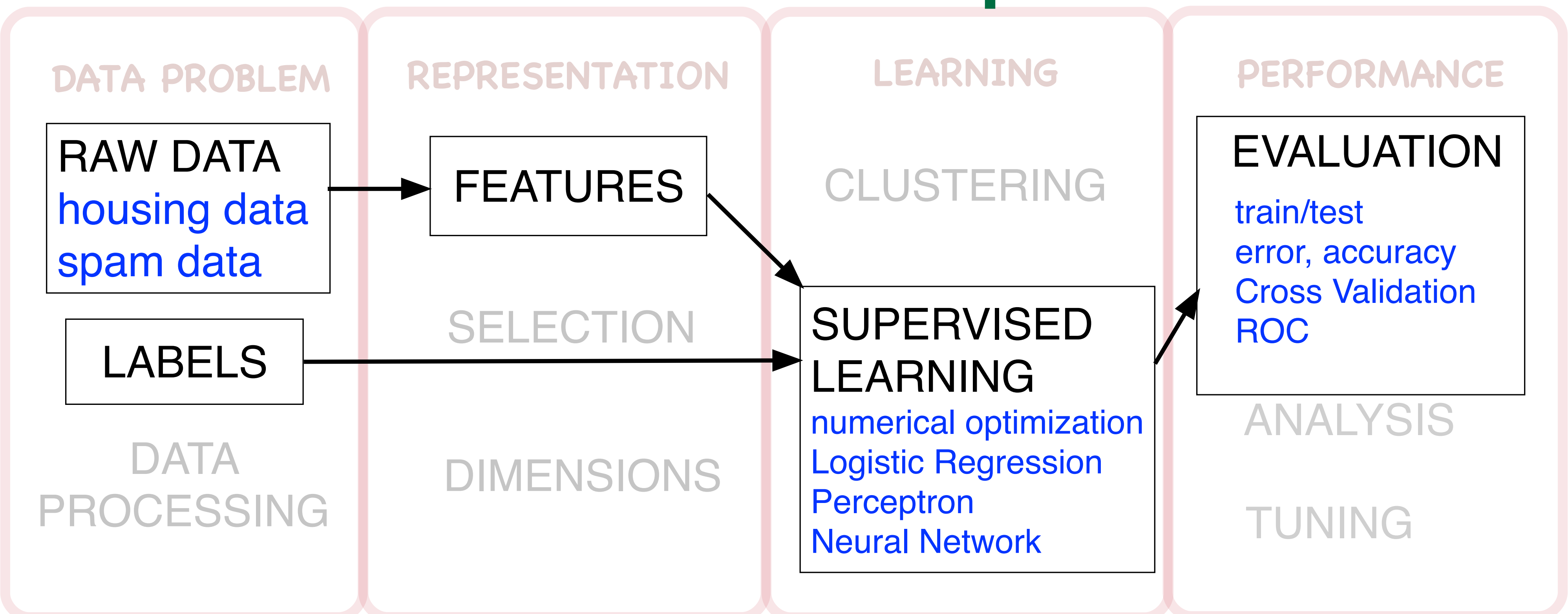
Gradient Descent Regression
Logistic Regression

Module2 : learning with Gradient Descent

module 2: numerical optimization



- formulate problem by model/parameters
- formulate error as mathematical objective
- optimize numerically the parameters for the given objective
- usually algebraic setup
  - involves matrices and calculus
- probabilistic setup (likelihoods) next module

Module 2 Objectives / Gradient Descent Regression

- numerical methods primer, gradient descent
- Regression using GD
  - learning rate
  - batch vs online modes
  - compare with normal eq regression (module 1)
- Logistic regression
- optional: Newton's optimization procedure

