Introduction to Machine Learning

Regression

Computer Science, Tel-Aviv University, 2013-14

Classification

Input: X

- Real valued, vectors over real.
- Discrete values (0,1,2,…)
- Other structures (e.g., strings, graphs, etc.)

Output: Y

- Discrete (0,1,2,…)

Regression

Input: X

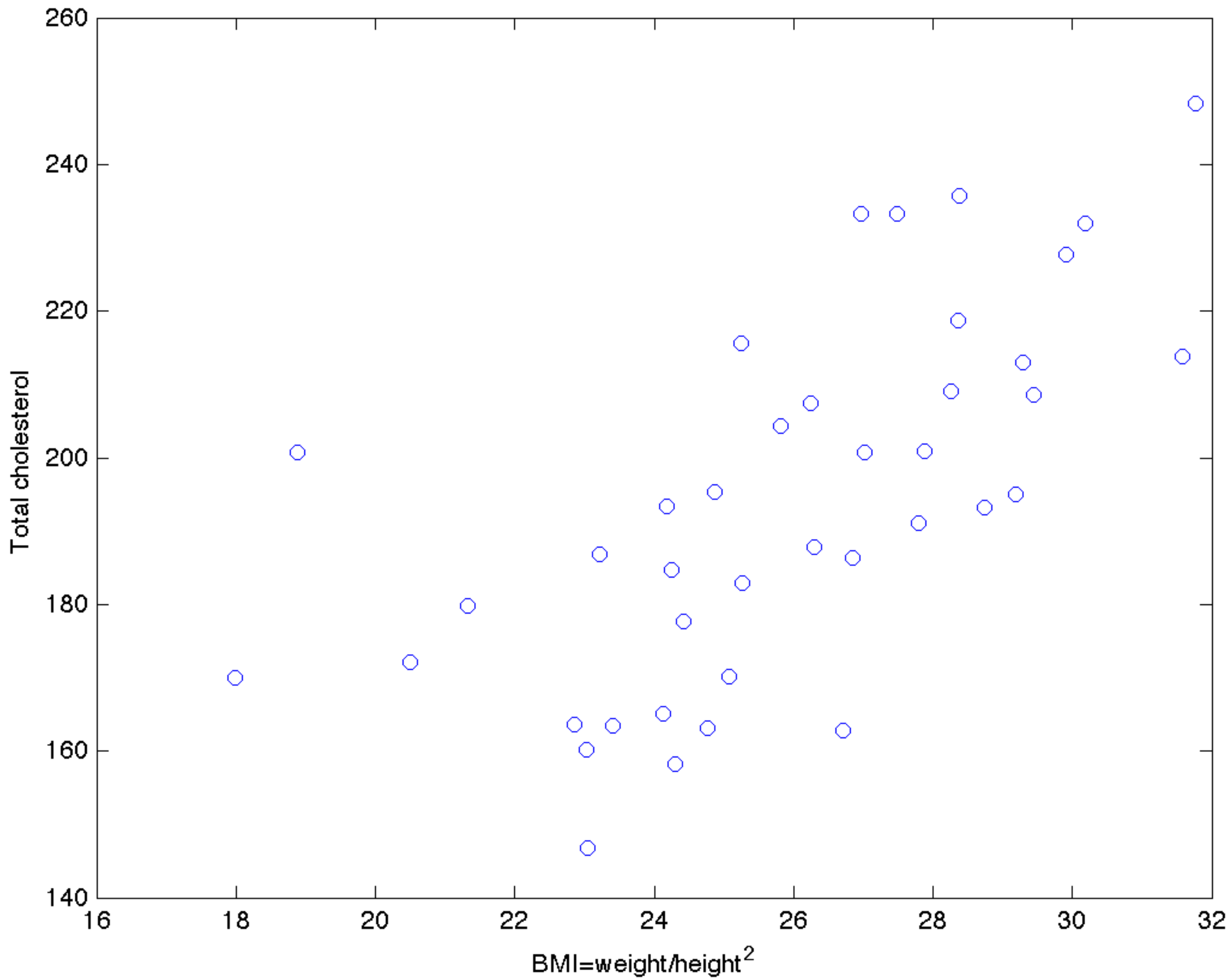
- Real valued, vectors over real.
- Discrete values (0,1,2,…)

