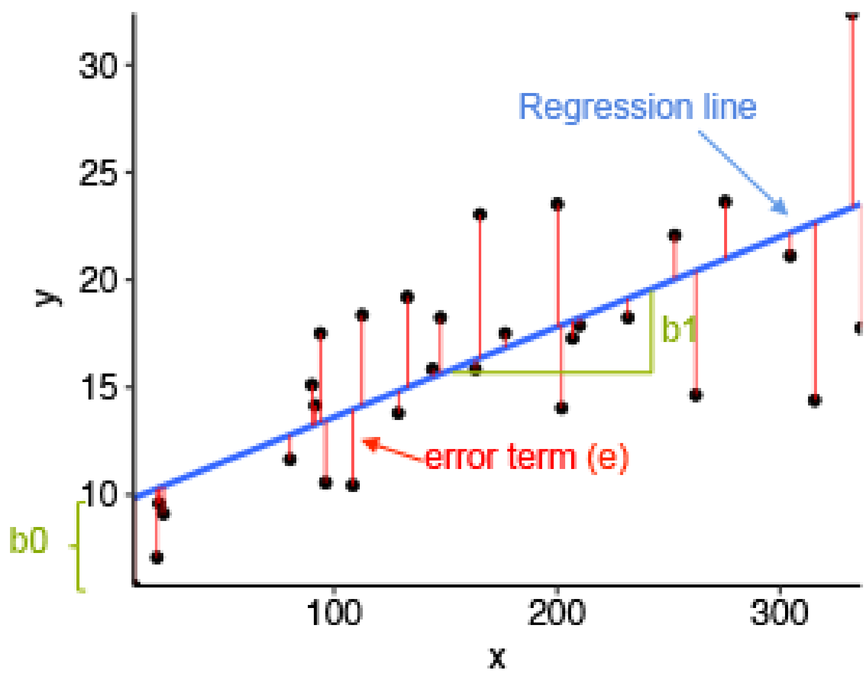
Geometric view of linear regression

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A$$

- We're trying to predict all our training data labels correctly, such that $y_i = w^T x_i$ for all $i \in [1 \dots n]$.
- In vector form, this means we're looking for

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = Aw = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} w$$

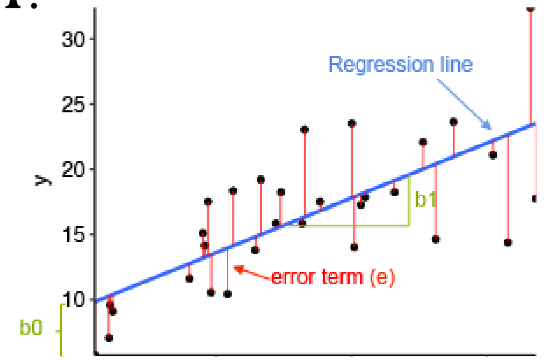
Generally, this is not possible because of noise; or incorrect model (e.g. missing some features).



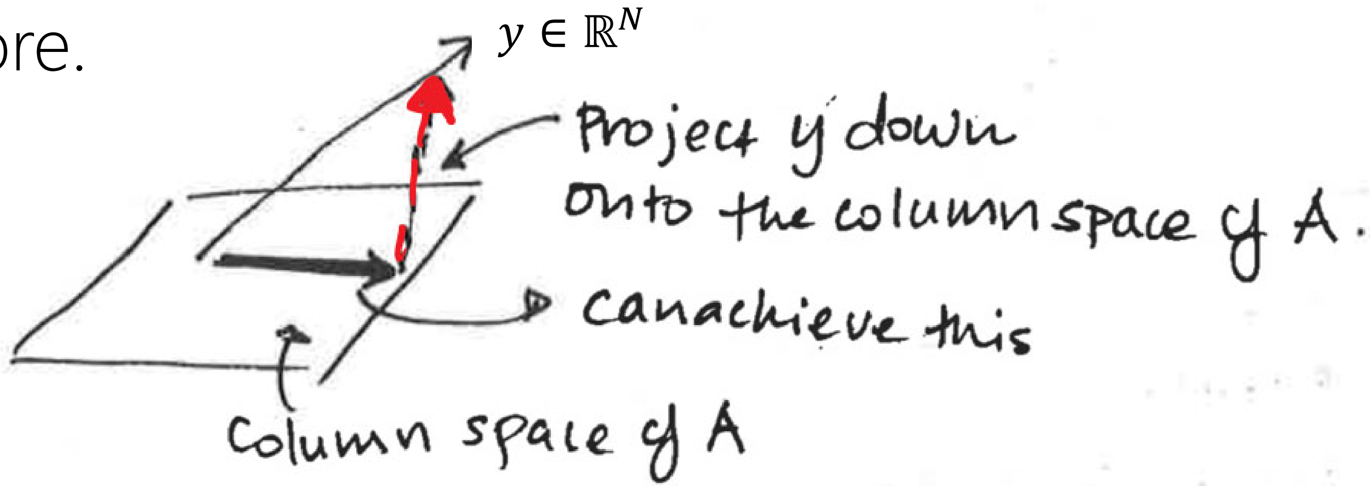
https://www.analyticsvidhya.com/blog/2021/06/25-questions-to-test-your-skills-on-linear-regression-algorithm/

