Applied Machine Learning

Linear Regression

Oumar Kaba

McGill
School of Computer Science

COMP 551 (Fall 2025)

Admin

- Assign yourself to a group for the assignment before the 13th. After that you will be assigned randomly
- Assignment 1 is out, deadline is September 30th. Start early!
- The quizz for this week is out
- To help you understand the material we have:
  - Code reviews for each class
  - Tutorial sessions

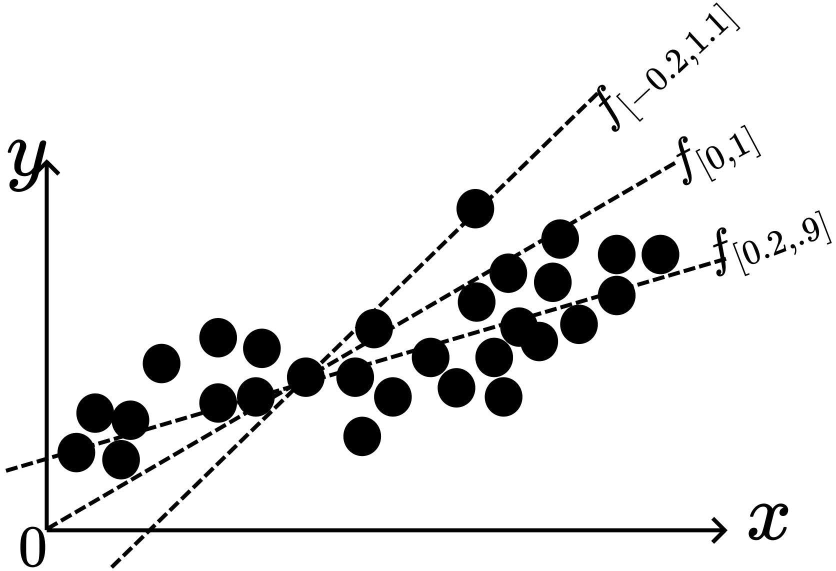
Linear regression

Linear regression is arguably the **most important** machine learning method

What is the best fit given a set of data points?

No need to guess or to use numerical optimization:

There is an exact analytical solution to this question



Learning objectives

- linear model
- evaluation criteria
- how to find the best fit
- geometric interpretation
- maximum likelihood interpretation

x | input features → **ML algorithm** **with parameters θ** → y | output labels

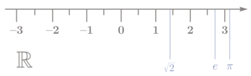
$f(x;\theta)$

Notation

each instance: $x \in \mathbb{R}^D$

$y \in \mathbb{R}$

$\mathbb{R}$ denotes set of real numbers



vectors are assume to be column vectors $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} = \begin{bmatrix} x_1, & x_2, & \ldots, & x_D \end{bmatrix}^\top$

a feature

example

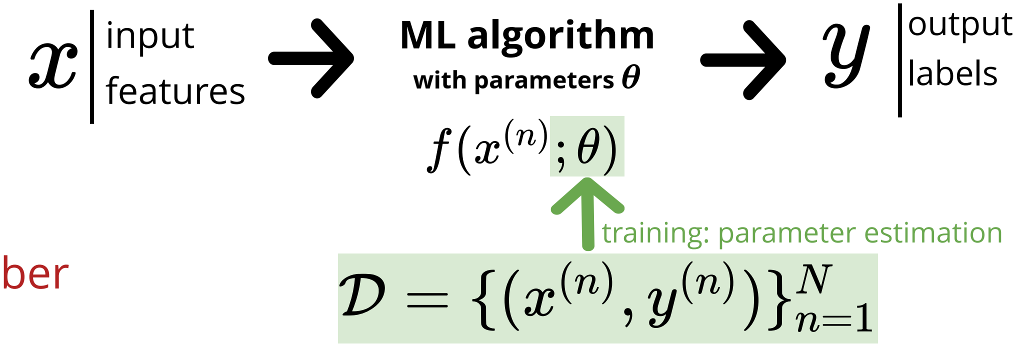
<tumorsize, texture, perimeter> = <18.2, 27.6, 117.5> → growth = +2

$$x = \begin{bmatrix} 18.2, & 27.6, & 117.5 \end{bmatrix}^\top$$

$$x = \begin{bmatrix} x_1, & x_2, & x_3 \end{bmatrix}^\top$$

$$y = 2$$

Notation



x | input features → **ML algorithm** **with parameters θ** → y | output labels

$f(x^{(n)};\theta)$

training: parameter estimation

$$\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$$

instance number

each instance:

$$x^{(n)} \in \mathbb{R}^D$$
$$y^{(n)} \in \mathbb{R}$$

we assume N instances in the dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$

each instance has D features indexed by d

for example, $x_d^{(n)} \in \mathbb{R}$ is the feature d of instance n

Notation

$$\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$$

design matrix: *concatenate all instances*

each row is a datapoint, each column is a feature

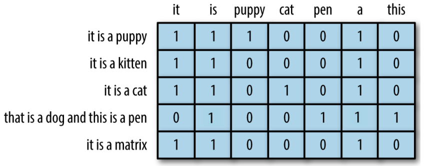
$$X = \begin{bmatrix} {x^{(1)}}^{\top} \\ {x^{(2)}}^{\top} \\ \vdots \\ {x^{(N)}}^{\top} \end{bmatrix} = \begin{bmatrix} x_1^{(1)}, & x_2^{(1)}, & \dots, & x_D^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)}, & x_2^{(N)}, & \dots, & x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times D}$$

one instance

one feature

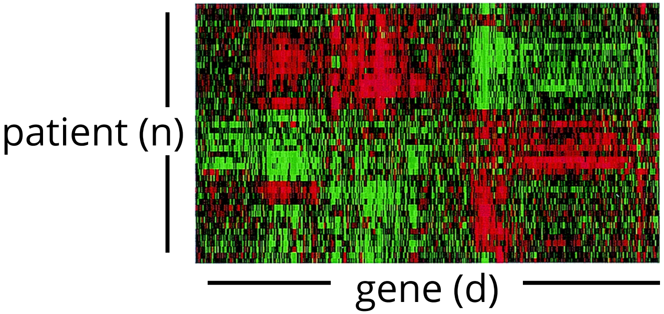
$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times 1}$$

Example:
instances: 5 sentences
features: 7 words



	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	1	0	0	1	1	1
it is a matrix	1	1	0	0	0	1	0

Example:
Micro array data (X), contains gene expression levels
labels (y) can be {cancer/no cancer classification} label for each patient, or how fast it is growing (regression)