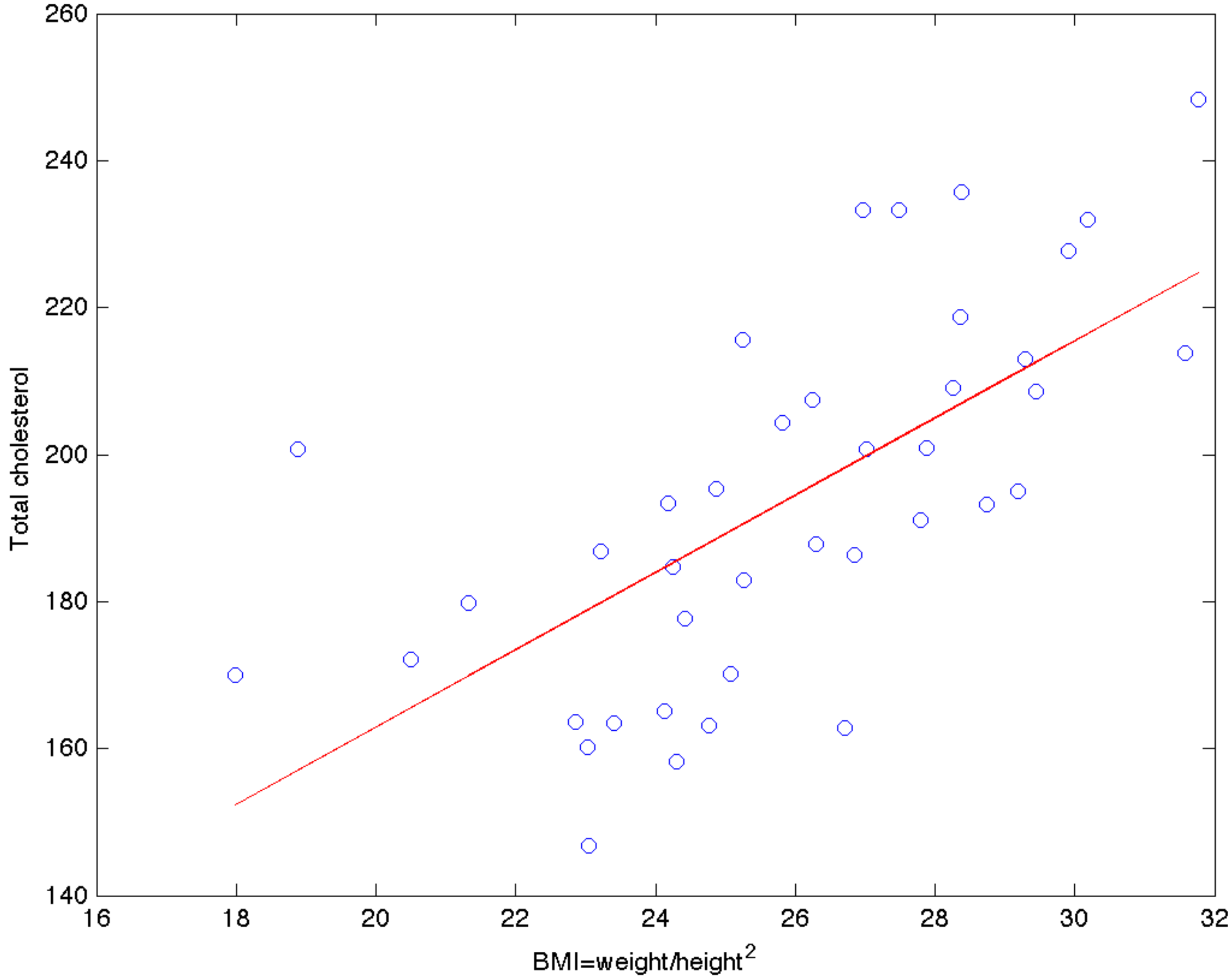
Output: Y

- Real valued, vectors over real.

Examples: Regression

- Weight + height → cholesterol level
- Age + gender → time spent in front of the TV
- Past choices of a user → 'Netflix score'
- Profile of a job (user, machine, time) → Memory usage of a submitted process.


260
240
220
200
180
160
140
16
18
20
22
24
26
28
30
32
Total cholesterol
BMI=weight/height^2



260
240
220
200
180
160
140
16
18
20
22
24
26
28
30
32
Total cholesterol
BMI=weight/height^2

Linear Regression

Input: A set of points (x_i, y_i)

- Assume there is a linear relation between y and x.

$$y \approx ax + b$$

Linear Regression

Input: A set of points (x_i, y_i)

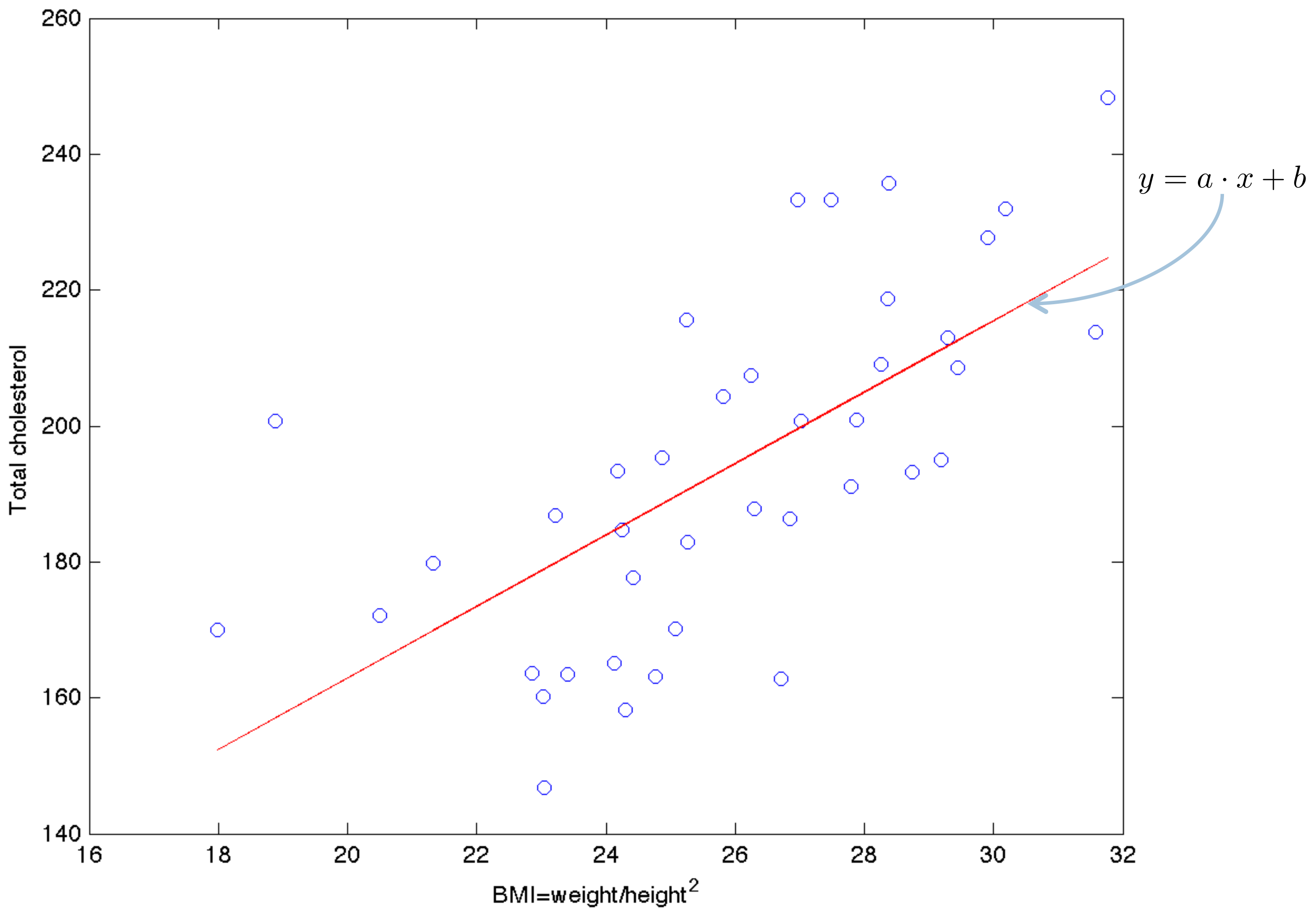
- Assume there is a linear relation between y and x.

