

Data Mining



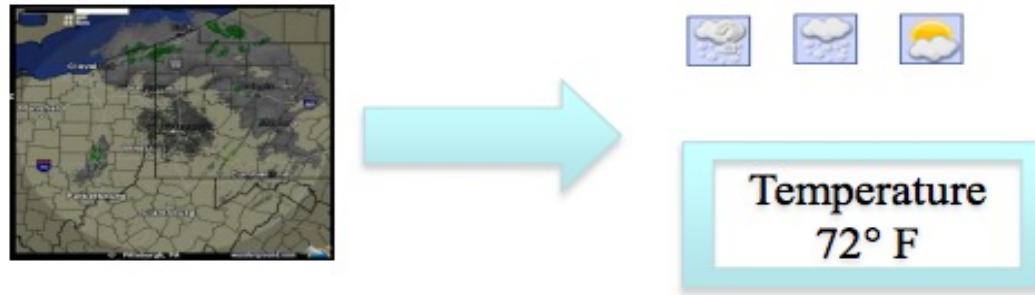
Regression Analysis Linear Regression (1)

Regression

Stock market



Weather prediction



Predict the temperature at any given location

Algorithms:

- Linear Regression
- Random Forest
- XGBoost

Regression Example: Manufacturing Yield Prediction

- Predict final wafer yield (%) from manufacturing process parameters.
- Output is continuous → REGRESSION.
- Input Features (X)
 - Furnace temperature (°C)
 - Chamber pressure (Pa)
 - Deposition time (seconds)
 - Gas flow rate (sccm)
 - Defect density (defects/cm²)
 - Tool age (days)
- Target Variable (Y)
 - Yield Percentage, eg. 93.2%, 88.7%, 95.1%,

Regression Example: Manufacturing Yield Prediction

- Example Data:

Temperature (°C)	Pressure (Pa)	Time (s)	Flow (sccm)	Defects (/cm²)	Tool Age (days)	Yield (%)
450	110	38	42	0.8	120	93.2
460	108	41	45	1.5	200	88.7
455	112	40	44	0.6	90	95.1
470	105	43	47	2.1	240	82.3

X Y

- $\text{yield} = f(\text{temperature}, \text{pressure}, \text{time}, \text{flow}, \text{defects}, \text{tool_age})$
- Pipeline:
 - Sensors → Features → ML Model → Predicted Yield
 - X → Model → \hat{Y}

Regression Example: Manufacturing Yield Prediction

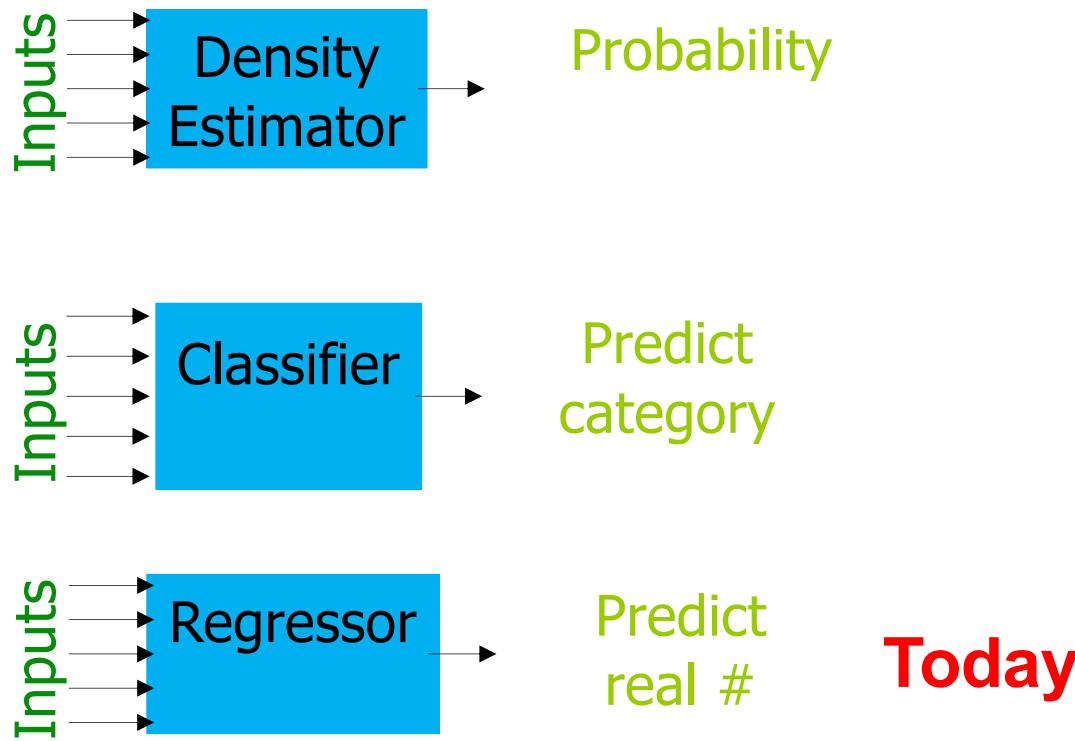
- Why is this important?
 - Used in:
 - Semiconductor fabs
 - Chemical plants
 - Pharma production
- Models learned can be used to support quality control and anomaly detection.
- Jupyter Lab Demo