$\in \mathbb{R}^{N \times D}$

Regression: examples

time-of-arrival-estimation.
input: route, weather, time of day
output: ETA

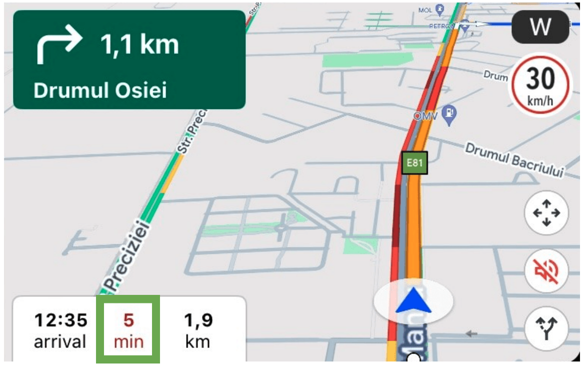
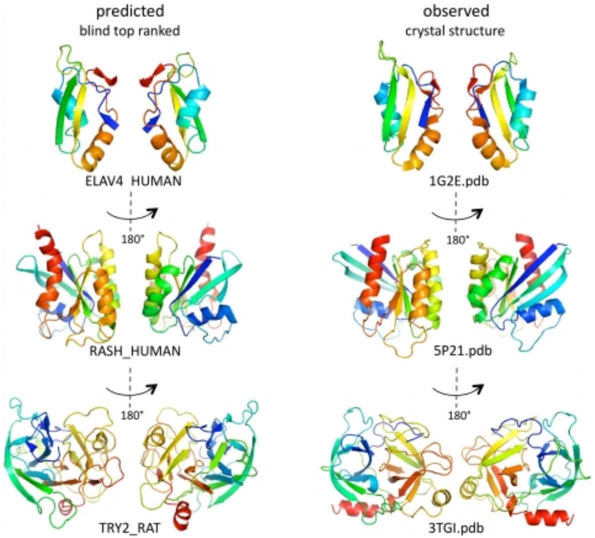


image from Google Maps

Protein folding.
input: sequences
output: 3D structure

Image from Marks et al. link



Origin of Regression

Method of least squares was invented by **Legendre** and **Gauss** (1800's)

Gauss used it to predict the future location of Ceres (largest asteroid in the asteroid belt)

ocean navigation

image from wiki history of navigation

Gauss
used it

Legendre
published it

Pearson
named it regression

find more here

Linear model of regression

x | input features → **ML algorithm** **with parameters w** $f(x; w)$ → y | output labels

$$f_w(x) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

model parameters or weights

$[w_0, w_1, \ldots w_D]$

bias or intercept

assuming a scalar output $f_w : \mathbb{R}^D \rightarrow \mathbb{R}$

will generalize to a vector later

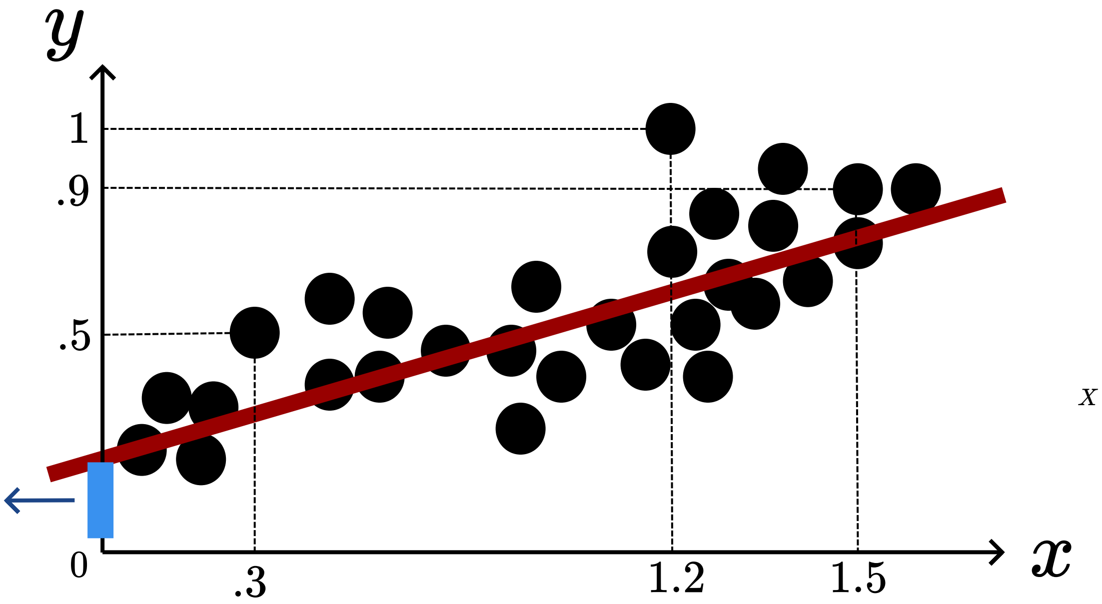
Linear model of regression: example $D = 1$

$$f_w(x) = w_0 + w_1 x_1$$

forms a line in 1 dimension

model parameters or weights

$$[w_0, w_1]$$



$f_w(0)$: bias or intercept

$$X = \begin{bmatrix} .3 \\ 1.2 \\ 1.5 \\ . \\ . \\ . \end{bmatrix}, \; Y = \begin{bmatrix} .5 \\ 1 \\ .9 \\ . \\ . \\ . \end{bmatrix}$$