$$y \approx ax + b$$

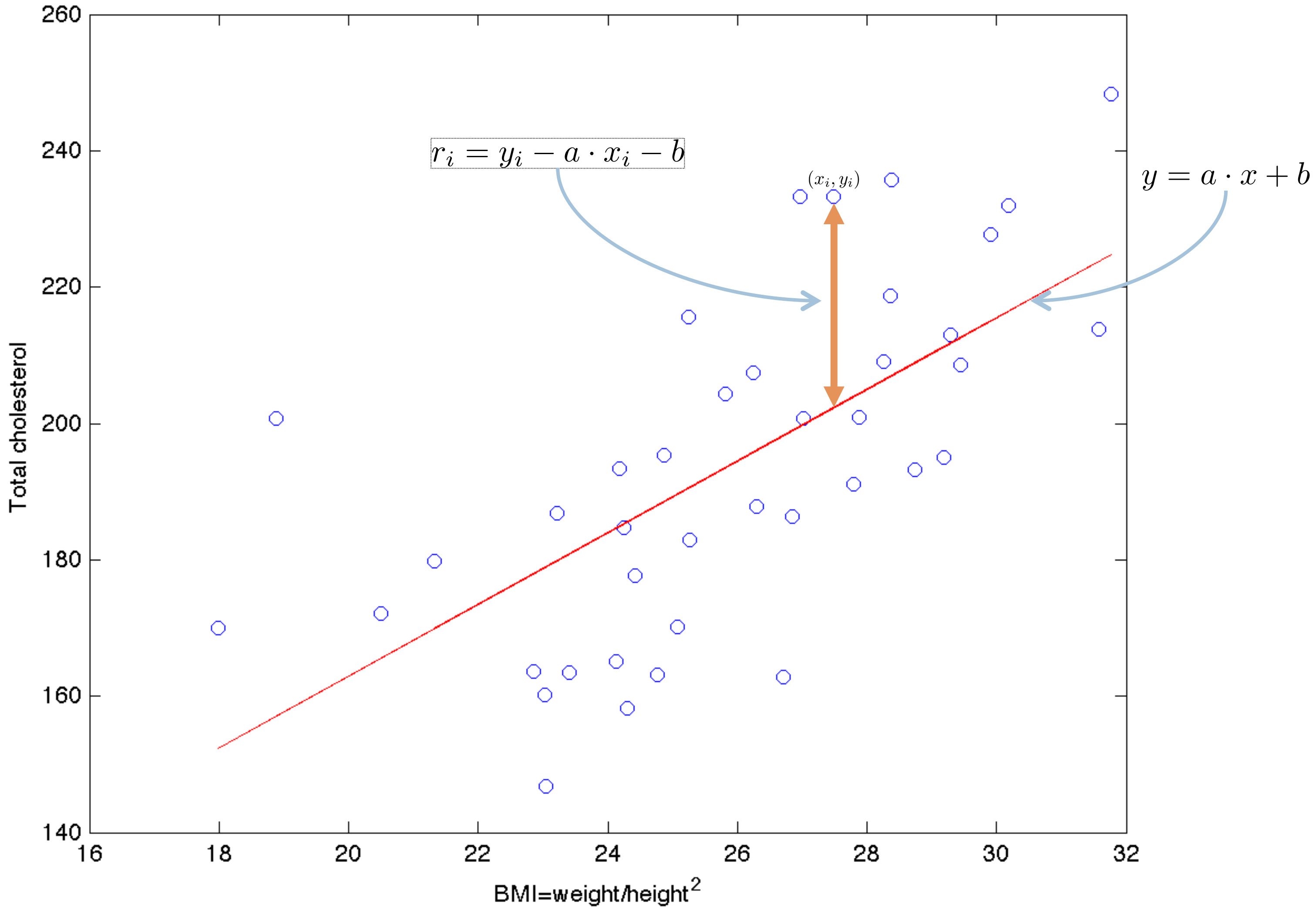
- Find a,b by solving

$$(a, b) = \arg\min_{a,b} \sum_i (y_i - ax_i - b)^2$$

Regression: Minimize the Residuals



Regression: Minimize the Residuals



Likelihood Formulation

Model assumptions:

$$y = ax + b + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$

Input:

$$Data = (x_1, y_1), \ldots, (x_n, y_n)$$

Likelihood Formulation

Model assumptions:

$$y = ax + b + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$

Input:

$$Data = (x_1, y_1), \ldots, (x_n, y_n)$$

$$\log L(Data; a, b, \sigma) = -\frac{1}{2\sigma^2}\sum_i (y_i - ax_i - b)^2 - \frac{n}{2}\log(2\pi\sigma^2)$$

Likelihood Formulation

Model assumptions:

$$y = ax + b + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$

Input:

$$Data = (x_1, y_1), \ldots, (x_n, y_n)$$

$$\log L(Data; a, b, \sigma) = -\frac{1}{2\sigma^2} \sum_i (y_i - ax_i - b)^2 - \frac{n}{2} \log(2\pi\sigma^2)$$

Likelihood maximized when $(a, b) = \arg\min_{a,b} \sum_i (y_i - ax_i - b)^2$

$$y = ax + b + \epsilon$$

Note:

$$a, x_i \in R^d \quad y, b \in R$$

We can add another variable x_{d+1}=1, and set a_{d+1}=b.

Therefore, without loss of generality

$$y = a \cdot x + \epsilon$$

Matrix Notations

$$f(a) = \sum_i (y_i - a \cdot x_i)^2$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad a = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix} \qquad x_i = (x_{i1} \ldots x_{id})$$

$$X = \begin{pmatrix} - x_1 - \\ \cdots \\ \cdots \\ \cdots \\ - x_n - \end{pmatrix} = \begin{pmatrix} | & \cdots & | \\ X_1 & \cdots & X_d \\ | & \cdots & | \end{pmatrix}$$

The Normal Equations

$$f(a) = \sum_i (y_i - a \cdot x_i)^2$$

$$\frac{\partial f}{\partial a_j} = -2 \sum_i (y_i - a \cdot x_i) x_{ij} = -2 X_j^t (y - Xa)$$