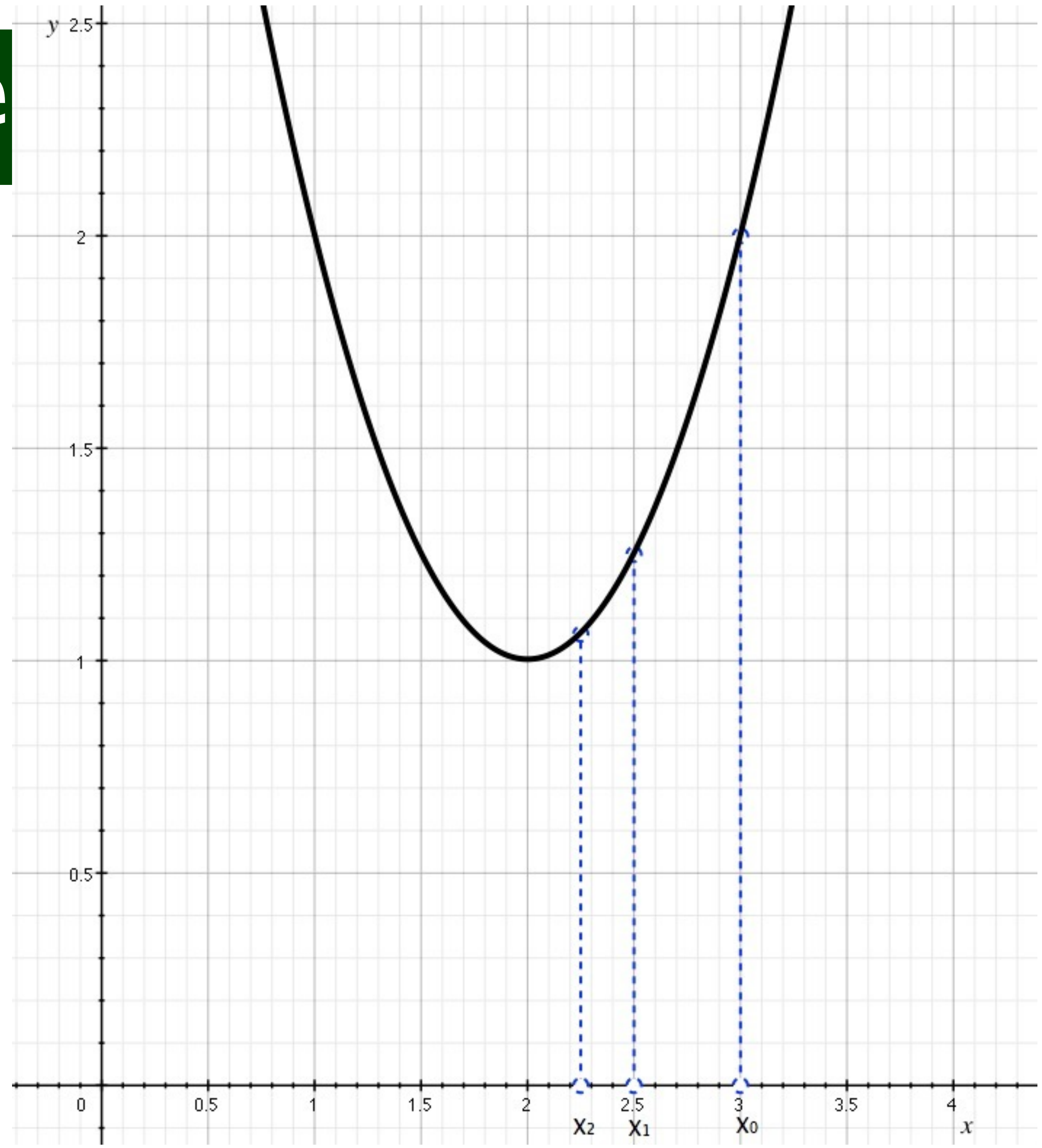
Gradient descent

- finds a local minima of the objective function (J) by guessing an initial set of parameters w and then ”walking” episodically in the opposite direction of the gradient $\partial J / \partial w$.

- update rule (per dimension)

$$w^j = w^j - \lambda \frac{\partial J(w)}{\partial w^j}$$

Gradient Descent Example



- $J(x) = (x - 2)^2 + 1$ and the initial guess for a minimum is $x_0 = 3$

- GD iteration 1
- GD iteration 2
- GD iteration 3

$$x_1 = x_0 - \lambda \frac{\partial J(x_0)}{\partial x} = 3 - .25(2 * 3 - 4) = 2.5$$

$$x_2 = x_1 - \lambda \frac{\partial J(x_1)}{\partial x} = 2.5 - .25(2 * 2.5 - 4) = 2.25$$

$$x_3 = x_2 - \lambda \frac{\partial J(x_2)}{\partial x} = 2.25 - .25(2 * 2.25 - 4) = 2.125$$

regression goal

- housing data, two features (toy example)

Living area (ft^2)	#bedrooms	price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
...	...	...