Linear model of regression

$$f_w(x) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

model parameters or weights

bias or intercept

simplification

concatenate a 1 to x $\longrightarrow$ $x = [1, x_1, \ldots, x_D]^\top$

$$f_w(x) = w^\top x \qquad w = [w_0, w_1, \ldots, w_D]^\top$$

Linear regression: Objective

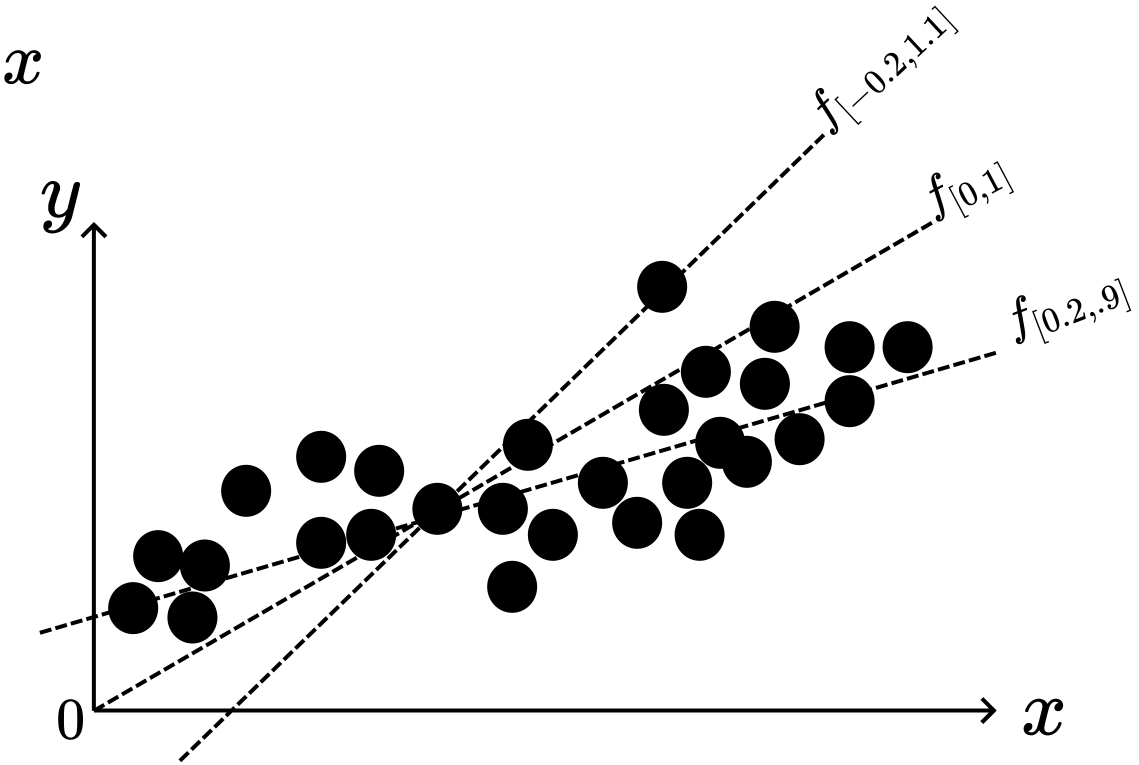
objective: find parameters to fit the data

model: $f_w(x) = w^\top x$

example $D = 1$

$$w = [w_0, w_1]$$

Which line is better?



Linear regression: Objective

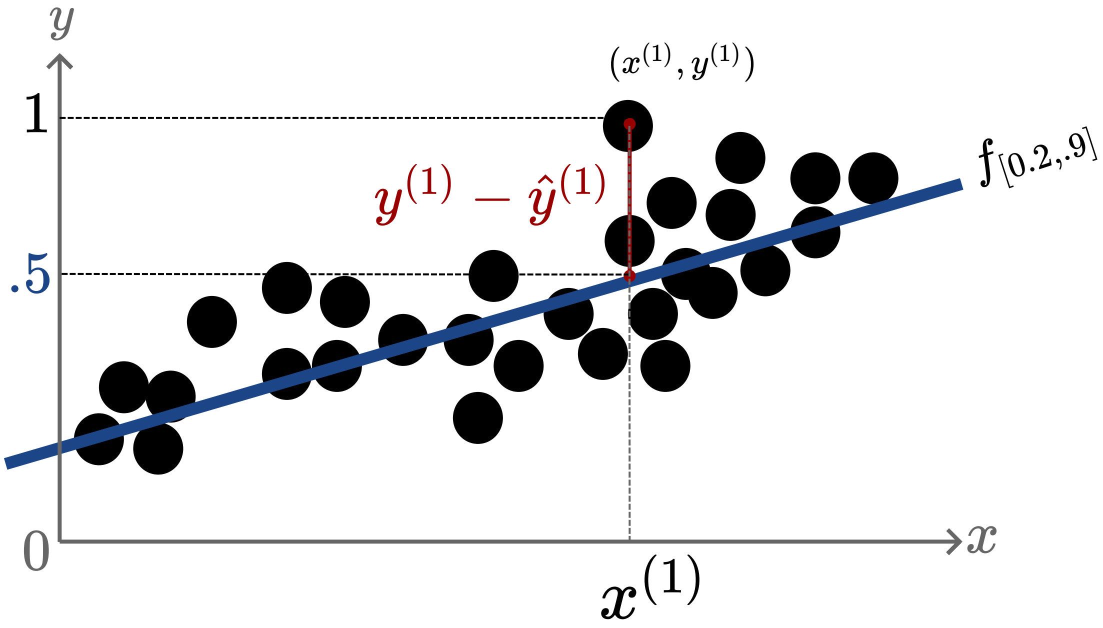
objective: find parameters to fit the data

true: $y^{(1)} = 1$

predicted: $\hat{y}^{(1)} = f(x^{(1)}) = .5$

residual: $y^{(1)} - f_w(x^{(1)})$

difference between
predicted (model output)
and true (observation)



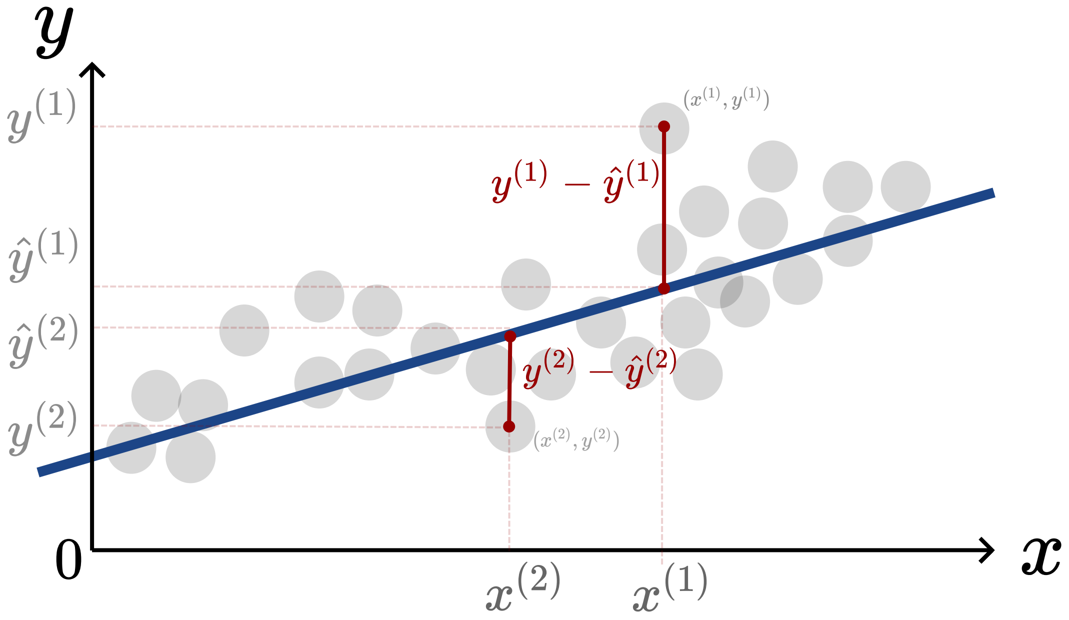
Linear regression: Objective

objective: find parameters to fit the data

how to consider all observations? sum all residuals?