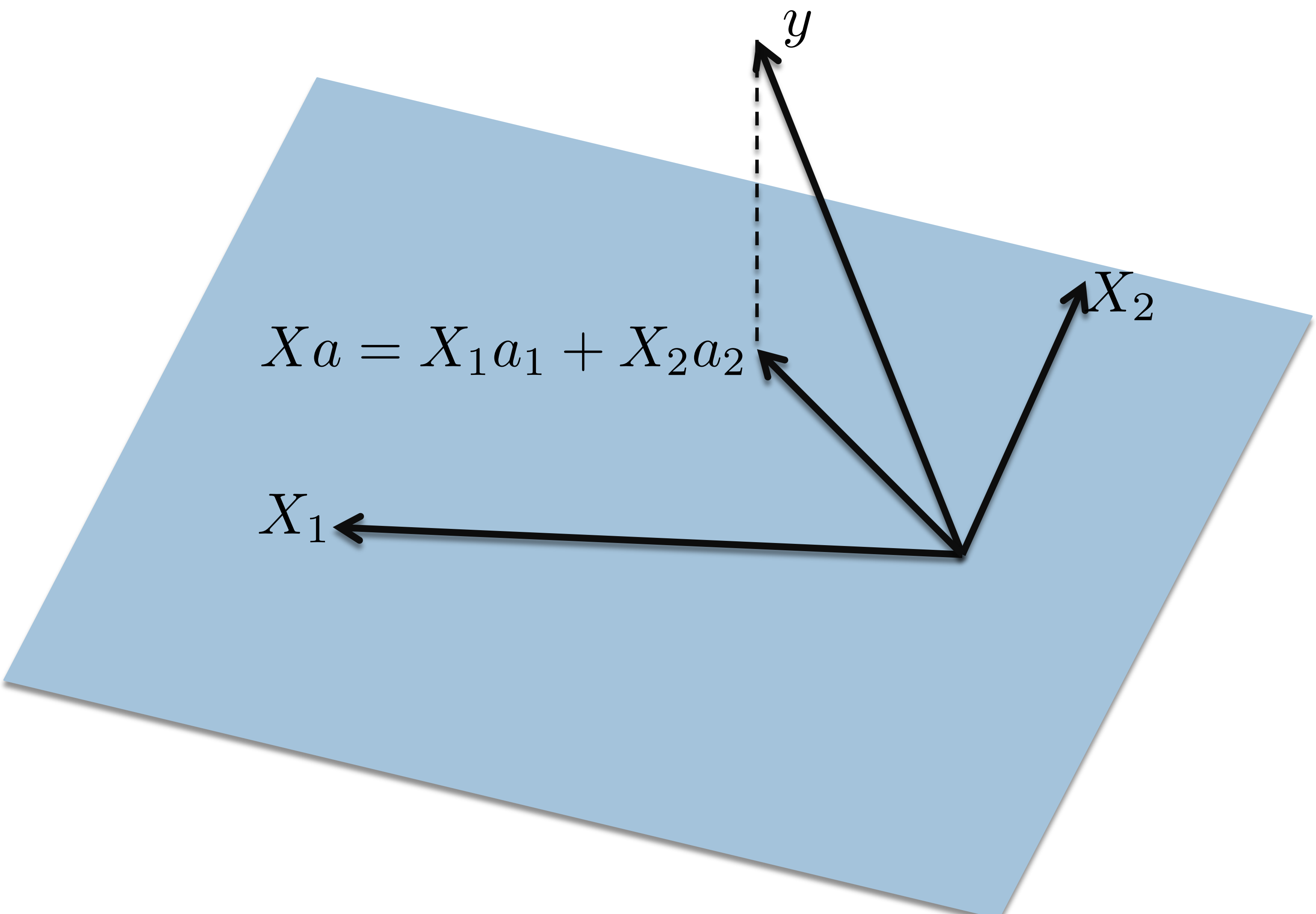
$$\Downarrow$$

$$X^t y = X^t X a$$

$$\Downarrow$$

$$a = (X^t X)^{-1} X^t y$$

$$X^t(y - Xa) = 0$$



Functions over n-dimensions

For a function $f(x_1, \ldots, x_n)$, the gradient is the vector of partial derivatives:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

In one dimension: derivative.

Gradient Descent

- Goal: Minimize a function $f(x_1, \ldots, x_n)$
- Algorithm:

1. Start from a point $(x_1^0, \ldots, x_n^0)$
2. Compute $u = \nabla f(x_1^i, \ldots, x_n^i)$
3. Update $x^{i+1} := x^i - \alpha \cdot u$
4. Return to (2), unless converged.

Gradient Descent

$$f(a) = \sum_i (y_i - a \cdot x_i)^2$$

$$\frac{\partial f}{\partial a_j} = -2 \sum_i (y_i - a \cdot x_i) x_{ij} = -2 X_j^t (y - Xa)$$

Gradient Descent iteration:

$$a_{k+1} = a_k + \alpha X^t (y - X a_k)$$

Advantage: simple, efficient.

Online Least Squares

$$f(a) = \sum_i (y_i - a \cdot x_i)^2$$

$$\frac{\partial f}{\partial a_j} = -2 \sum_i (y_i - a \cdot x_i) x_{ij} = -2 X_j^t (y - Xa)$$

Online update step:

$$a_{k+1} = a_k + \alpha (y_{k+1} - x_{k+1} a_k) x_{k+1}^t$$

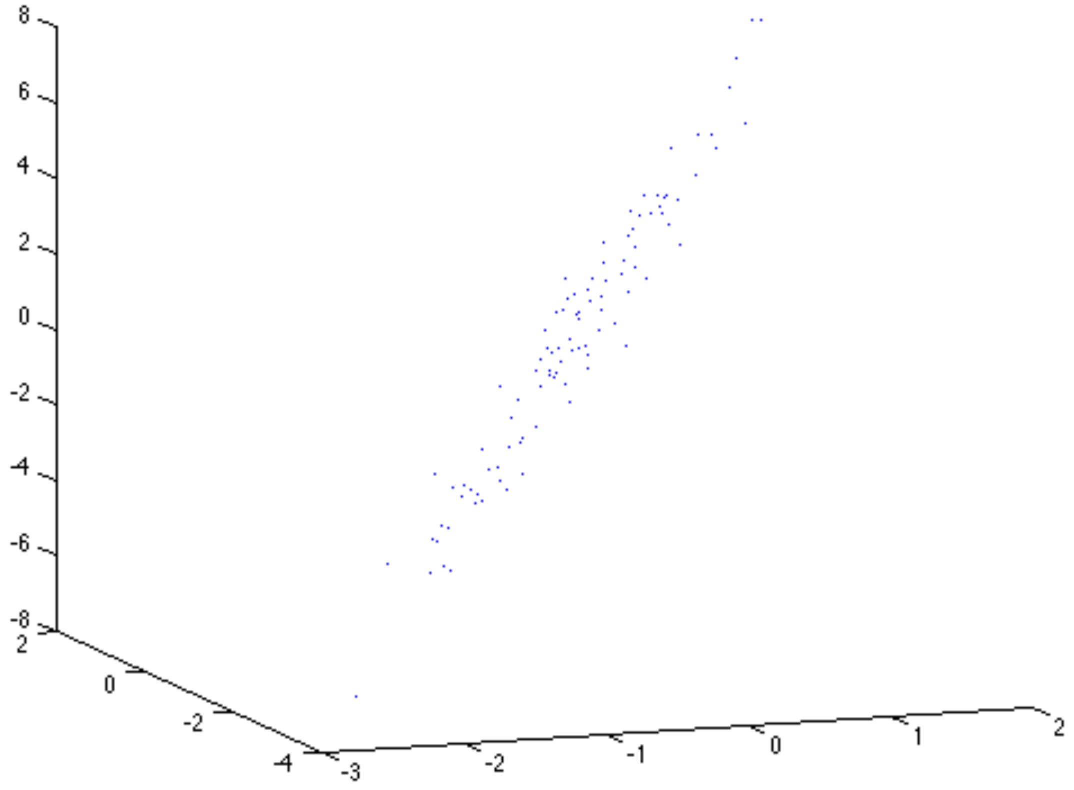
Advantage: Efficient, similar to perceptron.