Other Regression Examples

- Marketing Campaigns
 - How do different marketing campaigns affect the sale, customer satisfaction?
 - Variables including: campaign type, budget, duration, channel, and target audience
- Health Outcomes
 - How are medical condition or therapy outcome affected by factors/variables such as age, gender, weight, diet, exercise, and medication?
- Many Applications
 - Risk analysis, user emotion prediction, credit score prediction, sports player performance prediction, employee job satisfaction prediction,
...

Regressor vs Classification

- Regression vs Classification

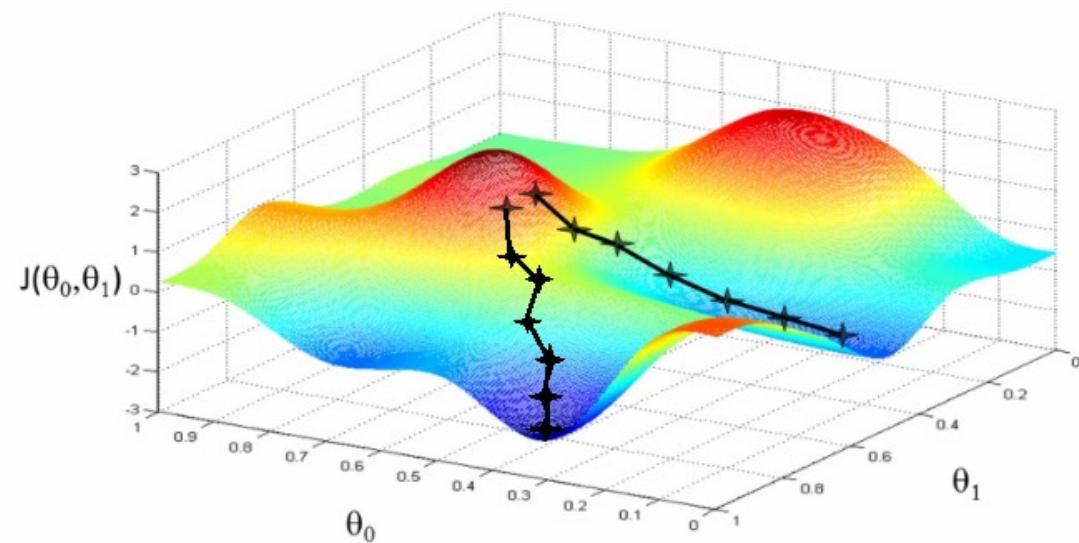


- Regression –a discriminative learning method
 - As optimization → Gradient descent