square error **loss** (a.k.a. **L2** loss)

$$L(y, \hat{y}) \triangleq (y - \hat{y})^2$$



Linear regression: cost function

objective: find parameters to fit the data

we want $f_w(x^{(n)}) \approx y^{(n)} \quad x^{(n)}, y^{(n)} \quad \forall n$

minimize a measure of difference between $\hat{y}^{(n)} = f_w(x^{(n)})$ and $y^{(n)}$

square error **loss** (a.k.a. **L2** loss) $L(y, \hat{y}) \triangleq \frac{1}{2}(y - \hat{y})^2$

for a single instance (a function of labels)

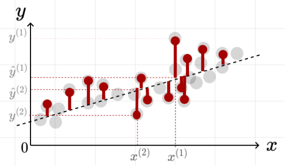
for future convenience

versus

for the whole dataset

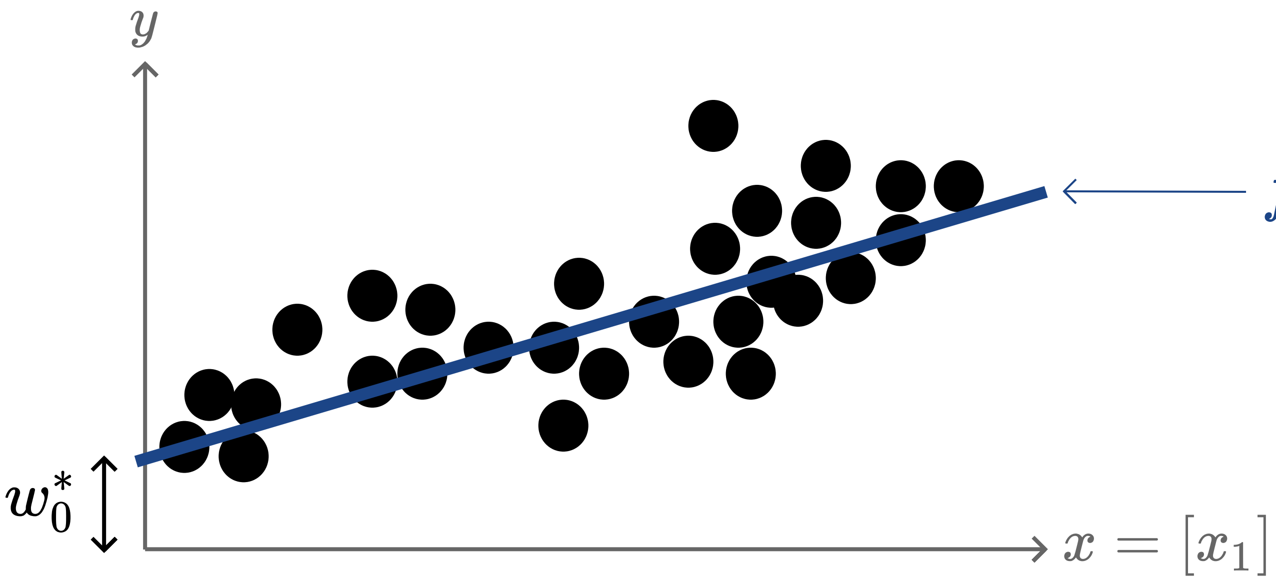
sum of squared errors **cost/loss function**

$$J(w) = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - w^\top x^{(n)}\right)^2$$



$$w^* = \arg\min_w J(w)$$

Example (D = 1) +bias (D=2)!

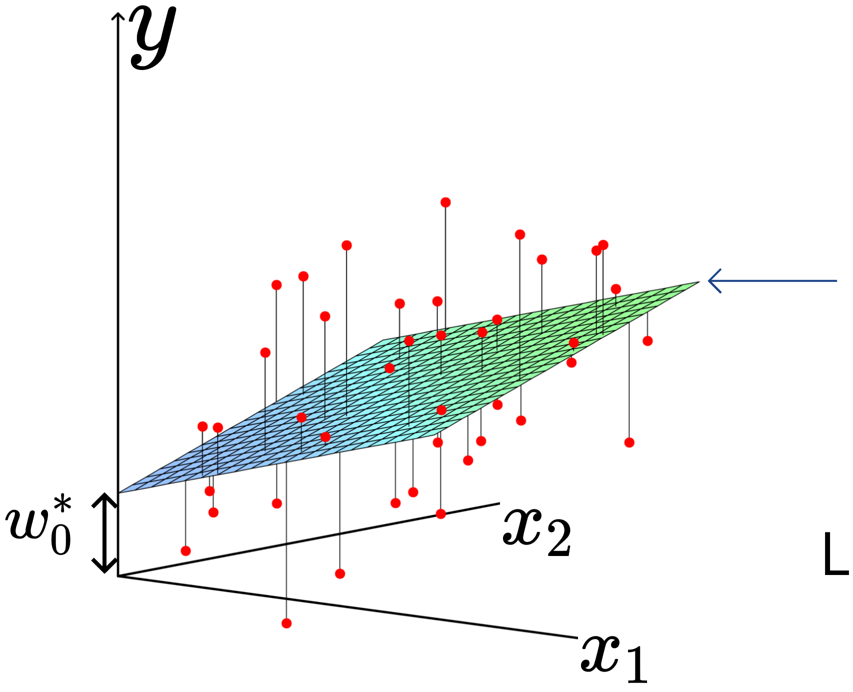


$$f_{w^*}(x) = w_0^* + w_1^* x$$

$$X = \begin{bmatrix} 1 & .3 \\ 1 & 1.2 \\ 1 & 1.5 \\ . & \\ . & \\ . & \end{bmatrix}, \; Y = \begin{bmatrix} .5 \\ .9 \\ 1 \\ . \\ . \\ . \end{bmatrix}$$

Linear Least Squares solution: $w^* = \arg\min_w \sum_n \frac{1}{2}\left(y^{(n)} - w^T x^{(n)}\right)^2$

Example (D=2) +bias (D=3)!