Singularity issues

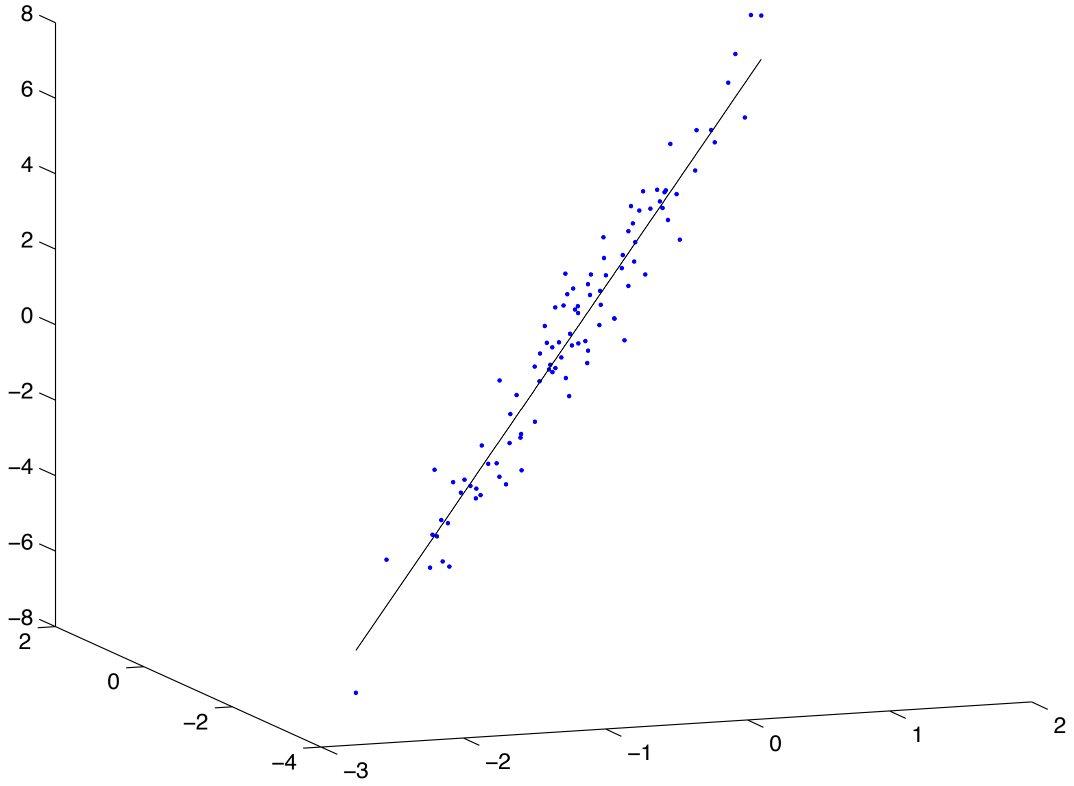
$$a = (X^t X)^{-1} X^t y$$

- Not very efficient since we need to inverse a matrix.
- The solution is unique if $X^t X$ is invertible.
- If $X^t X$ is singular, we have an infinite number of solutions to the equations. What is the solution minimizing $||y - Xa||_2$?