Regression

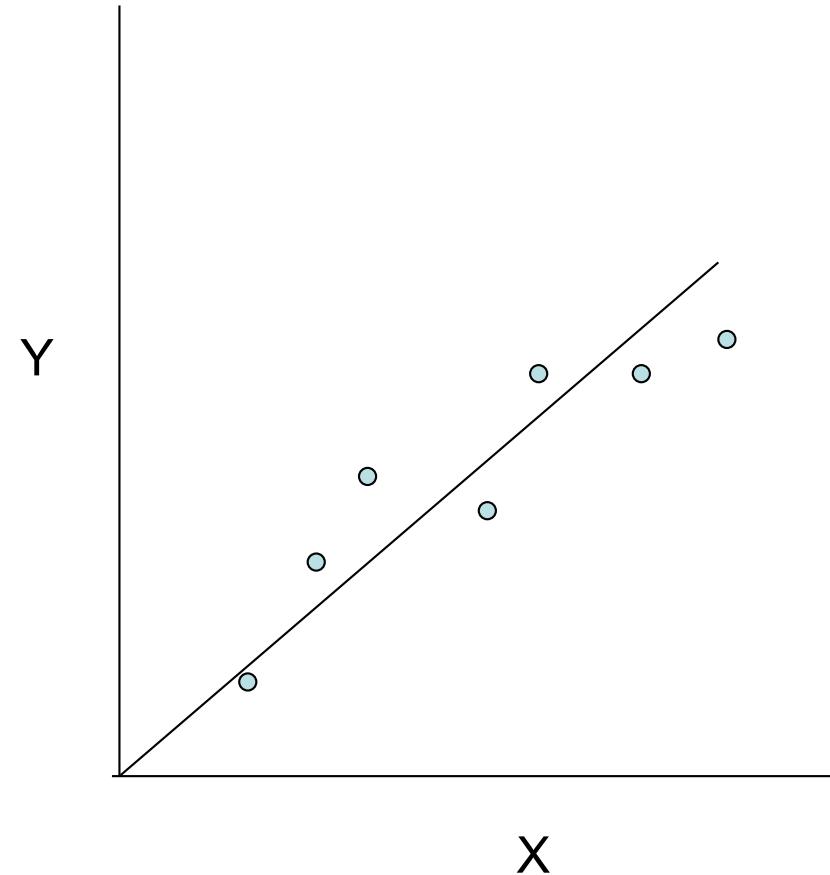
Least Mean
Squares

- Regression for LMS as optimization



Linear regression

- Given an input x we would like to compute an output y
- For example:
 - Predict yield from furnace temperature
 - Predict height from age
 - Predict distance from wall from sensor readings



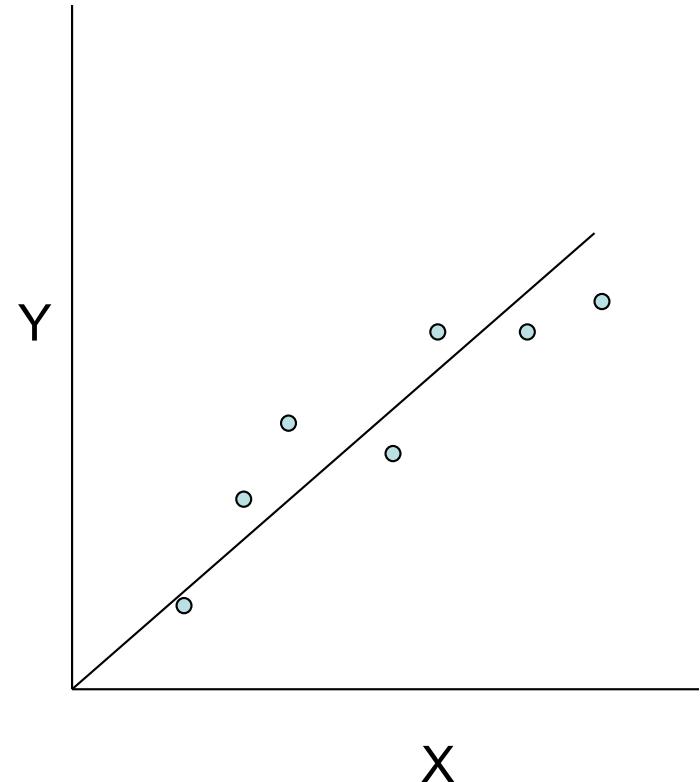
Linear regression

- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are
trying to predict Observed values

$$y = wx + \varepsilon$$

where w is a parameter and
 ε represents measurement
or other noise

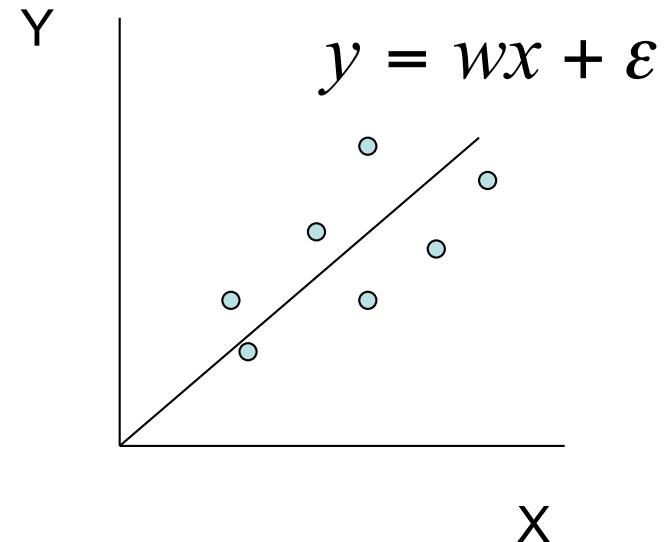


Linear regression

- Our goal is to estimate w from a training data of $\langle x_i, y_i \rangle$ pairs
- Optimization goal: minimize squared error (least squares):

$$\arg \min_w \sum_i (y_i - wx_i)^2$$

- Why least squares?
 - minimizes squared distance between measurements and predicted line