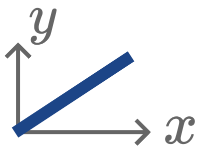
$$f_{w^*}(x) = w_0^* + w_1^* x_1 + w_2^* x_2$$

$$X = \begin{bmatrix} 1 & .3 & .5 \\ 1 & 1.2 & 1.6 \\ 1 & 1.5 & 1.2 \\ . & & \\ . & & \\ . & & \end{bmatrix}, \; Y = \begin{bmatrix} .5 \\ .9 \\ 1 \\ . \\ . \\ . \end{bmatrix}$$

Linear Least Squares

$$w^* = \arg\min_w \sum_n \left(y^{(n)} - w^T x^{(n)}\right)^2$$

Minimizing the cost



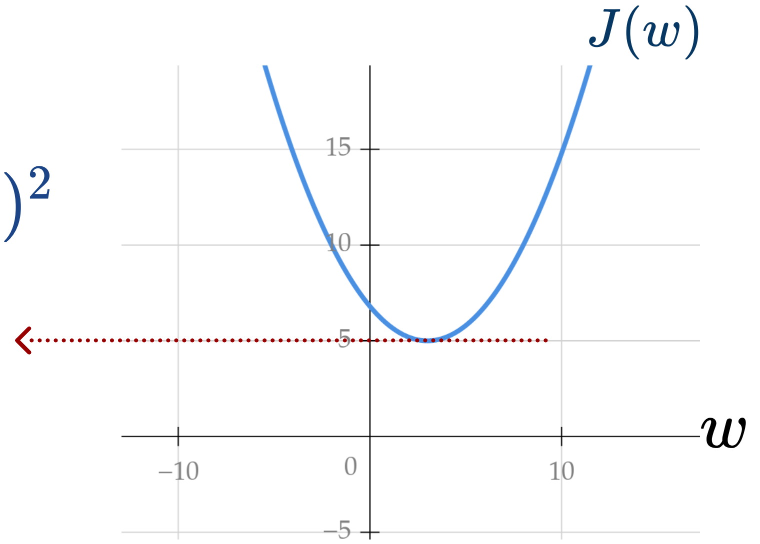
Simple case: D = 1 (no intercept)

model: $f_w(x) = wx$

both scalar

cost function $J(w) = \frac{1}{2} \sum_n (y^{(n)} - wx^{(n)})^2$

derivative $\frac{\mathrm{d}J}{\mathrm{d}w} = \sum_n x^{(n)} (wx^{(n)} - y^{(n)})$



setting the derivative to zero $w^* = \frac{\sum_n x^{(n)} y^{(n)}}{\sum_n {x^{(n)}}^2}$

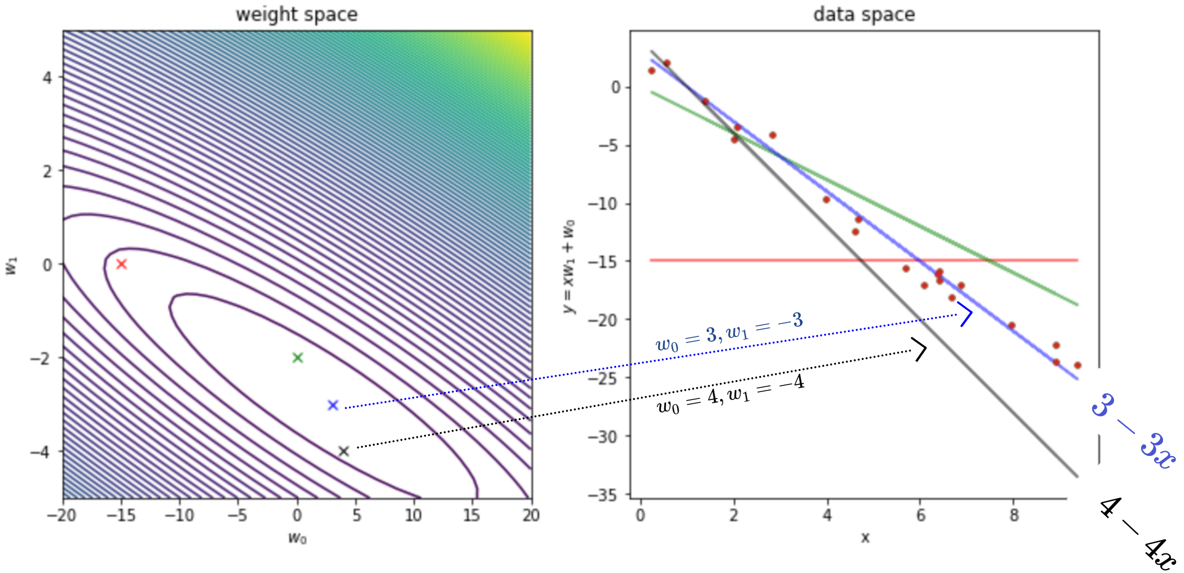
global minimum because the cost function is smooth and *convex*

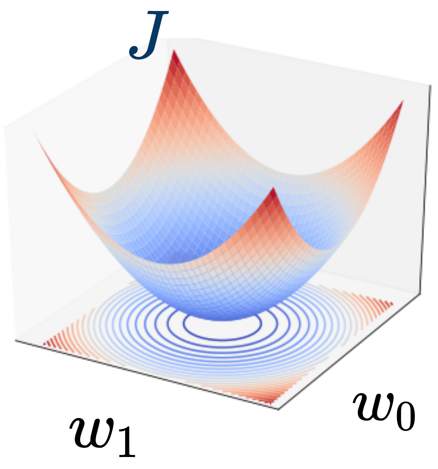
Minimizing the cost

D = 1 (with intercept)

