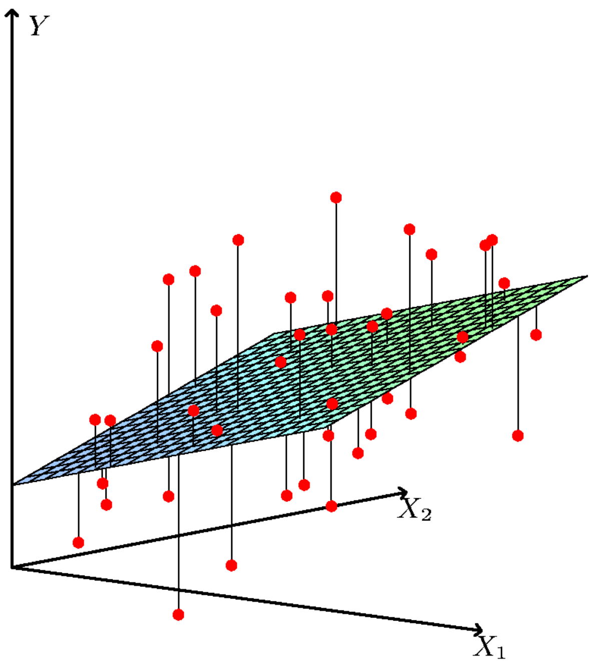
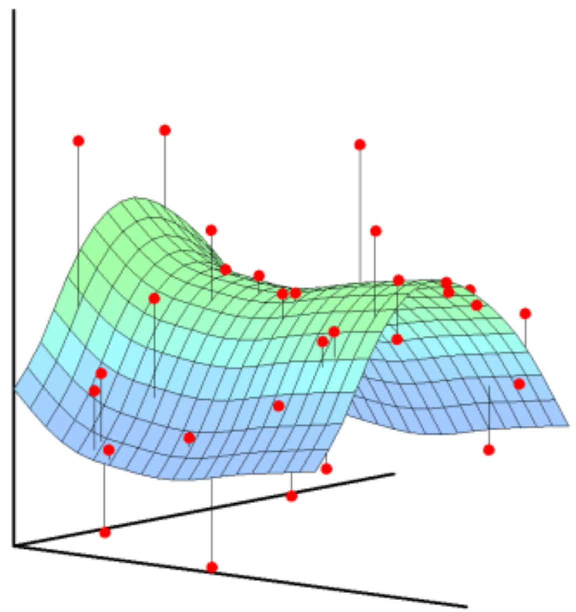
Introduction to Supervised Learning





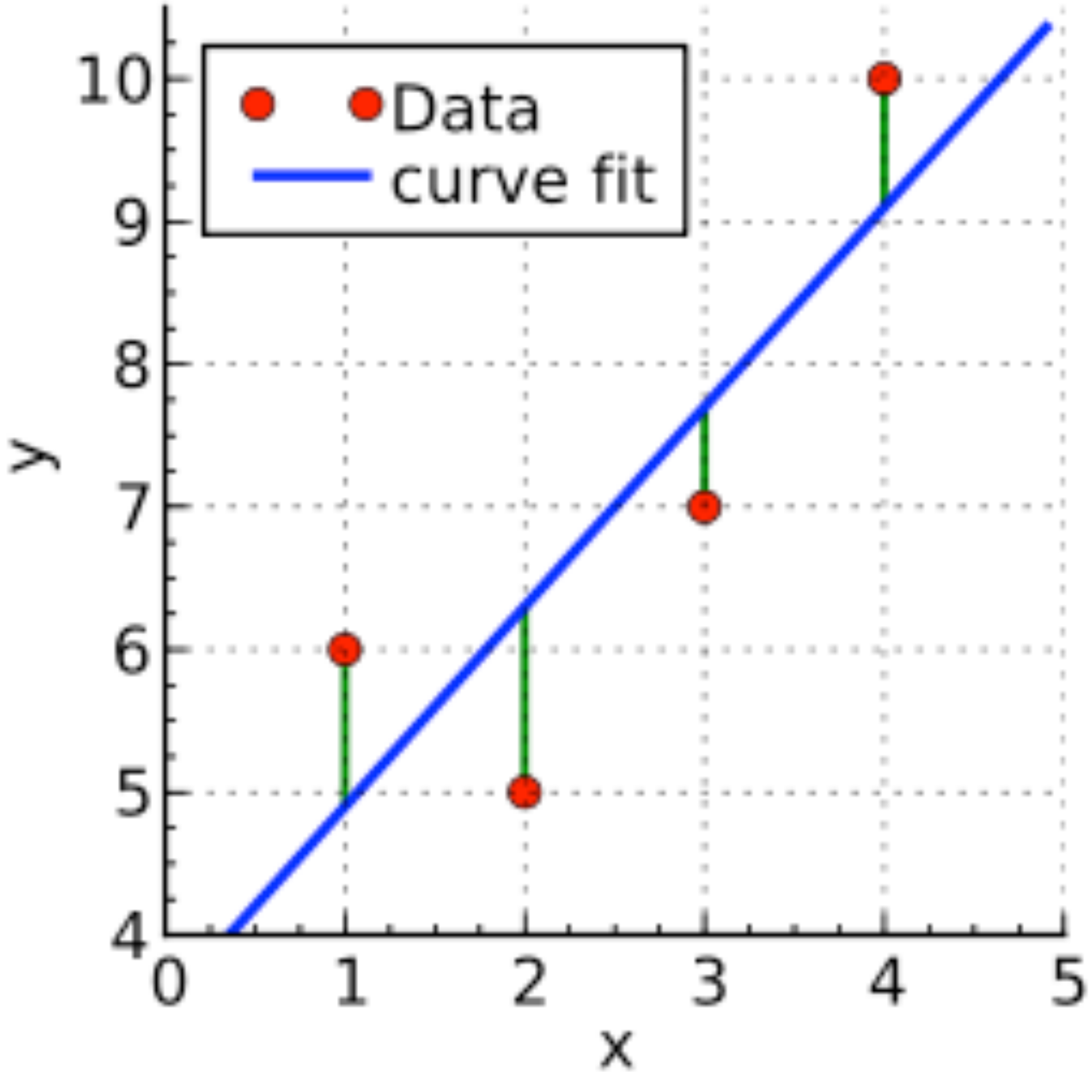
Week 2: Maximum Likelihood Parameter Estimation,
Linear Regression