Solving linear regression

- To optimize:

We just take the derivative w.r.t. to w

$$\frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = 2 \sum_i -x_i(y_i - wx_i)$$

prediction

Solving linear regression

- To optimize – closed form:
- We just take the derivative w.r.t. to w and set to 0:

$$\frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = 2 \sum_i -x_i(y_i - wx_i) \Rightarrow$$

$$2 \sum_i x_i(y_i - wx_i) = 0 \Rightarrow 2 \sum_i x_i y_i - 2 \sum_i w x_i x_i = 0$$

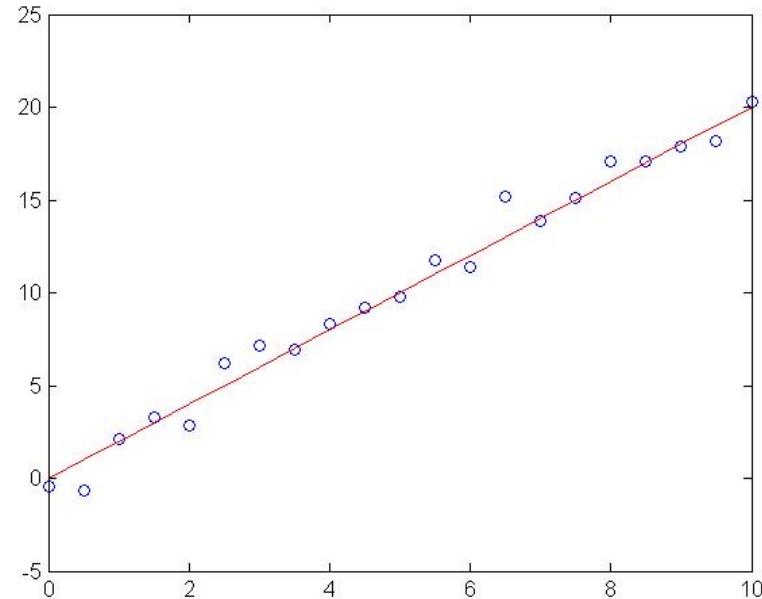
$$\sum_i x_i y_i = \sum_i w x_i^2 \Rightarrow$$