- regressor = a linear predictor

$$h_\theta(\mathbf{x}) = \theta^0 + \theta^1 x^1 + \theta^2 x^2$$

$$h(\mathbf{x}) = \sum_{d=0}^{D} \theta^d x^d$$

- such that h(x) approximates label(x)=y as close as possible, measured by square error

$$J(\theta) = \sum_t (h_\theta(\mathbf{x}_t) - y_t)^2$$

Regression Normal Equations (module1)

- Linear regression has a well known exact solution, given by linear algebra

- X= training matrix of feature values
- Y= corresponding labels vector

- then regression coefficients that minimize objective J are

$$\theta = (X^T X)^{-1} X^T Y$$

Problems with exact solution for regression

objective

$$J(\theta) = \sum_t (h_\theta(\mathbf{x}_t) - y_t)^2$$

solution

$$\theta = (X^T X)^{-1} X^T Y$$

- very unstable
- impractical for large matrices
- slow
- undesirable in cases with many outliers

Gradient Descent for linear regression

- differentiate the objective $J(w) = \frac{1}{2}\sum_t (h_w(\mathbf{x}_t) - y_t)^2$

$$
\begin{aligned}
\frac{\partial J(w)}{\partial w^j} &= \frac{\partial \frac{1}{2}(h_w(\mathbf{x}) - y)^2}{\partial w^j} \\
&= (h_w(\mathbf{x}) - y)\frac{\partial (h_w(\mathbf{x}) - y)}{\partial w^j} \\
&= (h_w(\mathbf{x}) - y)\frac{\partial (\sum_j w^j x^j - y)}{\partial w^j} \\
&= (h_w(\mathbf{x}) - y)x^j
\end{aligned}
$$

- GD update rule for one datapoint

$$w^j = w^j - \lambda(h_w(\mathbf{x}) - y)x^j$$

- GD for all datapoints (batch)

$$w^j = w^j - \lambda \sum_t (h_w(\mathbf{x}_t) - y_t)x_t^j$$

GD for linear regression

- batch (all datapoints) update step

$$w^j = w^j - \lambda \sum_t (h_w(\mathbf{x}_t) - y_t) x_t^j$$

- alternative stochastic (online) update
  - i or t indicate the datapoint
  - j indicates the feature (column in data)

LOOP for t=1 to m

$$w^j = w^j - \lambda (h_w(\mathbf{x}_t) - y_t) x_t^j \text{ for all } j$$

END LOOP

least mean square objective convexity

- GD "walks" the function argument towards a local minimum
  - it is possible (and sometimes likely) to obtain a local minimum that is not the GLOBAL minimum

- however this doesn't happen for regression objective, since it is convex
  - verify convexity by looking at the second derivative matrix
  - Hessian matrix of J(w) is positive semidefinite which implies J convex