Geometric view of linear regression, $y = Aw$

- So lets think about the error vector $(\mathbf{e} \equiv y - \hat{y} = y - Aw, \in \mathbb{R}^{n\times1})$.
- A "good" setting of $\boldsymbol{w}$ minimizes the magnitude of $\mathbf{e}$.
- Magnitude is minimized when $\boldsymbol{e}$ lies $\perp$ to column space of A.
- Thus we seek $\boldsymbol{w}$ such that $\boldsymbol{e}^T A = A^T\boldsymbol{e} = 0 = A^T(y - Aw)$.
- Thus $A^T y - A^T Aw = 0$.
- Thus $A^T y = A^T Aw$ (same as from MLE/least squares)!
- Thus $w = (A^T A)^{-1} A^T y$, as before.



$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = Aw = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} w$$



Using basis expansions instead of x_i

$$\begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = A$$

- $\{x_j\} \to \{\Phi(x_j)\}$?
- Just define $\mathbf{A}$ with $\Phi^T(x_j)$ instead of x_j because $\Phi(x)$ is fixed ahead of time, so its like someone just gave us different raw inputs x.

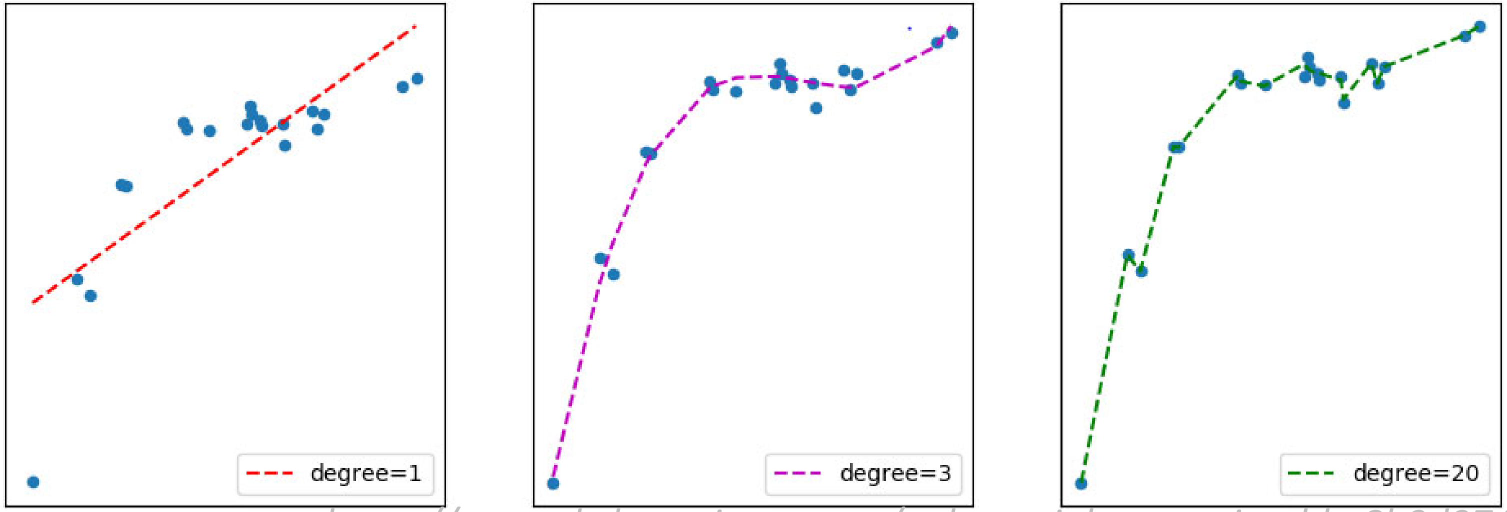
$$x = \Phi[x_1, x_2] = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2] \in \mathbb{R}$$