$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

cov(X,Y)/var(X)
if mean(X)=mean(Y)=0

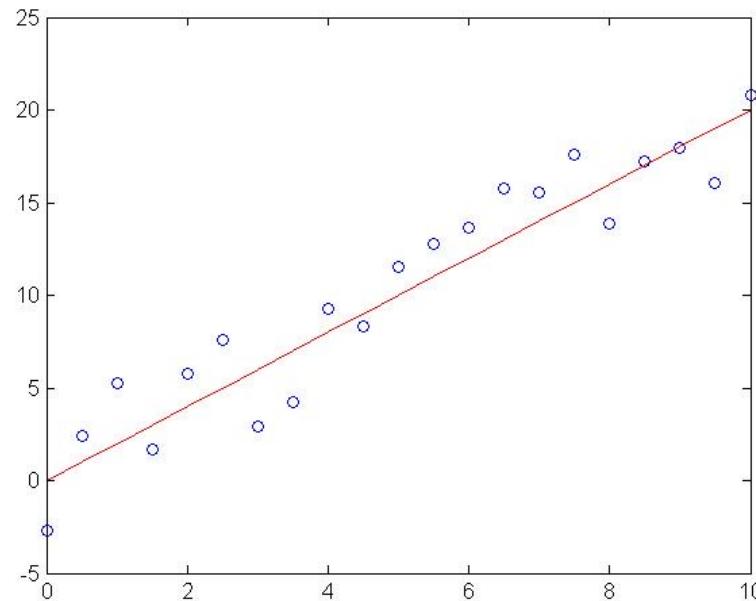
Regression example

- Generated: $w=2$
- Recovered: $w=2.03$
- Noise: $\text{std}=1$



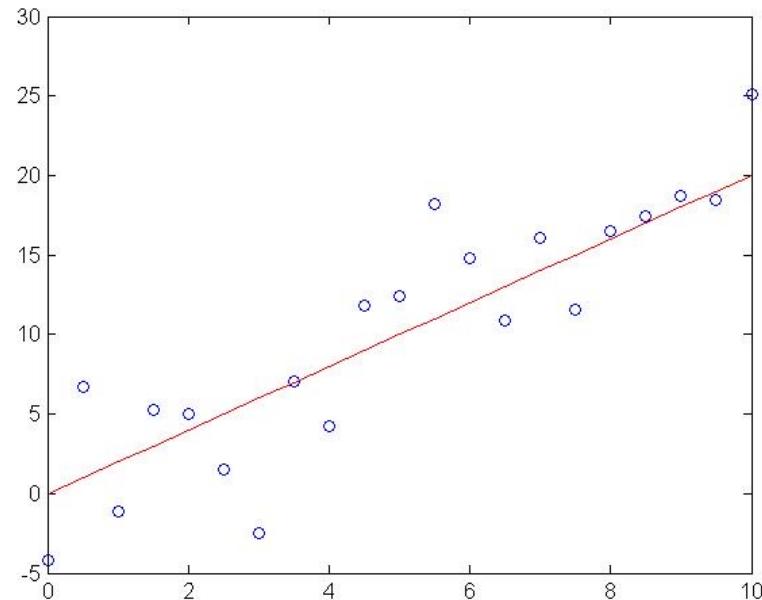
Regression example

- Generated: $w=2$
- Recovered: $w=2.05$
- Noise: $\text{std}=2$



Regression example

- Generated: $w=2$
- Recovered: $w=2.08$
- Noise: $\text{std}=4$

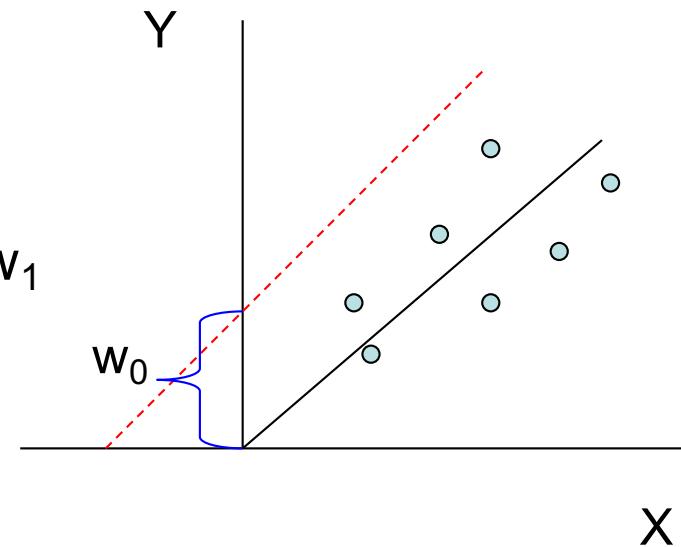


Bias term

- So far, we assumed that the line passes through the origin
- What if the line does not?
- No problem, simply change the model to

$$y = w_0 + w_1 x + \varepsilon$$

- Can use least squares to determine w_0 , w_1

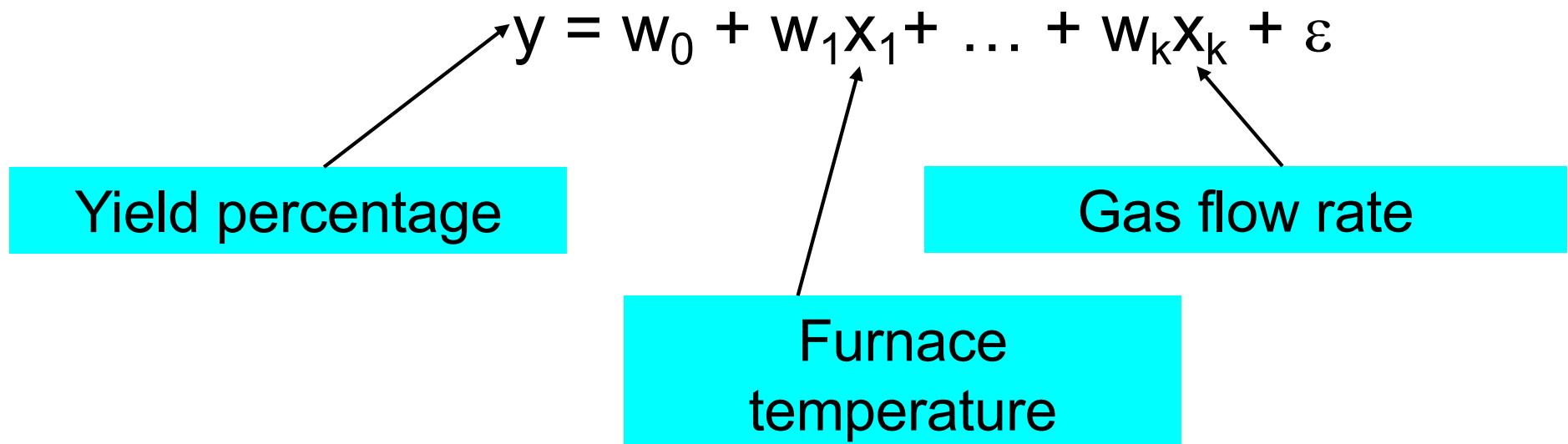


$$w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n}$$

Multivariate regression

- What if we have several inputs?
 - Furnace temperature (°C), Chamber pressure (Pa), Deposition time (seconds), Gas flow rate (sccm), Defect density (defects/cm²), Tool age (days)
- This becomes a multivariate regression problem
- Again, its easy to model:



Other functions of x

Not all functions can be approximated by a line/hyperplane...

$$y = 10 + 3x_1^2 - 2x_2^2 + \varepsilon$$

In some cases, we would like to use polynomial or other terms based on the input data, are these still linear regression problems?

Yes. As long as the *coefficients* are linear the equation is still a linear regression problem!

Non-Linear basis function

- So far, we only used the observed values x_1, x_2, \dots
- However, linear regression can be applied in the same way to **functions** of these values
 - Eg: to add a term $w x_1 x_2$ add a new variable $z=x_1 x_2$ so each example becomes: x_1, x_2, \dots, z
- As long as these functions can be directly computed from the observed values, the parameters are still linear in the data, and the problem remains a multi-variate linear regression problem

$$y = w_0 + w_1 x_1^2 + \dots + w_k x_k^2 + \varepsilon$$

Non-Linear basis function

- How can we use this to add an intercept term?

Add a new “variable” $z=1$ and weight w_0

Non-linear basis functions

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for $j=0 \dots n$

- Gaussian:

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

- Sigmoid:

$$\phi_j(x) = \frac{1}{1 + \exp(-s_j x)}$$

- Logs:

$$\phi_j(x) = \log(x + 1)$$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

General linear regression problem

- Using our new notations for the basis function linear regression can be written as

$$y = \sum_{j=0}^n w_j \phi_j(x)$$

- Where $\phi_j(\mathbf{x})$ can be either x_j for multivariate regression or one of the non-linear basis functions we defined
- ... and $\phi_0(\mathbf{x})=1$ for the intercept term

Data Mining



Learning/Optimizing
Multivariate Least Squares

Gradient Descent Approach

Gradient Descent for Linear Regression

Goal: minimize the following loss function:

$$J_{\mathbf{X}, \mathbf{y}}(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$

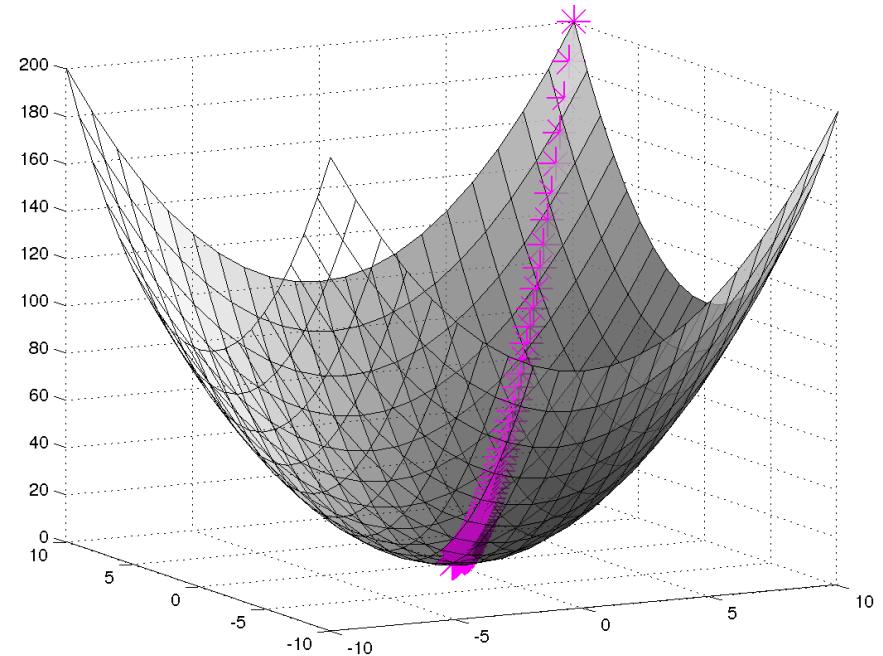
sum over n examples

sum over $k+1$ basis vectors

Gradient descent

The **gradient** is a fancy word for derivative, or the rate of change of a function.