model: $f_w(x) = w_0 + w_1 x$

cost: a multivariate function $J(w_0, w_1)$





the cost function is a smooth function of w

find minimum by setting partial derivatives to zero

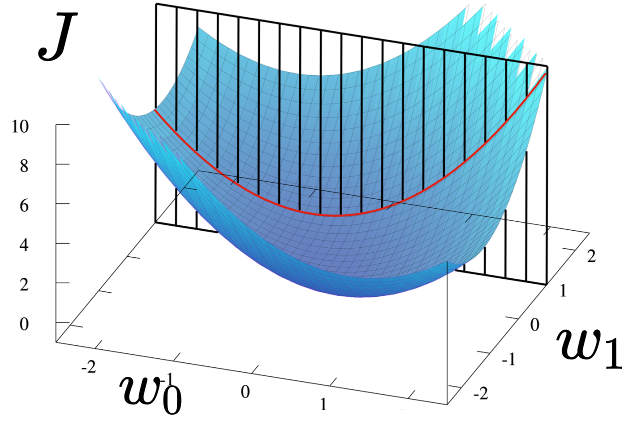
Minimizing the cost

for a multivariate function $J(w_0, w_1)$

partial derivatives instead of derivative

= derivative when other vars. are fixed

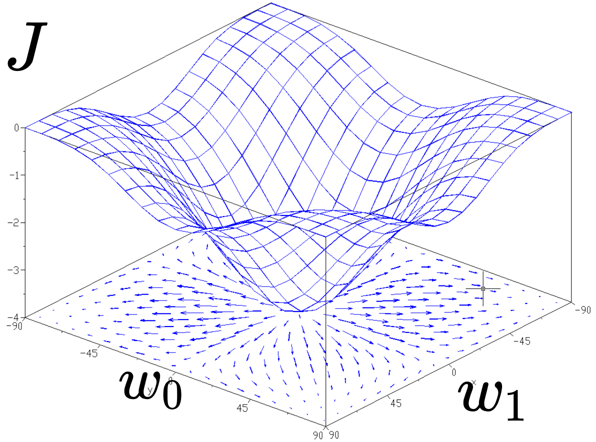
$$\frac{\partial}{\partial w_0} J(w_0, w_1) \triangleq \lim_{\epsilon \to 0} \frac{J(w_0+\epsilon, w_1) - J(w_0, w_1)}{\epsilon}$$



critical point: all partial derivatives are zero

gradient: vector of all partial derivatives

$$\nabla J(w) = [\frac{\partial}{\partial w_1} J(w), \cdots \frac{\partial}{\partial w_D} J(w)]^\top$$



Minimizing the cost

for general case (any D)

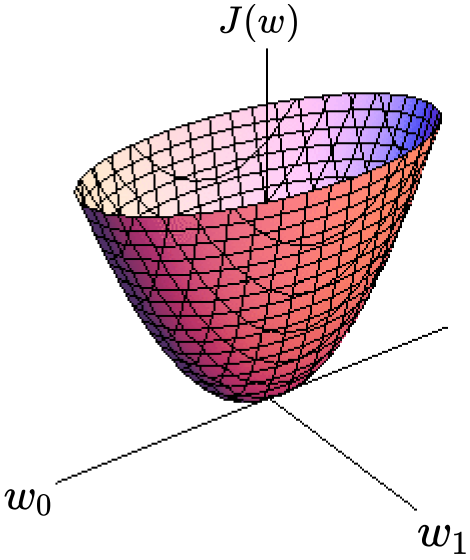
find the critical point by setting $\frac{\partial}{\partial w_d} J(w) = 0$

$$\frac{\partial}{\partial w_d} \sum_n \frac{1}{2}(y^{(n)} - f_w(x^{(n)}))^2 = 0$$

using **chain rule**: $\frac{\partial J}{\partial w_d} = \frac{\mathrm{d}J}{\mathrm{d}f_w} \frac{\partial f_w}{\partial w_d}$

we get $\sum_n (w^\top x^{(n)} - y^{(n)}) x_d^{(n)} = 0 \quad \forall d \in \{1, \ldots, D\}$

D equations with *D* unknowns



cost is a smooth and convex function of w

we are ignoring the bias term here, with the bias term, it would be D+1 equations and D+1 unknown for d in [0,D]

Linear regression: Matrix form

instead of $\underset{\in \mathbb{R}}{\hat{y}^{(n)}} = \underset{1 \times D}{w^{\top}} \underset{D \times 1}{x^{(n)}}$

Note: *D* is in fact dimensions of the input +1 due to the simplification and adding the bias/intercept term

use **design matrix** to write $\underset{N \times 1}{\hat{y}} = \underset{N \times D}{X} \underset{D \times 1}{w}$

$\hat{y}^{(1)} = w_0 + x_1^{(1)} w_1 + x_2^{(1)} w_2 + \cdots + x_D^{(1)} w_D$

$$\hat{Y} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)}, & x_2^{(1)}, & \cdots, & x_D^{(1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)}, & x_2^{(N)}, & \cdots, & x_D^{(N)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

Linear least squares

$$\arg\min_w \frac{1}{2}||y - Xw||_2^2 = \frac{1}{2}(y - Xw)^{\top}(y - Xw)$$

squared L2 norm of the **residual** vector

Minimizing the cost: Matrix form

Linear least squares

$$J(w) = \frac{1}{2}||y - Xw||^2 = \frac{1}{2}(y - Xw)^T(y - Xw)$$

$y^T Xw = w^T X^T y$

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}[y^T y + w^T X^T Xw - 2y^T Xw]$$

$$\frac{\partial Xw}{\partial w} = X^T$$

Using matrix differentiation

$$\frac{\partial w^T Xw}{\partial w} = 2Xw$$

$$\frac{\partial J(w)}{\partial w} = 0 + 2X^T Xw - 2X^T y = 2X^T(Xw - y)$$

Closed form solution

$$\overset{D \times N}{X^\top} \overset{N \times 1}{(y - Xw)} = \vec{0}$$

matrix form (using the design matrix)

$$X^\top X w = X^\top y$$

system of D linear equations ($Aw = b$)

each row enforces one of D equations

similar to non-matrix form: optimal weights w* satisfy

$$\sum_n (y^{(n)} - w^\top x^{(n)}) x_d^{(n)} = 0 \quad \forall d$$

D equations with *D* unknowns

pseudo-inverse of X

$$w^* = (\underbrace{X^\top X}_{D \times D})^{-1} \underbrace{X^\top}_{D \times N} \underbrace{y}_{N \times 1}$$

closed form solution

Uniqueness of the solution

we can get a closed form solution! $w^* = (X^\top X)^{-1} X^\top y$

unless $D > N$

or when the $X^\top X$ matrix is not invertible

this matrix is not invertible when some of eigenvalues are zero!

that is, if features are completely correlated

... or more generally if features are not linearly independent

examples having a binary feature x_1 as well as its negation $x_2 = (1 - x_1)$

Time complexity

$$w^* = \overbrace{(X^\top X)}^{D \times D}{}^{-1} \overbrace{X^\top}^{D \times N} \overbrace{y}^{N \times 1}$$

$\mathcal{O}(ND)$ D elements, each using N ops.