Iasonas Kokkinos

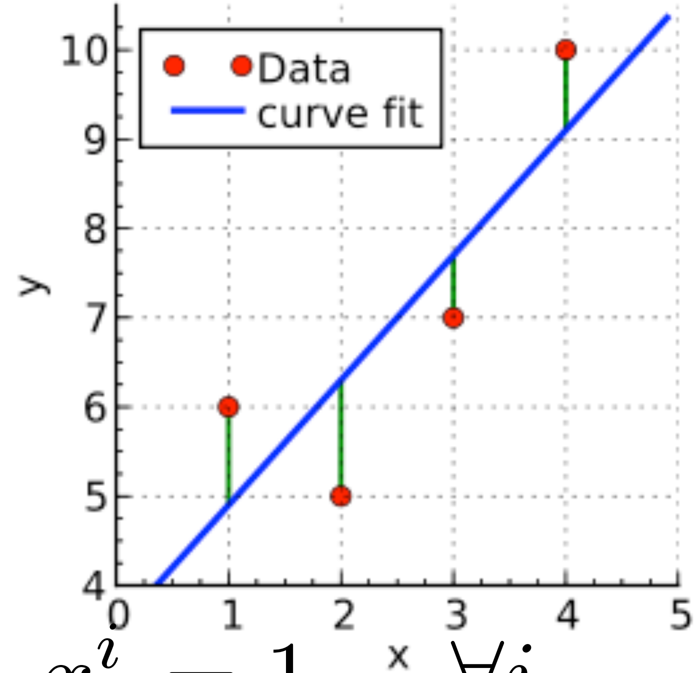
i.kokkinos@cs.ucl.ac.uk

University College London

Linear regression in 1D



Linear regression in 1D



$$
\begin{aligned}
y^i &= w_0 + w_1 x_1^i + \epsilon^i \\
&= w_0 x_0^i + w_1 x_1^i + \epsilon^i, \quad x_0^i = 1, \ \forall i \\
&= \mathbf{w}^T \mathbf{x}^i + \epsilon^i
\end{aligned}
$$

$$L(\mathbf{w}) = \sum_{i=1}^{N} (\epsilon^i)^2$$

$$L(w_0, w_1) = \sum_{i=1}^{N} \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right]^2$$

Fitting a line

$$L(w_0, w_1) = \sum_{i=1}^{N} \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right]^2$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = \sum_{i=1}^{N} \frac{\partial \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right]^2}{\partial w_0}$$

$$= \sum_{i=1}^{N} 2 \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right] \left(-x_0^i \right)$$

$$= -2 \sum_{i=1}^{N} \left(y^i x_0^i - w_0 x_0^i x_0^i - w_1 x_1^i x_0^i \right)$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = 0_N \quad \Leftrightarrow$$

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

Fitting a line, continued

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = 0 \quad \Leftrightarrow$$

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

$$\frac{\partial L(w_0, w_1)}{\partial w_1} = 0 \quad \Leftrightarrow$$

$$\sum_{i=1}^{N} y^i x_1^i = w_0 \sum_{i=1}^{N} x_0^i x_1^i + w_1 \sum_{i=1}^{N} x_1^i x_1^i$$

2 linear equations, 2 unknowns

Fitting a line, continued

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

$$\sum_{i=1}^{N} y^i x_1^i = w_0 \sum_{i=1}^{N} x_0^i x_1^i + w_1 \sum_{i=1}^{N} x_1^i x_1^i$$

2x2 system of equations:

$$\left[\begin{array}{c} \sum_{i=1}^{N} y^i x_0^i \\ \sum_{i=1}^{N} y^i x_1^i \end{array} \right] = \left[\begin{array}{cc} \sum_{i=1}^{N} x_0^i x_0^i & \sum_{i=1}^{N} x_0^i x_1^i \\ \sum_{i=1}^{N} x_0^i x_1^i & \sum_{i=1}^{N} x_1^i x_1^i \end{array} \right] \left[\begin{array}{c} w_0 \\ w_1 \end{array} \right]$$

Fitting a line, continued

2x2 system of equations:

$$\begin{bmatrix} \sum_{i=1}^{N} y^i x_0^i \\ \sum_{i=1}^{N} y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_0^i x_0^i & \sum_{i=1}^{N} x_0^i x_1^i \\ \sum_{i=1}^{N} x_0^i x_1^i & \sum_{i=1}^{N} x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