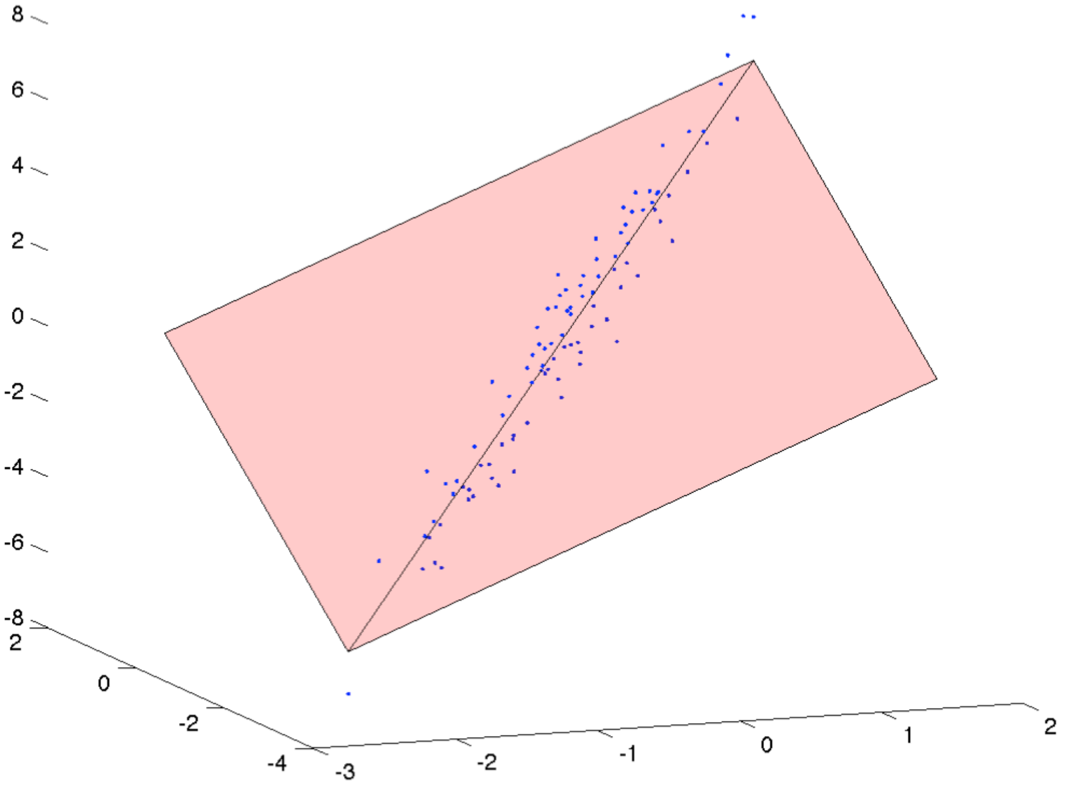
The Singular Case



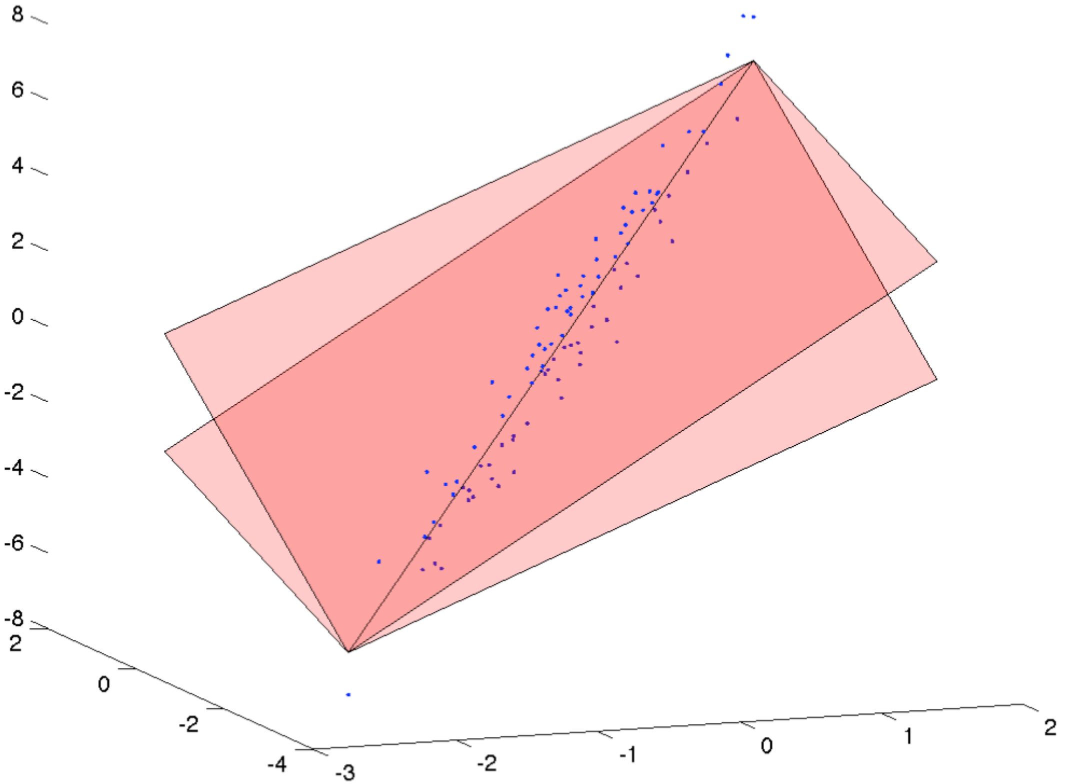
The Singular Case



The Singular Case



The Singular Case



Risk of Overfitting

- Say we have a very large number of variables (d is large).
- When the number of variables is large we usually have co-linear variables, and therefore $X^t X$ is singular.
- Even if $X^t X$ is non-singular, there is a risk for over-fitting. For instance, if d=n we can get

$$a = X^{-1} y \longrightarrow \|y - Xa\|_2 = 0$$

- Intuitively, we want a small number of variables to explain y.

Regularization

Let λ be a regularization parameter. Ideally, we need to solve the following:

$$\hat{a} = \arg\min_{a}\{\|y - Xa\|_2^2 + \lambda\|a\|_0\}$$

This is a hard problem (NP-hard).

Shrinkage Methods

Lasso regression:

$$\hat{a} = \arg\min_{a}\{\|y - Xa\|_2^2 + \lambda\|a\|_1\}$$

Ridge regression:

$$\hat{a} = \arg\min_{a}\{\|y - Xa\|_2^2 + \lambda\|a\|_2^2\}$$

Ridge Regression

$$\hat{a} = \arg\min_a \{\sum_{i=1}^{n}(y_i - x_i \cdot a)^2 + \lambda \sum_i a_i^2\}$$

$$\frac{\partial f}{\partial a_j} = -2X_j^t(y - Xa) + 2\lambda a_j$$

Ridge Regression

$$\hat{a} = \arg\min_a \{ \sum_{i=1}^{n} (y_i - x_i \cdot a)^2 + \lambda \sum_i a_i^2 \}$$

$$\frac{\partial f}{\partial a_j} = -2X_j^t(y - Xa) + 2\lambda a_j$$

$$\Downarrow$$

$$X^t y = (X^t X + \lambda I)a$$

Ridge Regression

$$\hat{a} = \arg\min_a \{\sum_{i=1}^{n}(y_i - x_i \cdot a)^2 + \lambda \sum_i a_i^2\}$$

$$\frac{\partial f}{\partial a_j} = -2X_j^t(y - Xa) + 2\lambda a_j$$

$$X^t y = (X^t X + \lambda I)a$$

$$\hat{a} = (X^t X + \lambda I)^{-1} X^t y$$

Positive definite and therefore nonsingular

Ridge Regression – Bayesian View

$$y = \sum_j a_j X_j + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

$$a_j \sim N(0, \tau^2)$$ ← Prior on a

Ridge Regression – Bayesian View

$$y = \sum_j a_j x_j + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

$$a_j \sim N(0, \tau^2)$$ Prior on a

$$\log Posterior(a \mid \sigma, \tau, Data) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - a \cdot x_i)^2 - \frac{1}{2\tau^2}\sum_{i=1}^{d} a_i^2 - \frac{n}{2}\log(2\pi\sigma^2) - \frac{d}{2}\log(2\pi\tau^2)$$

Maximizing the posterior is equivalent to Ridge with $\lambda = \frac{\sigma^2}{\tau^2}$

Lasso Regression

$$\hat{a} = \arg\min_a \{\|y - Xa\|_2^2 + \lambda \|a\|_1\}$$

$$\min_a \{a^t (X^t X) a - 2y^t X a + \lambda \sum_i |a_i|\}$$

Lasso Regression

$$\hat{a} = \arg\min_a \{\|y - Xa\|_2^2 + \lambda \|a\|_1\}$$

$$\min_a \{a^t (X^t X) a - 2y^t X a + \lambda \sum_i |a_i|\}$$

The above is equivalent to the following quadratic program:

$$\boxed{\begin{aligned} & min_{a,b} \{a^t (X^t X) a - 2y^t X a + \lambda \sum_{i=1}^{d} b_i\} \\ s.t. \quad & b_i \geq a_i, \qquad i = 1, \ldots, d \\ & b_i \geq -a_i, \qquad i = 1, \ldots, d \end{aligned}}$$