- It's a vector (a direction to move) that points in the direction of greatest increase (or decrease) of a function
- Is zero at a local maximum or local minimum (because there is no single direction of increase)



Closed form solution (1)

$$J(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{\partial}{\partial w_j} \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right)^2$$

$$= 2 \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \frac{\partial}{\partial w_j} \left(y^i - \sum_k w_k \phi_k(x^i) \right)$$

$$= 2 \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \left(-\frac{\partial}{\partial w_j} \sum_k w_k \phi_k(x^i) \right)$$

$$= -2 \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \frac{\partial}{\partial w_j} \sum_k w_k \phi_k(x^i)$$

$$= -2 \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \phi_j(x^i)$$

Closed form solution (2)

$$-2 \sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \phi_j(x^i) = 0$$

$$\sum_i \left(y^i - \sum_k w_k \phi_k(x^i) \right) \phi_j(x^i) = 0$$

$$\sum_i y^i \phi_j(x^i) - \sum_i \sum_k w_k \phi_k(x^i) \phi_j(x^i) = 0$$

$$\sum_i \sum_k w_k \phi_k(x^i) \phi_j(x^i) = \sum_i y^i \phi_j(x^i)$$

$$\sum_k w_k \left(\sum_i \phi_k(x^i) \phi_j(x^i) \right) = \sum_i y^i \phi_j(x^i)$$

Closed form solution (3)

$$\sum_k w_k \left(\sum_i \phi_k(x^i) \phi_j(x^i) \right) = \sum_i y^i \phi_j(x^i)$$

$$\sum_k A_{jk} w_k = b_j$$

$$A_{jk} = \sum_i \phi_j(x^i) \phi_k(x^i)$$

$$b_j = \sum_i y^i \phi_j(x^i)$$

$$\sum_k A_{jk} w_k = b_j \quad \forall j$$

$$\mathbf{Aw} = \mathbf{b}$$

$$\mathbf{w} = \mathbf{A}^{-1} \mathbf{b}$$

$$w_j = \sum_k (\mathbf{A}^{-1})_{jk} b_k$$

$$w_j = \sum_k (\mathbf{A}^{-1})_{jk} \left(\sum_i y^i \phi_k(x^i) \right)$$

Rarely used in practice:

- Equivalent algebraic form:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- For data \mathbf{X} with n samples (row), d number of features (columns), the complexity is:

$$O(nd^2 + d^3)$$

when d is large, the d^3 term dominates.

Gradient Descent for Linear Regression

Goal: minimize the following loss function:

predict with: $\hat{y}^i = \sum_j w_j \phi_j(\mathbf{x}^i)$

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{\partial}{\partial w_j} \sum_i (y^i - \hat{y}^i)^2$$

$$= 2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} (y^i - \hat{y}^i)$$

$$= 2 \sum_i (y^i - \hat{y}^i) \left(-\frac{\partial}{\partial w_j} \hat{y}^i \right)$$

$$= -2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \hat{y}^i$$

$$= -2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \sum_k w_k \phi_k(x^i)$$

$$= -2 \sum_i (y^i - \hat{y}^i) \phi_j(x^i)$$

Gradient Descent for Linear Regression

Learning algorithm:

- Initialize weights $\mathbf{w} = \mathbf{0}$
- For $t=1, \dots$ until convergence:
 - Predict for each example \mathbf{x}^i using \mathbf{w} :
$$\hat{y}^i = \sum_{j=0}^k w_j \phi_j(\mathbf{x}^i)$$
 - Compute gradient of loss:
$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = -2 \sum_i (y^i - \hat{y}^i) \phi_j(x^i)$$
 - This is a vector \mathbf{g}
- Update: $\mathbf{w} = \mathbf{w} - \lambda^* \mathbf{g}$
- λ is the learning rate.

Gradient descent illustration ($\lambda=0.1$)

Model: $\hat{y} = wx$

Loss: $J(w) = \sum_i (y_i - wx_i)^2$

Gradient: $g(w) = \frac{dJ}{dw} = -2 \sum_i x_i(y_i - wx_i)$

Update: $w \leftarrow w - \lambda g(w)$

true w is 2

Data: $(x, y) = \{(1, 2), (2, 4), (3, 6)\}$

Choose: $w_0 = 0, \lambda = 0.1$

large λ , big oscillation

Iteration 1 (from $w_0 = 0$)

$$g(w_0) = -2[(1)(2 - 0) + (2)(4 - 0) + (3)(6 - 0)] = -2(2 + 8 + 18) = -56$$

$$w_1 = w_0 - \lambda g(w_0) = 0 - 0.1(-56) = 5.6$$

Iteration 2 (from $w_1 = 5.6$)

$$\text{Residuals: } y - w_1 x = [2 - 5.6, 4 - 11.2, 6 - 16.8] = [-3.6, -7.2, -10.8]$$

$$g(w_1) = -2[(1)(-3.6) + (2)(-7.2) + (3)(-10.8)] = -2(-3.6 - 14.4 - 32.4) = 100.8$$

$$w_2 = w_1 - \lambda g(w_1) = 5.6 - 0.1(100.8) = -4.48$$

Gradient descent illustration ($\lambda=0.1$)