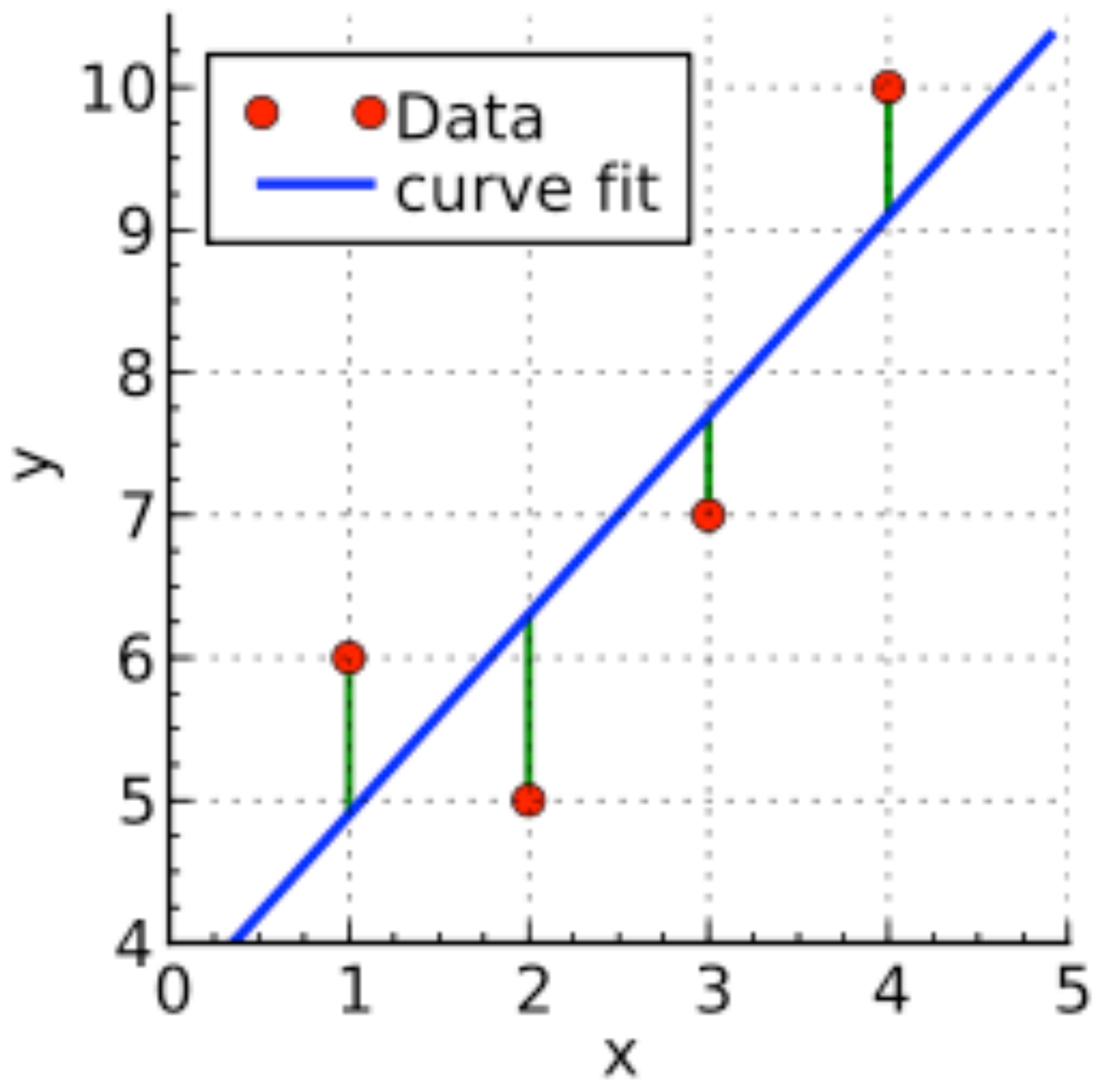
Or, without summations:

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w}$$

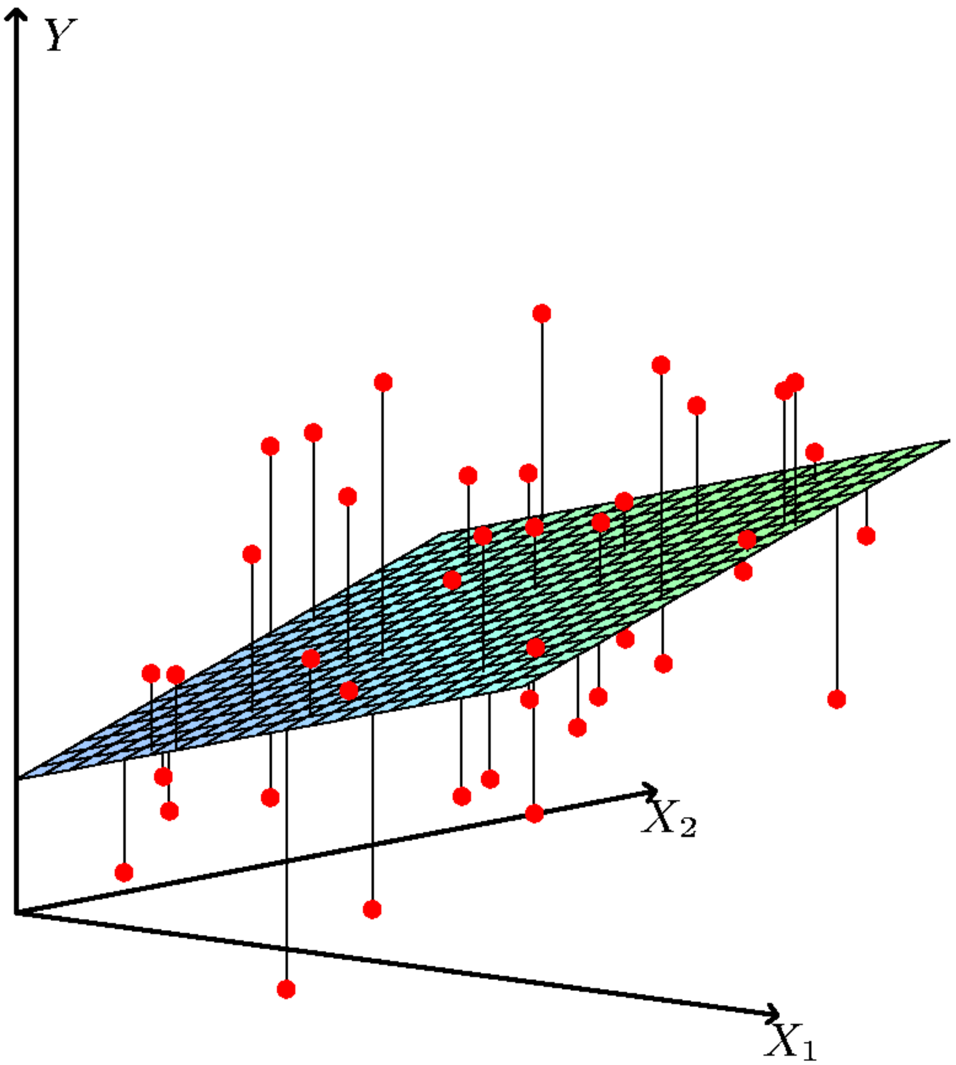
$$\mathbf{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_0^1 & x_1^1 \\ \vdots & \vdots \\ x_0^N & x_2^N \end{bmatrix}$$

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

Linear regression in 1D



Linear regression in 2D (or ND)



What we would like to be happening

Training set: $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \ldots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

If N>D (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

What is happening: approximations

Training set: $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \ldots, N$

$$y^1 \simeq w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 \simeq w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

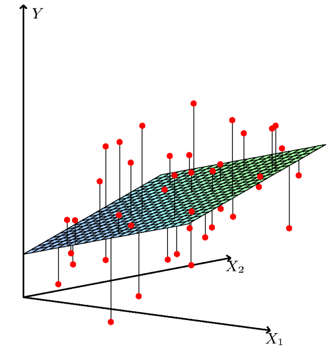
$$\vdots$$

$$y^N \simeq w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

If N>D (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

Goal: fit outputs with linear function



Training set: $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \ldots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1 + \epsilon^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2 + \epsilon^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N + \epsilon^N$$

Objective: minimize sum of squared residuals

$$L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$$

Least squares solution for linear regression

$$
\begin{aligned}
y^1 &= w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1 + \epsilon^1 \\
y^2 &= w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2 + \epsilon^2 \\
&\vdots \\
y^N &= w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N + \epsilon^N
\end{aligned}
$$

Least squares solution for linear regression

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} x_0^1 & x_1^1 & \dots & x_D^1 \\ x_0^2 & x_2^2 & \dots & x_D^2 \\ \vdots & & & \\ x_0^N & x_2^N & \dots & x_D^N \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Nx1 **Nx(D+1)** **(D+1)x1** **Nx1**

Least squares solution for linear regression

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$L(\mathbf{w}) = \left[\begin{array}{cccc} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{array}\right] \left[\begin{array}{c} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{array}\right]$$

Least squares solution for linear regression

Loss function: $$L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$$

$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

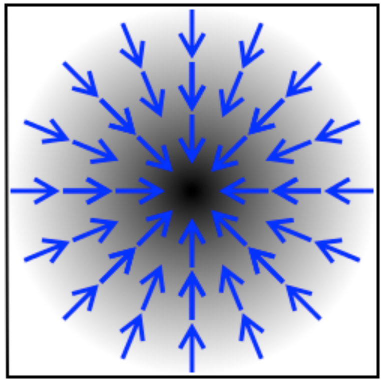
$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

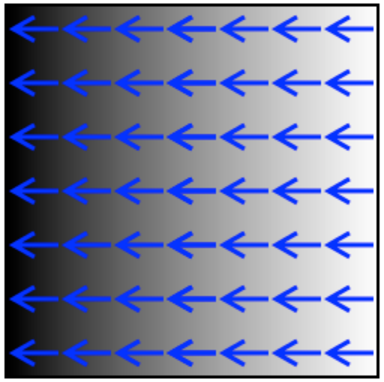
Gradient reminder

multivariate function: $f(x_1, \ldots, x_N)$

gradient:
$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix}$$

at extremum: $\nabla f = \mathbf{0}$





Minimizing the sum of squared errors in D-dimensions

$$L(\mathbf{w}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}$$

Condition for minimum:

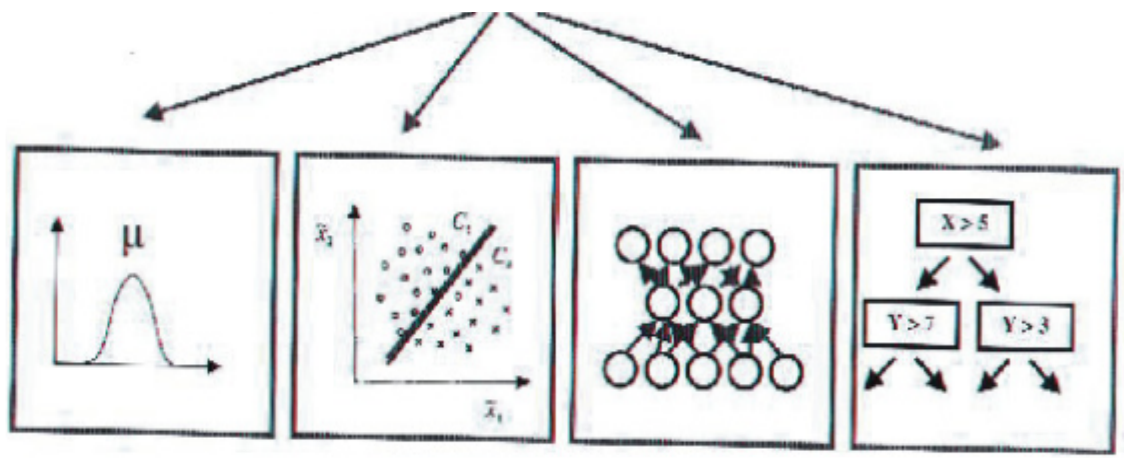
$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

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

Classifier function

- Input-output mapping
  - Output: y
  - Input: x
  - Method: f
  - Parameters: w

$$y = f_w(x) \quad (= f(x, w))$$

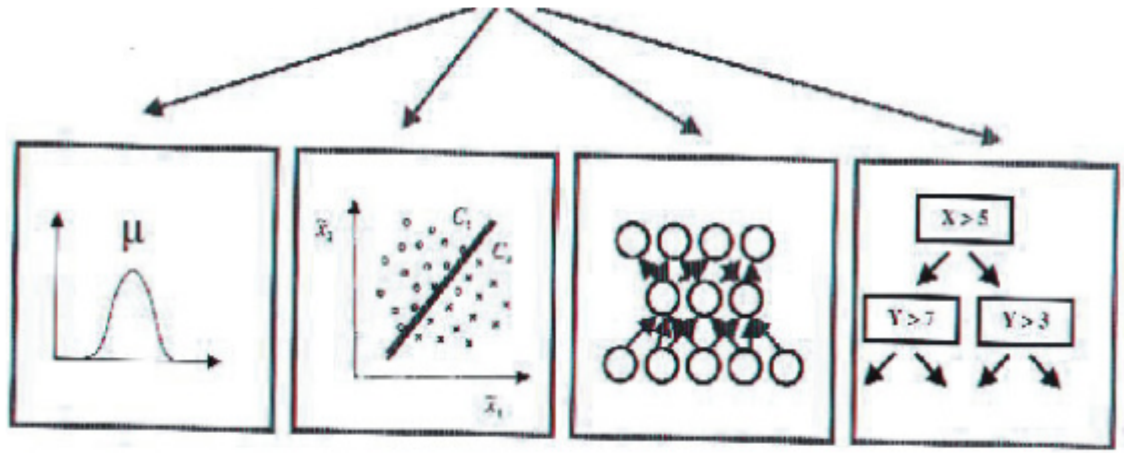


- Aspects of the learning problem
  - Identify methods that fit the problem setting
  - Determine parameters that properly classify the training set
  - Measure and control the `complexity' of these functions

Classifier function

- Input-output mapping
  - Output: y
  - Input: x
  - Method: f
  - Parameters: w

$$y = f_w(x) \quad (= f(x, w))$$



- Aspects of the learning problem
  - Identify methods that fit the problem setting
  - Determine parameters that properly classify the training set
  - Measure and control the `complexity’ of these functions

Done, for this method, and this notion of ‘proper’

Questions

Is the loss function appropriate for classification?

Quadratic loss: convex cost, closed-form solution

But does the optimized quantity indicate classifier's performance?

Is the classifier appropriate?

Linear classifier: fast computation

But could e.g. a non-linear classifier have better performance?

Are the estimated parameters good?

Parameters recover input-output mapping on training data

How can we know they do not simply memorize training data?

Questions

Is the loss function appropriate **for classification**?

Quadratic loss: convex cost, closed-form solution

But does the optimized quantity indicate classifier's performance?

Is the classifier appropriate?

Linear classifier: fast computation

But could e.g. a non-linear classifier have better performance?

Are the estimated parameters good?

Parameters recover input-output mapping on training data

How can we know they do not simply memorize training data?