$\mathcal{O}(D^3)$ matrix inversion

$\mathcal{O}(D^2 N)$ D x D elements, each requiring N multiplications

total complexity for is $\mathcal{O}(ND^2 + D^3)$ which becomes $\mathcal{O}(ND^2)$ for $N > D$

in practice we don't directly use matrix inversion (unstable)

however, other more stable solutions (e.g., Gaussian elimination) have similar complexity

Multiple targets

instead of $y \in \mathbb{R}^N$ we have $Y \in \mathbb{R}^{N \times D'}$

a different weight vectors for each target

each column of Y is associated with a column of W

$$\underbrace{\hat{Y}}_{N \times D'} = \underbrace{X}_{N \times D} \underbrace{W}_{D \times D'}$$

$$W^* = \underbrace{(X^\top X)}_{D \times D}{}^{-1} \underbrace{X^\top}_{D \times N} \underbrace{Y}_{N \times D'}$$

$$\hat{Y} = \begin{bmatrix} \hat{y}_1^{(1)} & \hat{y}_2^{(1)} \\ \hat{y}_1^{(2)} & \hat{y}_2^{(2)} \\ \vdots & \\ \hat{y}_1^{(N)} & \hat{y}_2^{(N)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)}, & x_2^{(1)}, & \cdots, & x_D^{(1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)}, & x_2^{(N)}, & \cdots, & x_D^{(N)} \end{bmatrix} \begin{bmatrix} w_{0,1} & w_{0,2} \\ w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ \vdots & \\ w_{D,1} & w_{D,2} \end{bmatrix}$$

$$\hat{y}_1^{(1)} = w_{0,1} + x_1^{(1)} w_{1,1} + x_2^{(1)} w_{2,1} + \cdots + x_D^{(1)} w_{D,1}$$

$$\hat{y}_2^{(1)} = w_{0,2} + x_1^{(1)} w_{1,2} + x_2^{(1)} w_{2,2} + \cdots + x_D^{(1)} w_{D,2}$$

Fitting non-linear data

so far we learned a linear function $f_w = \sum_d w_d x_d$

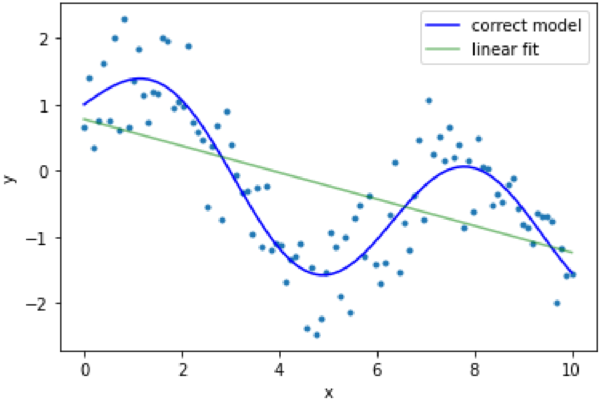
sometimes this may be too simplistic

example

Synthetic data when we generated data from a function $y^* = \sin(x) + cos(\sqrt{x})$

$$\mathcal{D} = \{(x^{(n)}, y^*(x^{(n)}) + \epsilon\}_{n=1}^{N}$$

small noise



we see linear fit is not close to correct model that the data is generated from, can we get a better fit?

idea create new more useful features out of initial set of given features

e.g., $x_1^2, x_1 x_2, \log(x),$ how about $x_1 + 2x_3$?

Nonlinear basis functions

so far we learned a linear function $f_w = \sum_d w_d x_d$

let's denote the set of all features by $\phi_d(x) \forall d$

the problem of linear regression doesn't change $f_w = \sum_d w_d \phi_d(x)$

$\phi_d(x)$ *is the new x*

solution simply becomes $(\Phi^\top \Phi) w^* = \Phi^\top y$

replacing X with Φ

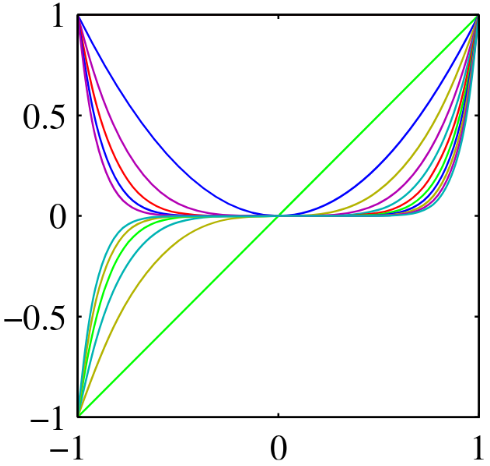
a (nonlinear) feature

$$\Phi = \begin{bmatrix} \phi_1(x^{(1)}), & \phi_2(x^{(1)}), & \cdots, & \phi_D(x^{(1)}) \\ \phi_1(x^{(2)}), & \phi_2(x^{(2)}), & \cdots, & \phi_D(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(N)}), & \phi_2(x^{(N)}), & \cdots, & \phi_D(x^{(N)}) \end{bmatrix}$$

one instance

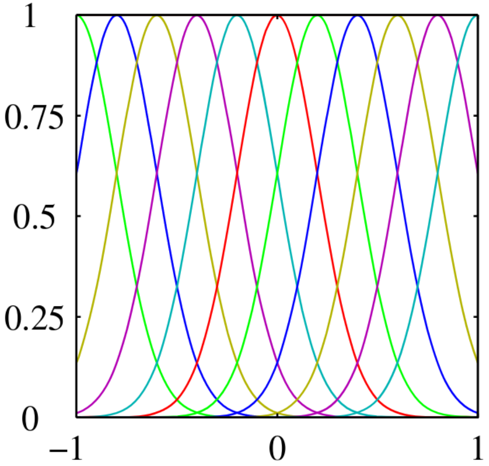
Nonlinear basis functions

example original input is scalar $x \in \mathbb{R}$



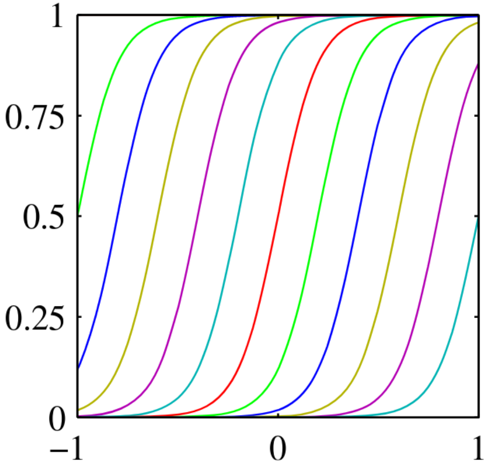
polynomial bases

$$\phi_k(x) = x^k$$



Gaussian bases

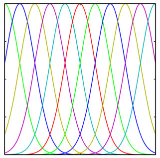
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



Sigmoid bases

$$\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$$

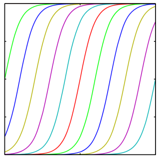
Linear regression with nonlinear bases: example