Iteration 3 (from $w_2 = -4.48$)

Residuals:

$$y - w_2 x = [2 + 4.48, 4 + 8.96, 6 + 13.44] = [6.48, 12.96, 19.44]$$

Gradient:

$$\begin{aligned} g(w_2) &= -2[(1)(6.48) + (2)(12.96) + (3)(19.44)] \\ &= -2(6.48 + 25.92 + 58.32) = -181.44 \end{aligned}$$

Update:

$$w_3 = w_2 - 0.1(-181.44) = 13.664$$

Iteration 4 (from $w_3 = 13.664$)

Residuals:

$$y - w_3 x = [-11.664, -23.328, -34.992]$$

Gradient:

$$\begin{aligned} g(w_3) &= -2[(1)(-11.664) + (2)(-23.328) + (3)(-34.992)] \\ &= -2(-163.296) = 326.592 \end{aligned}$$

Update:

$$w_4 = 13.664 - 0.1(326.592) = -18.9952$$

Gradient descent illustration

($\lambda=0.01$)

Iteration 1

$$g(w_0) = -2(28) = -56$$

smaller λ

$$w_1 = 0 - 0.01(-56) = 0.56$$

Iteration 2

$$g(w_1) = -2(28 - 14 \cdot 0.56) = -40.32$$

$$w_2 = 0.56 + 0.4032 = 0.9632$$

Iteration 3

$$g(w_2) = -2(28 - 14 \cdot 0.9632) = -29.0304$$

$$w_3 = 0.9632 + 0.290304 = 1.2535$$

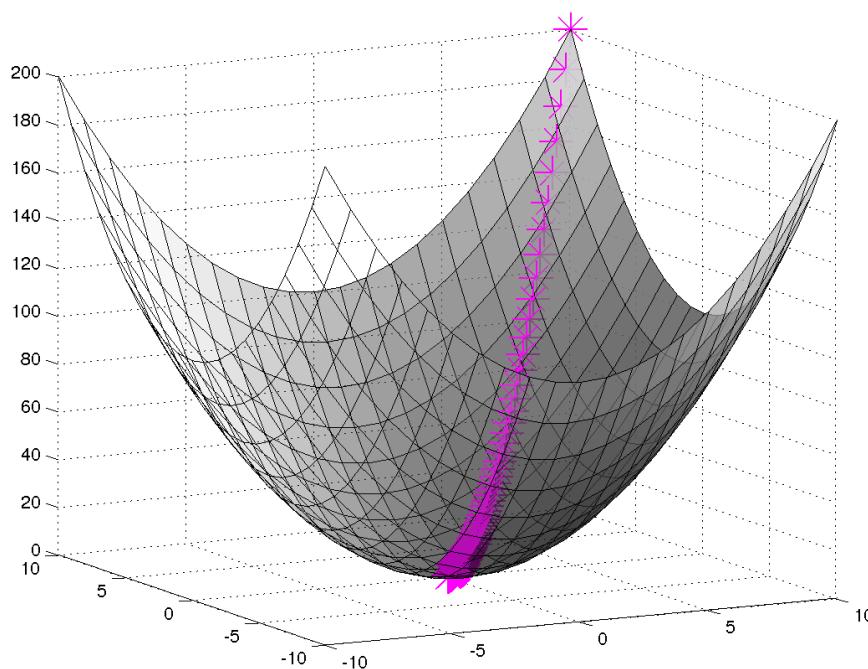
Iteration 4

$$g(w_3) = -2(28 - 14 \cdot 1.2535) = -20.9019$$

$$w_4 = 1.2535 + 0.2090 = 1.4625$$

Linear regression is a *convex* optimization problem

so again, gradient descent will reach a *global* optimum



First derivative

$$\frac{dL}{dw} = -2 \sum_i x_i(y_i - wx_i)$$

Differentiate again to get the second derivative:

$$\frac{d^2L}{dw^2} = 2 \sum_i x_i^2$$

Second derivative is **positive everywhere**:

- Only one minimum
- Gradient descent always finds the global optimum