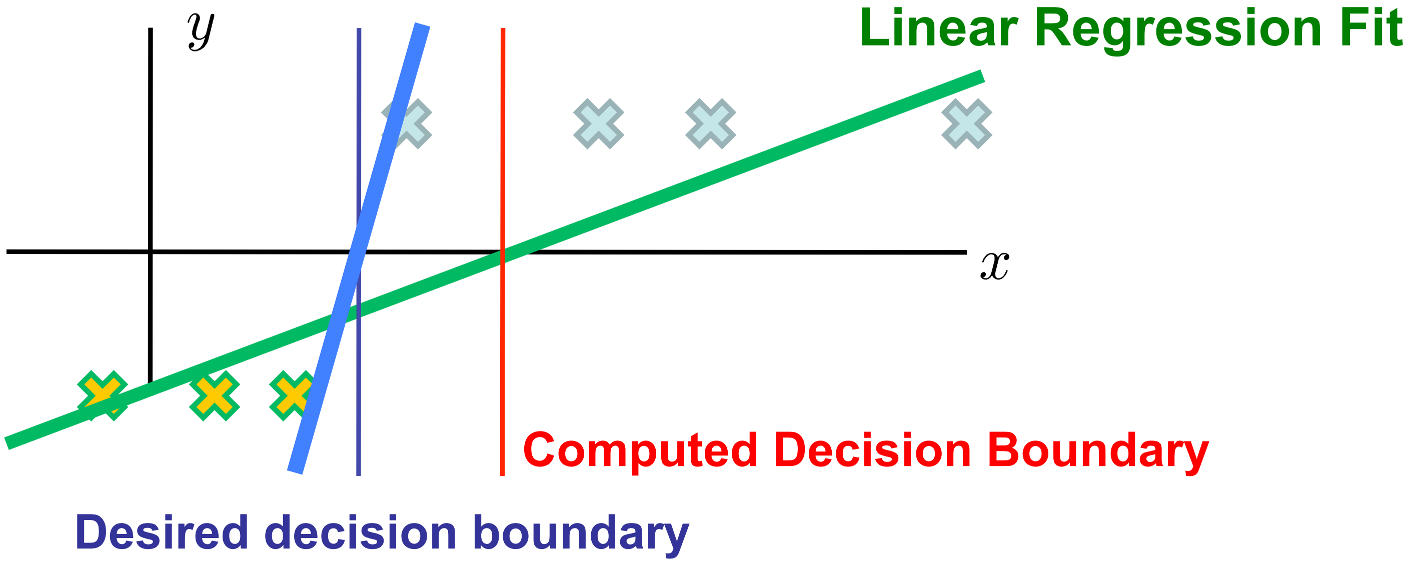
Inappropriateness of quadratic penalty

We chose the quadratic cost function for convenience

Single, global minimum & closed form expression

But does it indicate classification performance?

'Bad fit' according to our loss



Quadratic norm penalizes outputs that are `too good'

Logistic regression, SVMs, Adaboost: more appropriate loss

Questions

Is the loss function appropriate for classification?

Quadratic loss: convex cost, closed-form solution

But does the optimized quantity indicate classifier's performance?

Is the classifier appropriate?

Linear classifier: fast computation

But could e.g. a non-linear classifier have better performance?

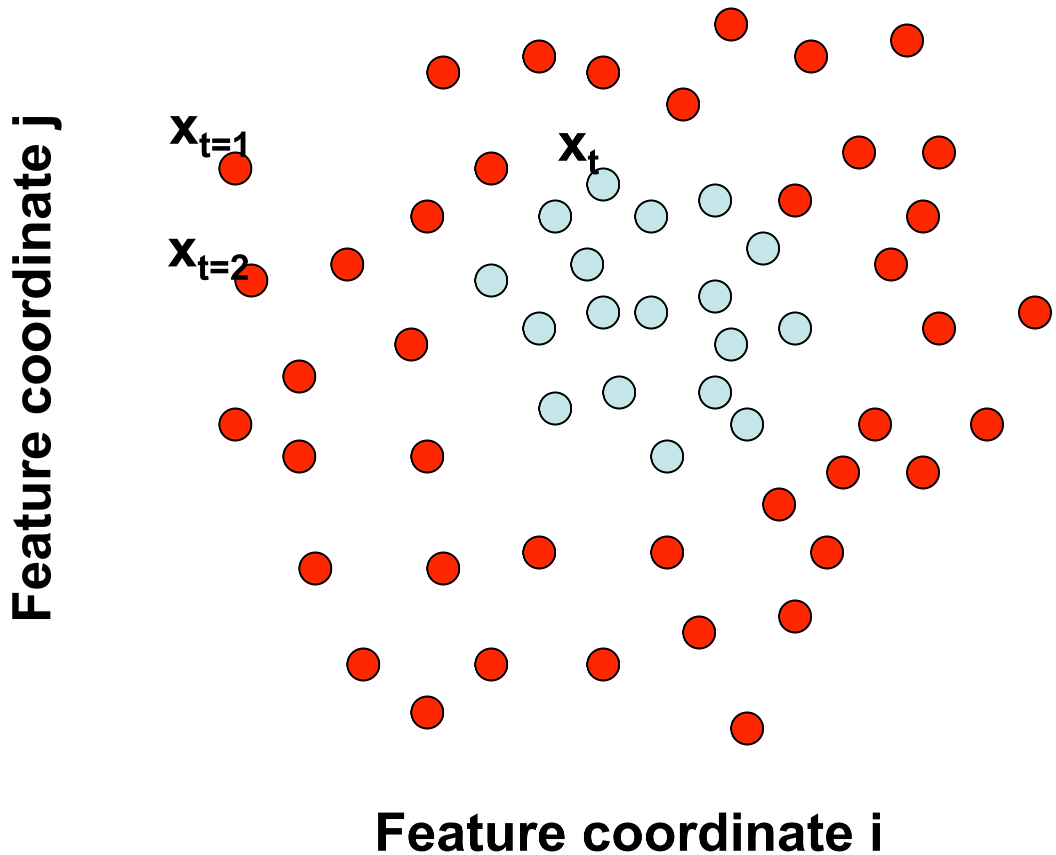
Are the estimated parameters good?

Parameters recover input-output mapping on training data

How can we know they do not simply memorize training data?