Polynomial expansion of order 2 (i.e. quadratic)

What can go wrong in linear regression

As we add higher and higher order polynomials, a few things happen:

1. # features, d gets bigger and bigger.
2. For $d \geq n$ can perfectly fit any data (*i.e.*, polynomials are a complete basis).
3. Even when we don't perfectly fit the training data, we are still in danger of overfitting (worse prediction on test set). Our goal is not to fit a line through the training data exactly, it is to do well on unseen test cases!



https://towardsdatascience.com/polynomial-regression-bbe8b9d97491

What can go wrong in linear regression

Two main categories of fixes:

1. Remove features until the problem is well-behaved “feature selection”).
2. Leave the features as they are, but *add constraints to the system* to “tighten it up” (aka “regularization”)—next class.

NB: Moore-Penrose inverse is not a general fix for ML models, only works for linear regression.

A thought experiment

Consider:

- Use MLE on data, $D = \{x_i, y_i\}$ to get $\hat{y}_i = p_\theta(y|\Phi(x))$ using linear regression.
- Assume an abundance of data (millions of data points), and only 100 parameters.
- Suppose get accuracy $\mp$\$1000 of sale price when applying to held out part of our data.
- Can we assume this model will get $\mp$\$1000 on any test set that may come in the future?

Actual *vs.* predicted sale price of house

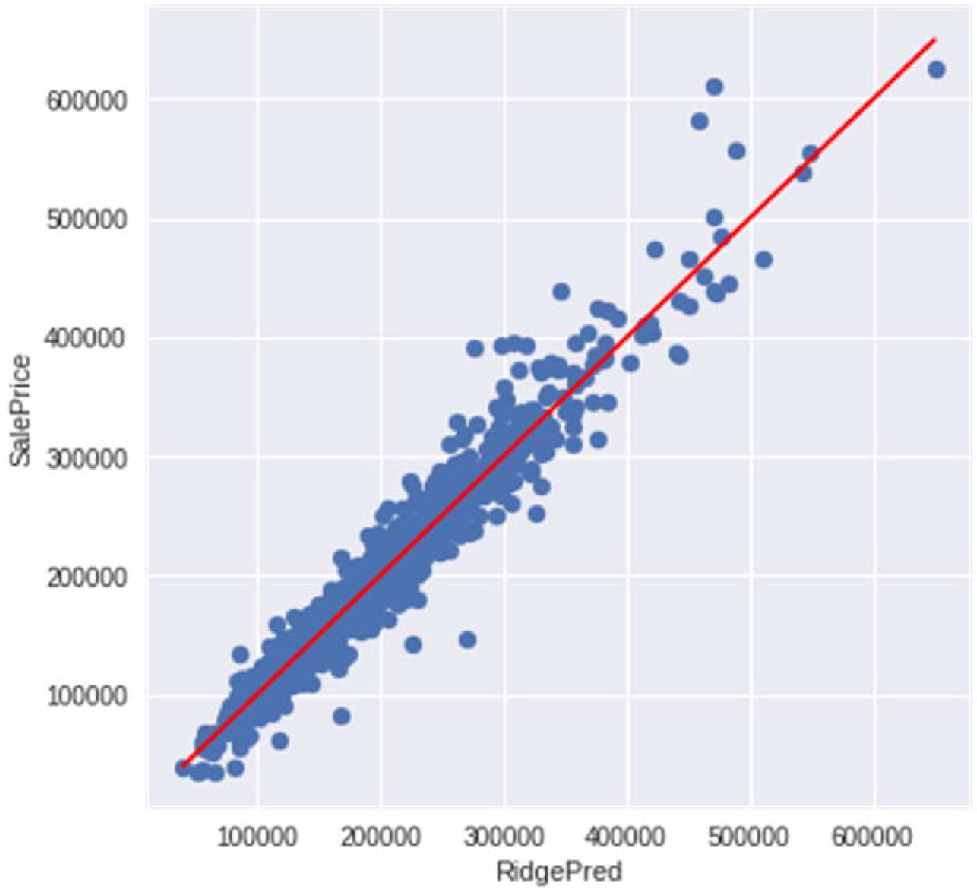


Fig. 4 Ridge Prediction for Training Data.

Causation vs correlation

Breakingviews

Zillow’s failed house flipping

Reuters

WSJ NOV. 2021 : *“The company expects to record losses of more than $500 million from home-flipping by the end of this year and is laying off a quarter of its staff.”*

Actual *vs.* predicted sale price of house

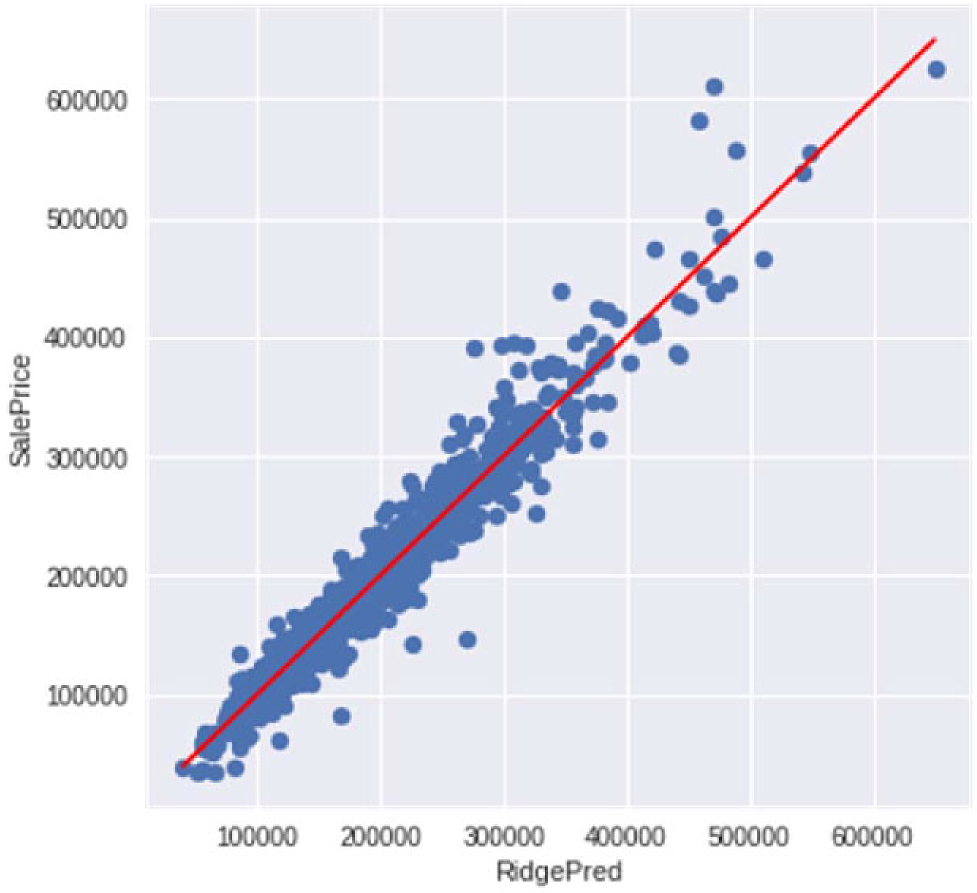


Fig. 4 Ridge Prediction for Training Data.