Gaussian bases

$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$

we are using a fixed standard deviation of s=1

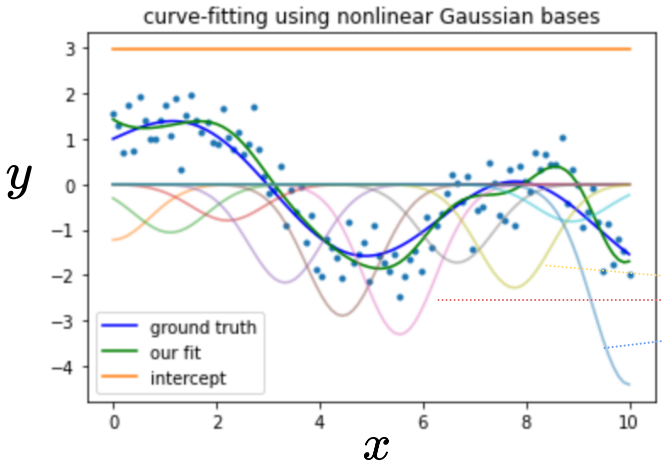


Sigmoid bases

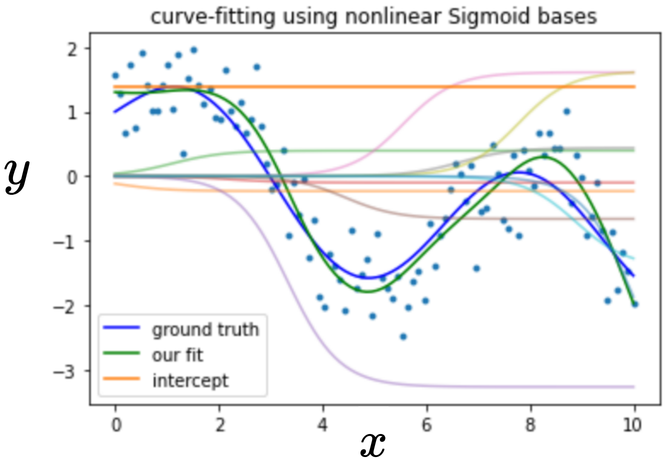
$$\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$$

we are using a fixed standard deviation of s=1

$$\hat{y}^{(n)} = w_0 + \sum_k w_k \phi_k(x)$$



the green curve (our fit) is the sum of these scaled Gaussian bases plus the intercept. Each basis is scaled by the corresponding weight



Probabilistic interpretation

idea given the dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$

learn a probabilistic model $p(y|x;w)$

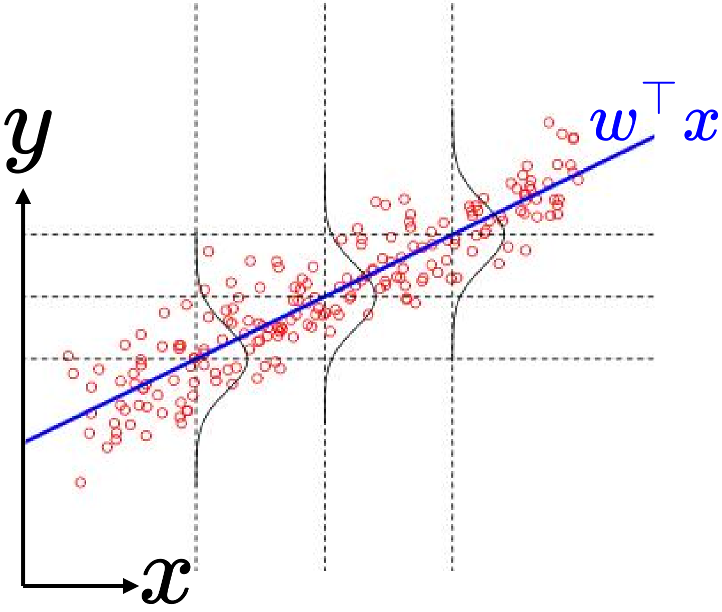


image from here

consider $p(y|x;w)$ with the following form

$$p_w(y \mid x) = \mathcal{N}(y \mid w^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^\top x)^2}{2\sigma^2}}$$

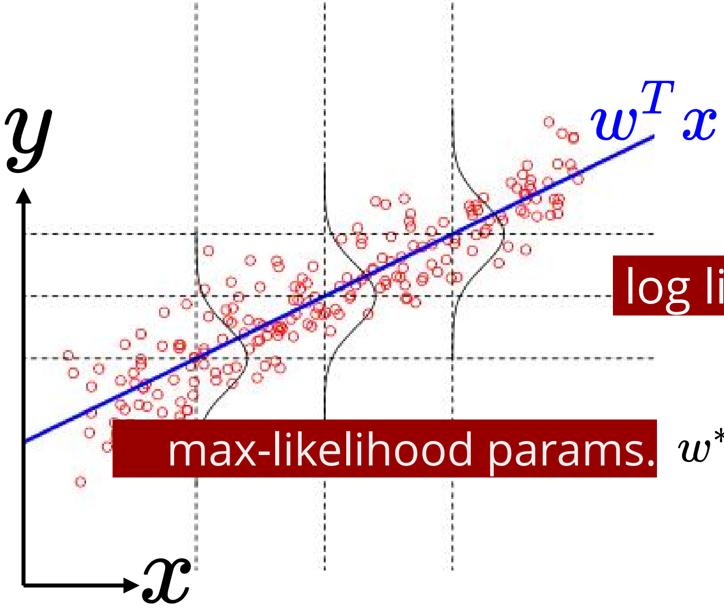
assume a fixed variance, say $\sigma^2 = 1$

Q: how to fit the model?

A: maximize the conditional likelihood!

Maximum likelihood & linear regression

cond. probability $p(y \mid x; w) = \mathcal{N}(y \mid w^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^\top x)^2}{2\sigma^2}}$



likelihood $L(w) = \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; w)$

log likelihood $\ell(w) = \sum_n -\frac{1}{2\sigma^2}(y^{(n)} - w^\top x^{(n)})^2 + \text{constants}$

max-likelihood params. $w^* = \arg\max_w \ell(w) = \arg\min_w \frac{1}{2}\sum_n (y^{(n)} - w^\top x^{(n)})^2$

linear least squares!

image from here

whenever we use square loss, we are assuming Gaussian noise!

Summary

linear regression:

- models targets as a linear function of features
- fit the model by minimizing the sum of squared errors
- has a direct solution with $\mathcal{O}(ND^2 + D^3)$ complexity
- probabilistic interpretation

we can build more expressive models:

- using any number of non-linear features

Looking forward

linear regression has some clear advantages

- computationally simple and efficient
- easy to interpret: it aligns well with our intuitions of simple correlations

but it is also fundamentally limited

- real life is rarely linear
- using non-linear features can solve this issue,
    - but how to choose the features?
    - using too many of them leads to overfitting