Classes may not be linearly separable

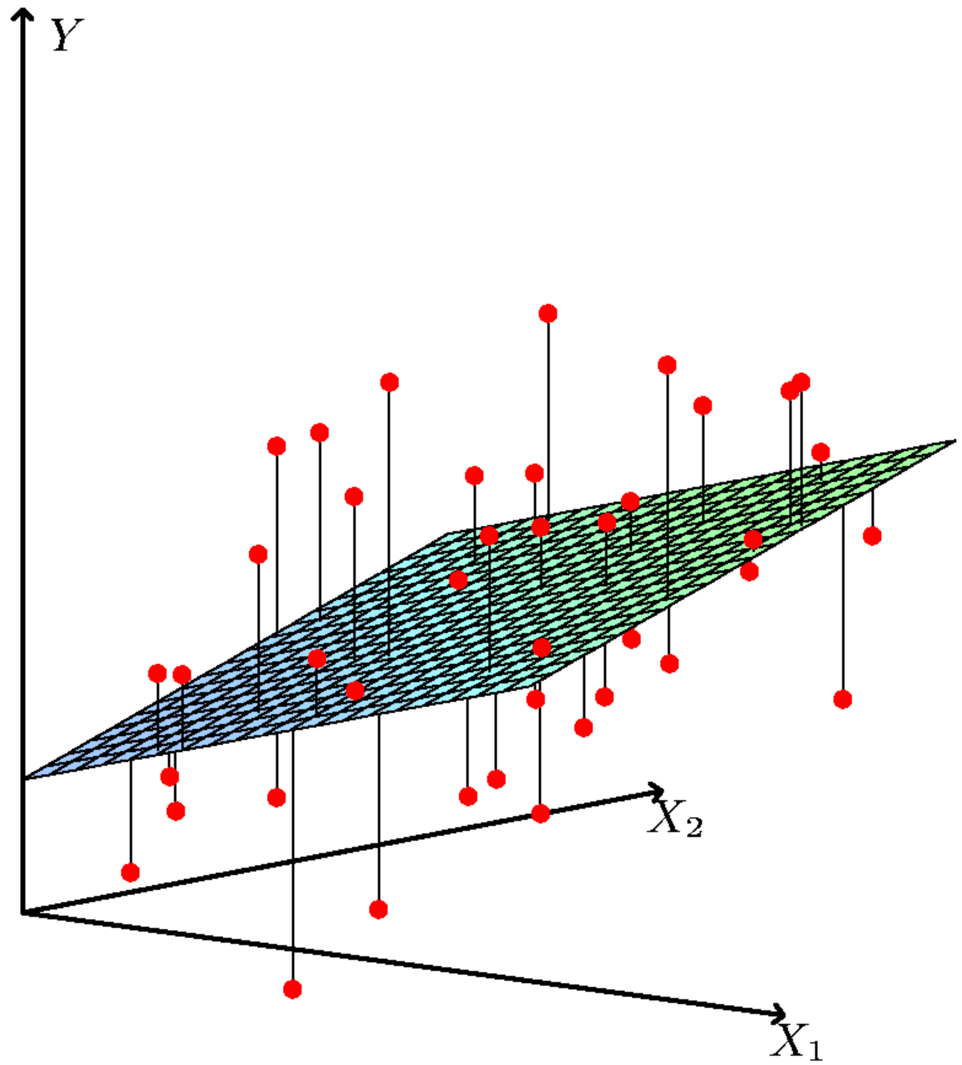
Linear function cannot properly separate these data



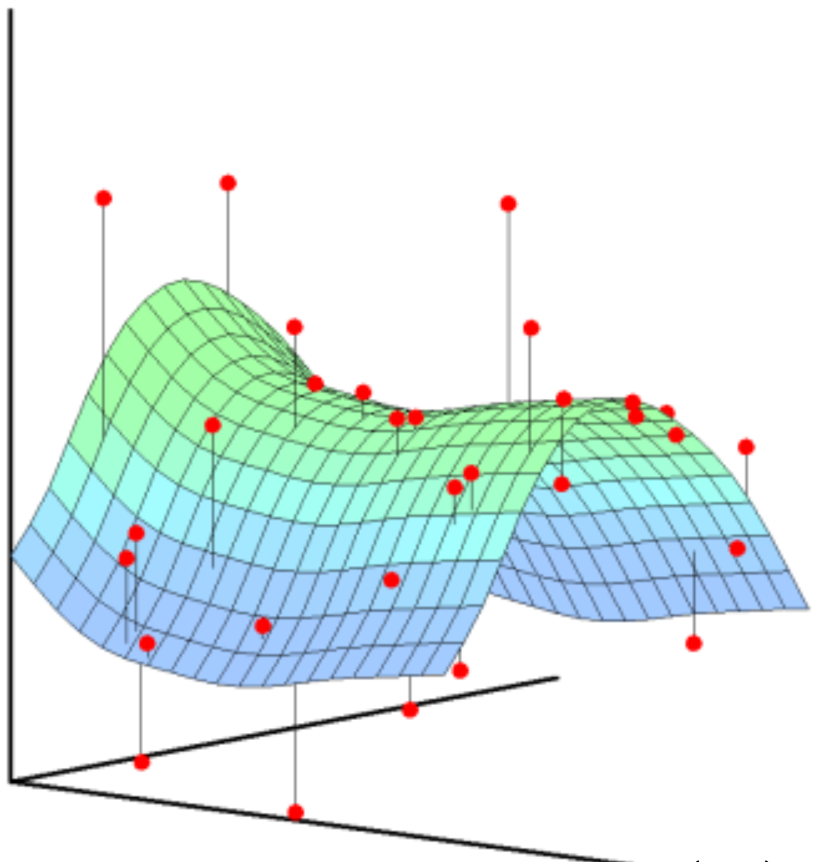
Each data point has a class label:

$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

Linear regression in 2D



Generalized linear regression



$$\mathbf{x} \to \boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}$$

Example: second-order polynomials

$$\mathbf{x} = (x_1, x_2)$$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ (x_1)^2 \\ (x_2)^2 \\ x_1 x_2 \end{bmatrix}$$

$$\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

Reminder: linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$
\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} x_0^1 & x_1^1 & \dots & x_D^1 \\ x_0^2 & x_2^2 & \dots & x_D^2 \\ \vdots & & & \\ x_0^N & x_2^N & \dots & x_D^N \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}
$$

Reminder: linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} (\mathbf{x}^1)^T \\ \hline (\mathbf{x}^2)^T \\ \hline \vdots \\ \hline (\mathbf{x}^N)^T \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

generalized linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}^i))^T = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \left[\begin{array}{c} \boldsymbol{\phi}(\mathbf{x}^1)^T \\ \hline \boldsymbol{\phi}(\mathbf{x}^2)^T \\ \hline \vdots \\ \hline \boldsymbol{\phi}(\mathbf{x}^N)^T \end{array}\right] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Nx1 **NxM** **Mx1** **Nx1**

$$\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^M$$

Least squares solution for generalized linear regression

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{\Phi} = \left[\begin{array}{c} \phi(\mathbf{x}^1)^T \\ \hline \phi(\mathbf{x}^2)^T \\ \hline \vdots \\ \hline \phi(\mathbf{x}^N)^T \end{array}\right]$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$$

Minimize (as before):

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

Questions

Is the loss function appropriate?

Quadratic loss: convex cost, closed-form solution

But does the optimized quantity indicate classifier's performance?

Is the classifier appropriate?

Linear classifier: fast computation

But could e.g. a non-linear classifier have better performance?

Are the estimated parameters good?

Parameters recover input-output mapping on training data

How can we know they do not simply memorize training data?

Example: second-order polynomials

$$\mathbf{x} = (x_1, x_2)$$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ (x_1)^2 \\ (x_2)^2 \\ x_1 x_2 \end{bmatrix}$$

$$\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

Example: fourth-order polynomials in 5 dimensions

$$\mathbf{x} = (x_1, \ldots, x_5)$$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_5 \\ \vdots \\ (x_1 x_2 x_3 x_4 x_5)^4 \end{bmatrix}$$

15625 Dimensions =>15625 parameters

What was happening before: approximations

Training: $S = \{(\mathbf{x}^i, y^i)\}, i =, 1, \ldots, N$

$$y^1 \simeq w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 \simeq w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

$$\vdots$$

$$y^N \simeq w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

If N>D (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

What is happening now: overfitting

Training: $S = \{(\mathbf{x}^i, y^i)\}, i =, 1, \ldots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

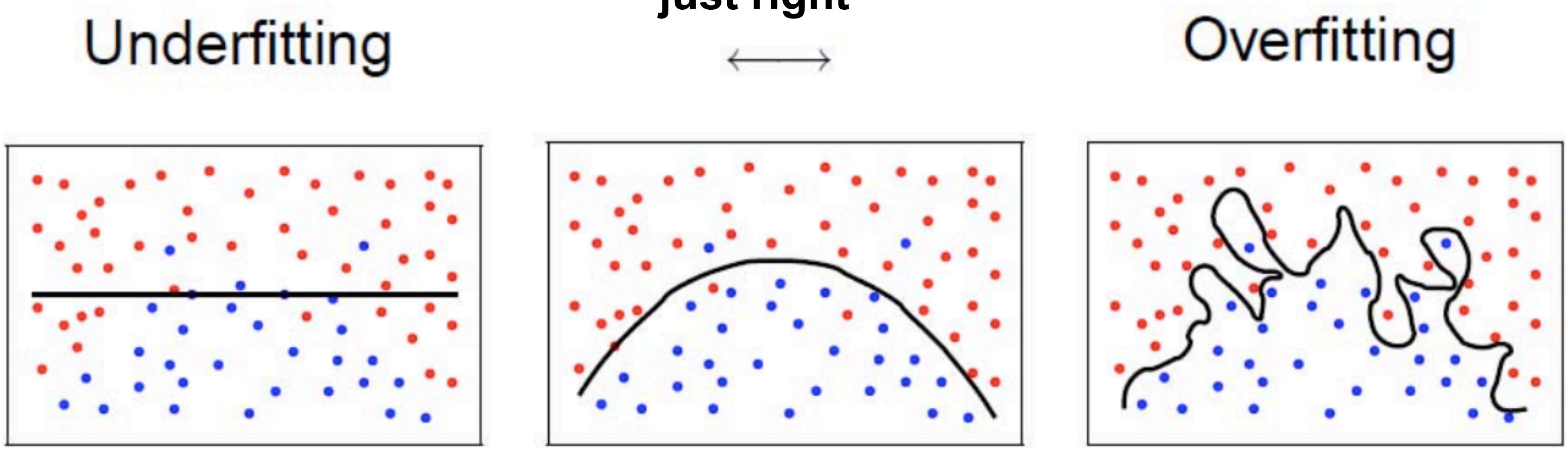
If N<D (e.g. 30 points, 15265 dimensions) we have more unknowns than equations: **underdetermined** system!

Input-output equations hold exactly, but we are simply memorizing data

Overfitting, in images

Classification

just right



Regression