$$J(w) = \frac{1}{2}\sum_t (h_w(\mathbf{x}_t) - y_t)^2$$

$$\frac{\partial(J(w))}{\partial w^i} = \sum_t (\sum_d w^d x_t^d - y_t) x_t^i$$

$$\frac{\partial^2 J(w)}{\partial w^i \partial w^j} = \sum_t x_t^i x_t^j$$

Hessian matrix of $J(w)$ is $X^T X$

$$\forall w, w^T X^T X w = (Xw)^T (Xw) \geq 0.$$

Logistic regression for classification

- Logistic transformation

$$g(z) = \frac{1}{1 + e^{-z}}$$

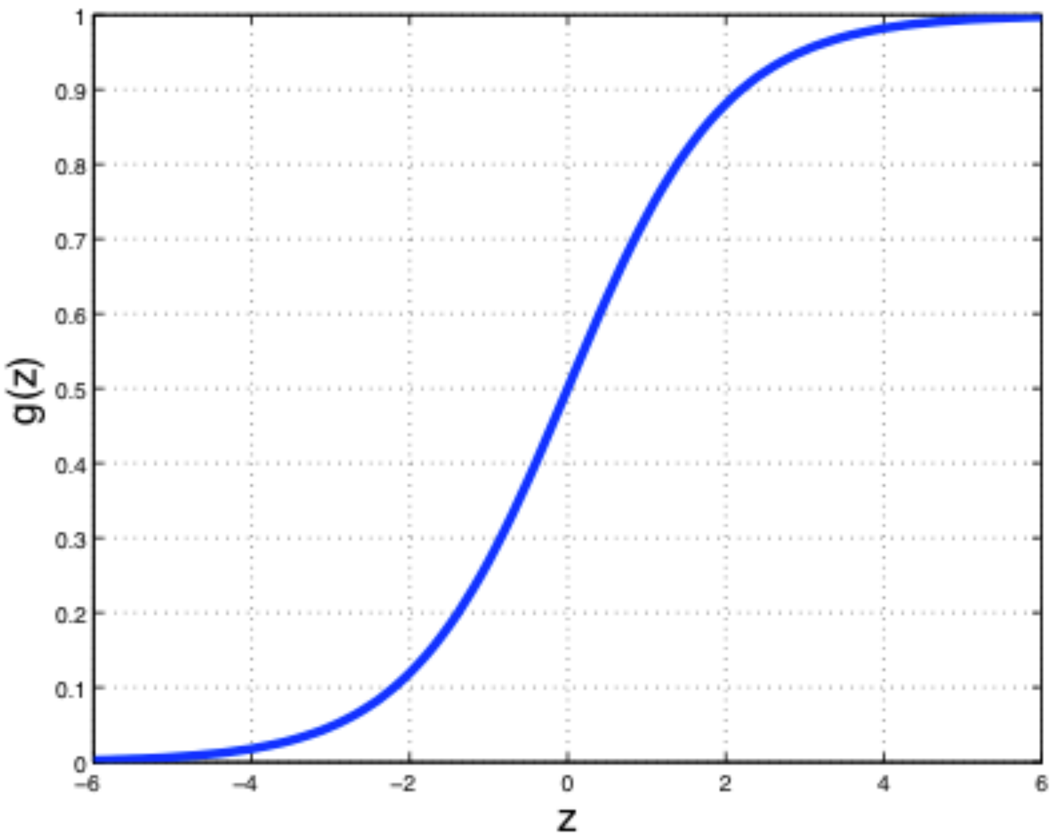


Figure 1: Logistic function

- Logistic differential

$$\begin{aligned} g'(z) &= \frac{\partial g(z)}{\partial z} \\ &= \frac{1}{(1 + e^{-z})^2} e^{-z} \\ &= \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}}\right) \\ &= g(z)(1 - g(z)) \end{aligned}$$

Logistic regression

- Logistic regression function

$$h_w(\mathbf{x}) = g(w\mathbf{x}) = \frac{1}{1+e^{-w\mathbf{x}}} = \frac{1}{1+e^{-\sum_d w^d x^d}}$$

- transform outcome into "probabilities"

$$P(y=1|x;w) = h_w(x)$$

$$P(y=0|x;w) = 1 - h_w(x)$$

$$P(y|x;w) = (h_w(x))^y (1-h_w(x))^{1-y}$$

- objective = likelihood of observations
  - a.k.a how likely is the data observed, given the regression model
  - and take the log

$$\begin{aligned} L(w) &= p(y|X;w) \\ &= \prod_{i=1}^{m} p(y_i|x_i;w) \\ &= \prod_{i=1}^{m} (h_w(x_i))^{y_i} (1-h_w(x_i))^{1-y_i} \end{aligned}$$

$$\begin{aligned} l(w) &= \log L(w) \\ &= \sum_{i=1}^{m} y_i \log h(x_i) + (1-y_i)\log(1-h(x_i)) \end{aligned}$$

Logistic Regression

$$L(w) = p(y|X; w) = \prod_{i=1}^{m} p(y_i|x_i; w) = \prod_{i=1}^{m} (h_w(x_i))^{y_i} (1 - h_w(x_i))^{1-y_i}$$

- consider the likelihood of observations
  - and take the log

$$l(w) = \log L(w) = \sum_{i=1}^{m} y_i \log h(x_i) + (1 - y_i) \log(1 - h(x_i))$$

- maximize log likelihood using gradient ascent
  - one datapoint derivation

$$\begin{aligned}
\frac{\partial}{\partial w^j} l(w) &= (y \frac{1}{g(wx)} - (1-y) \frac{1}{1-g(wx)}) \frac{\partial}{\partial w^j} g(wx) \\
&= (y \frac{1}{g(wx)} - (1-y) \frac{1}{1-g(wx)}) g(wx)(1-g(wx)) \frac{\partial}{\partial w^j} wx \\
&= (y(1-g(gx)) - (1-y) g(wx)) x^j \\
&= (y - h(x)) x^j
\end{aligned}$$

- write down the update rules
  - batch or stochastic

the stochastic gradient ascent rule:

$$w^j := w^j + \lambda (y_i - h_w(x_i))) x_i^j$$

› the batch gradient ascent rule:

$$w^j := w^j + \lambda \sum_i (y_i - h_w(x_i))) x_i^j$$