Lasso Regression – Bayesian View

$$y = \sum_j a_j x_j + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

$$a_j \sim Laplace(0, \frac{2\sigma^2}{\lambda})$$

Prior on a

$$pdf(a_j) = \frac{\lambda}{4\sigma^2} e^{-\frac{\lambda}{2\sigma^2}|a_j|}$$

Lasso Regression – Bayesian View

$$y = \sum_j a_j x_j + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

$$a_j \sim Laplace(0, \frac{2\sigma^2}{\lambda})$$

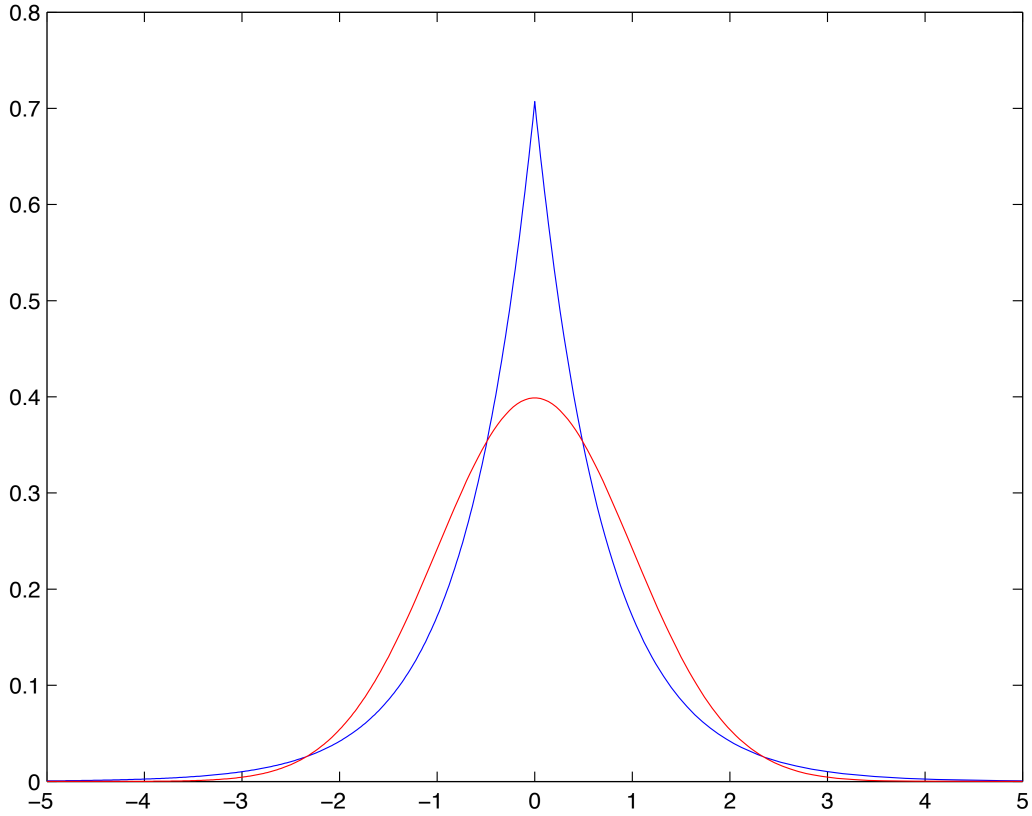
Prior on a

$$pdf(a_j) = \frac{\lambda}{4\sigma^2} e^{-\frac{\lambda}{2\sigma^2}|a_j|}$$

$$\log Posterior(a \mid Data, \lambda, \sigma) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - a \cdot x_i)^2 - \frac{n}{2}\log(2\pi\sigma^2)$$

$$+\log(\frac{\lambda}{4\sigma^2}) - \frac{\lambda}{2\sigma^2}\sum_{i=1}^{d}|a_i|$$

Lasso vs. Ridge



Laplace vs. Normal priors (mean 0, variance 1)

An Equivalent Formulation

Lasso:

$$\min_a \|y - Xa\|_2$$

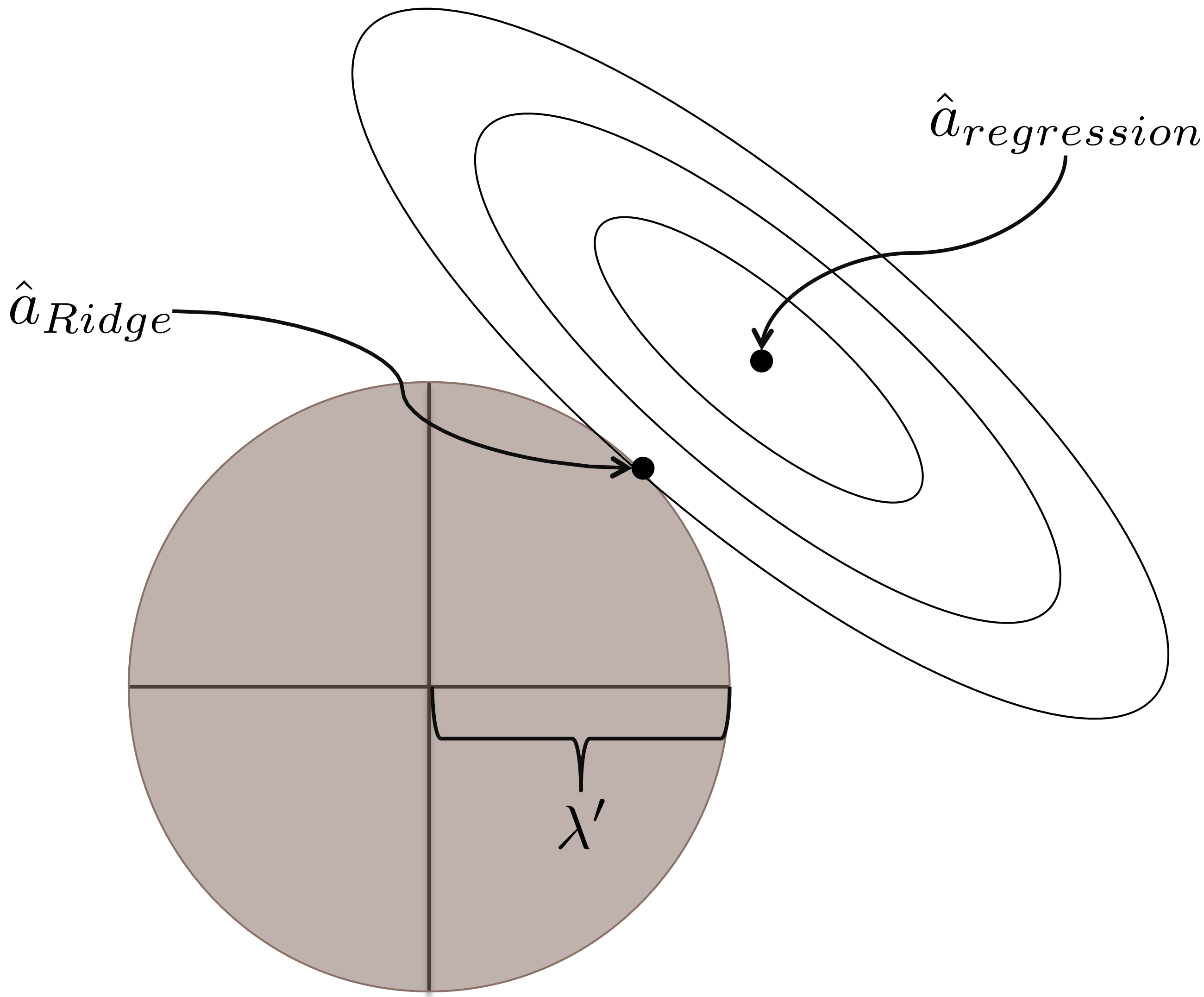
$$s.t. \quad \sum_i |a_i| \leq \lambda'$$

Ridge:

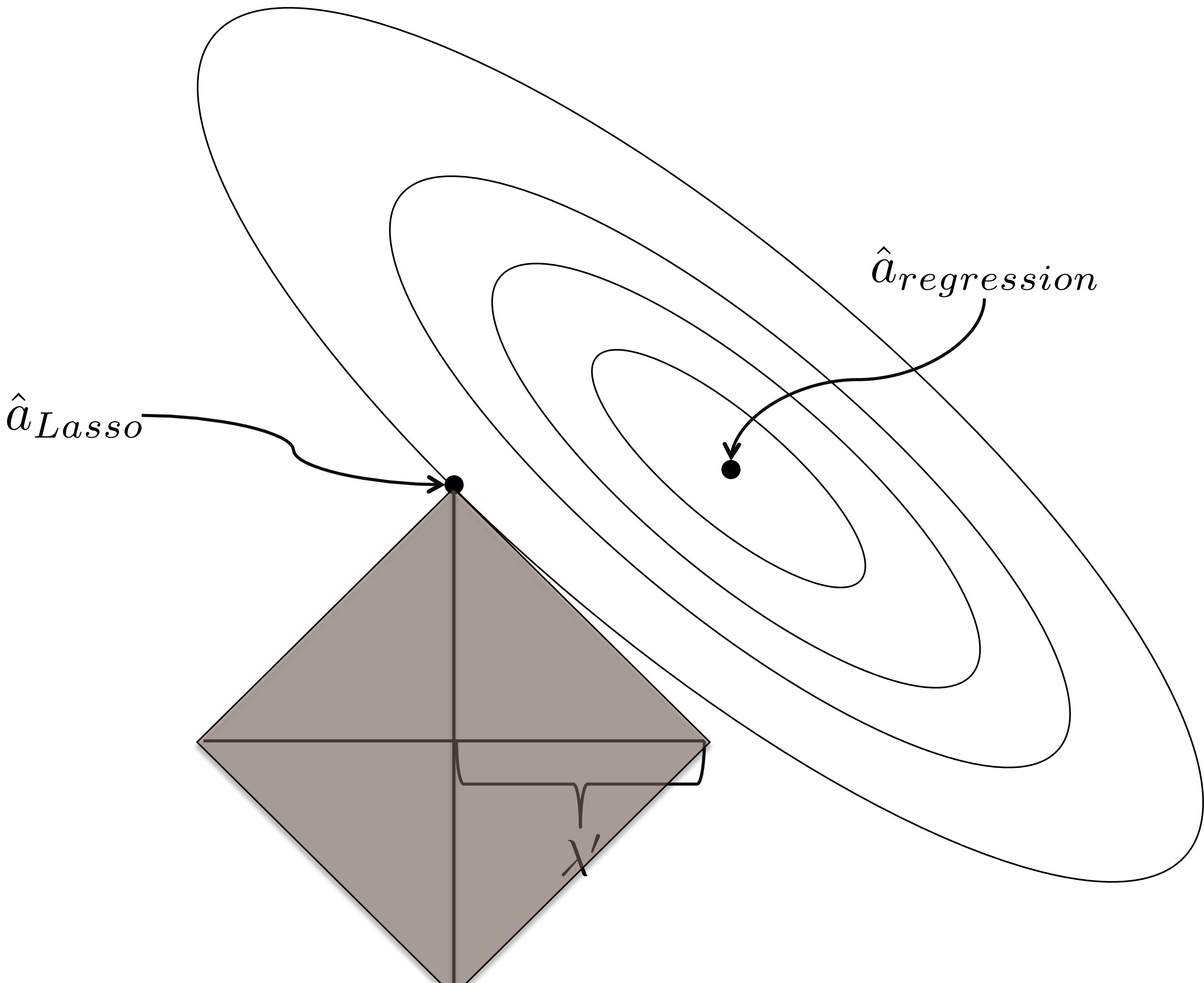
$$\min_a \|y - Xa\|_2$$

$$s.t. \quad \sum_i a_i^2 \leq \lambda'$$

Claim: for every λ there is λ' that produces the same solution $\hat{a}$



$\hat{a}_{regression}$
$\hat{a}_{Ridge}$
λ'


$\hat{a}_{regression}$
$\hat{a}_{Lasso}$
λ'

Breaking Linearity

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + a_3 e^{x_i} + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & e^{x_1} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & e^{x_n} \end{pmatrix}$$

This can be solved using the usual linear regression by plugging X

Regression for Classification

Input: X

- Real valued, vectors over real.
- Discrete values (0,1,2,…)
- Other structures (e.g., strings, graphs, etc.)

Output: Y

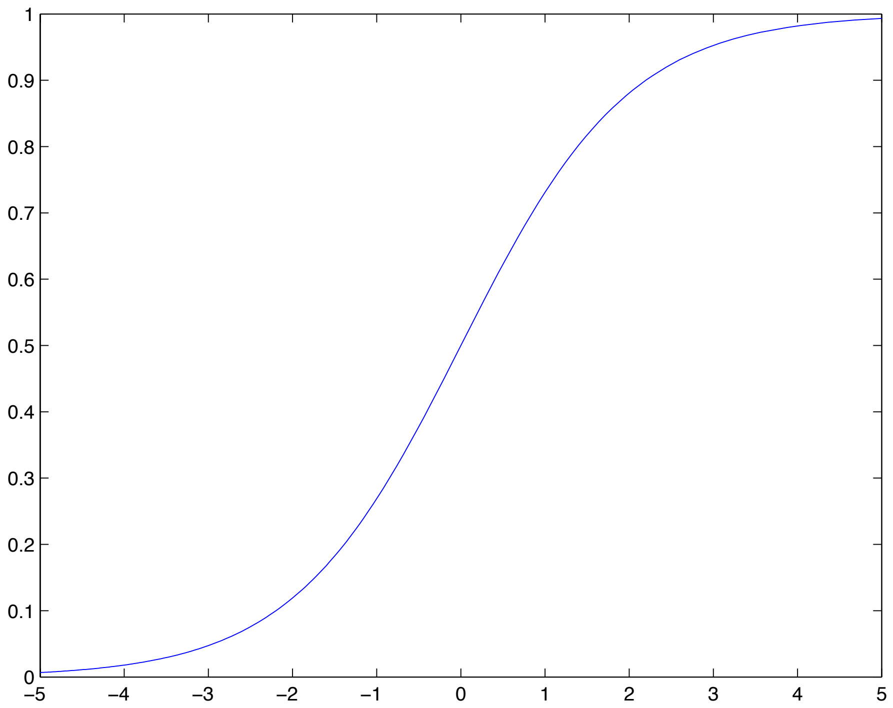
- Discrete (0 or 1)

We treat the probability Pr(Y|X) as a linear function of X.

Problem: Pr(Y|X) should be bounded in [0,1].

Logistic Regression

Model: $$Pr(y = 1 \mid x) = \frac{e^{a \cdot x}}{1 + e^{a \cdot x}}$$



Logistic Regression

Given training data, we can write down the likelihood:

$$L(a; (x_1, y_1), \ldots, (x_n, y_n)) = \prod_{i=1}^{n} \frac{e^{y_i a \cdot x_i}}{1 + e^{a \cdot x_i}}$$

$$\log L(a; (x_1, y_1), \ldots, (x_n, y_n)) = \underbrace{a \cdot \sum_{i=1}^{n} y_i x_i}_{\text{linear}} - \underbrace{\sum_{i=1}^{n} \log(1 + e^{a \cdot x_i})}_{\text{concave}}$$

There is a unique solution – can be found using gradient descent.