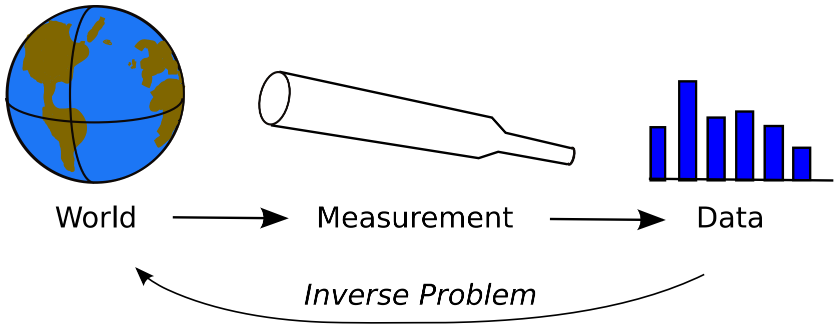
Advanced Machine Learning

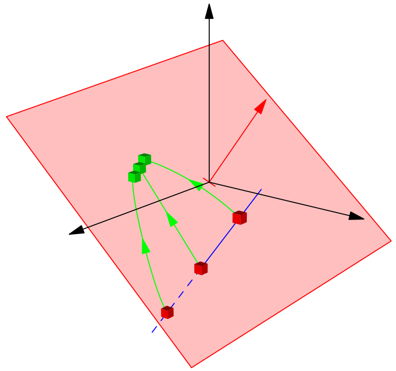
Understand Mappings



Mappings, Linear Maps, Solving Linear Systems

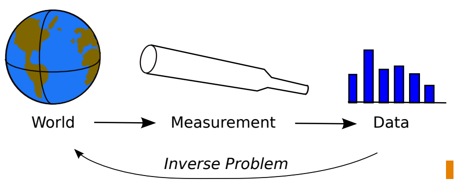
Outline

1. **Mappings**
2. Linear Maps



Transforming Data

- In the last lecture we spent time developing a sophisticate view of vector spaces and operators
- At a mathematical level machine learning can be viewed as performing an inverse mapping



- Although our mappings are not necessarily linear in either direction we learn a lot by understanding linear operators

Inverse Problems

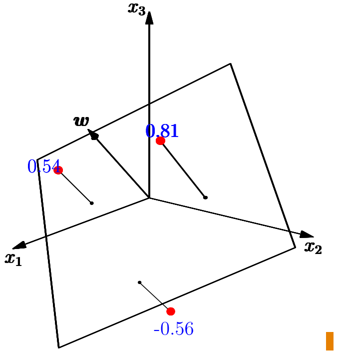
- Given m observations $\{(\boldsymbol{x}_k, y_k) | k = 1, \ldots, m\}$ and p unknown $\boldsymbol{w} = (w_1, w_2, \ldots w_p)$ such that $\boldsymbol{x}_k^\mathsf{T} \boldsymbol{w} = y_k$ then to find $\boldsymbol{w}$
- Define the *design matrix* as the matrix of feature vectors

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^\mathsf{T} \\ \boldsymbol{x}_2^\mathsf{T} \\ \cdots \\ \boldsymbol{x}_m^\mathsf{T} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mp} \end{pmatrix}$$

- and the target vector $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^\mathsf{T}$
- Then if $m = p$ we have $\boldsymbol{y} = \mathbf{X}\boldsymbol{w}$ or $\boldsymbol{w} = \mathbf{X}^{-1}\boldsymbol{y}$

Linear Regression

- $\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w}$ depends on distance from separating



- If $m > p$ then $\mathbf{X}$ isn't square so doesn't have an inverse
- Worse unless the data is accurate $\boldsymbol{y} \approx \mathbf{X}\boldsymbol{w} \Rightarrow$ no "solution"
- Problem solved by Gauss to predict the orbit of the asteroid Ceres

Linear Least Squares

- The error of input pattern $\boldsymbol{x}_k$ is

$$\epsilon_k = \boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - y_k$$

- The squared error

$$E(\boldsymbol{w}|\mathcal{D}) = \sum_{k=1}^{m} \left(\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \sum_{k=1}^{m} \epsilon_k^2 = \|\boldsymbol{\epsilon}\|^2$$

- We can define the error vector

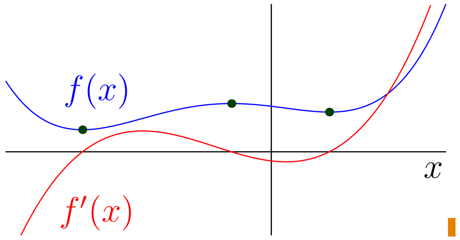
$$\boldsymbol{\epsilon} = \mathbf{X}\boldsymbol{w} - \boldsymbol{y}$$

(note that $\epsilon_k = \boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - y_k$)

- Minimising this error is known as the least squares problem

Finding a Minimum

- The minima of a one dimensional function, $f(x)$, are given by $f'(x) = 0$



- The minima of an n-dimensions function $f(\boldsymbol{x})$ are given by the set of equations

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} = 0 \quad \forall i = 1, \ldots n$$

Gradients

- The **grad** operator $\boldsymbol{\nabla}$ is the gradient operator in high dimensions

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{pmatrix}$$

- The partial derivatives (curly d's)

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i}$$

means differentiate with respect to x_i treating all other components x_j as constants

Least Squares Solution

- The least squared solution is give by

$$\begin{aligned}\nabla E(\boldsymbol{w}|\mathcal{D}) &= \nabla \|\boldsymbol{\epsilon}\|^2 = \nabla \|\mathbf{X}\boldsymbol{w} - \boldsymbol{y}\|^2 \\ &= \nabla \left(\boldsymbol{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}\right) \\ &= 2\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{w} - \mathbf{X}^{\mathsf{T}}\boldsymbol{y}\right) = 0\end{aligned}$$

- Or

$$\boldsymbol{w} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} = \mathbf{X}^{+}\boldsymbol{y}$$

- $\mathbf{X}^{+} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}$ is known as the pseudo inverse
- For non-square matrices Matlab uses the pseudo inverse so in Matlab we can write

```
w = X\y
```


Missing Bits of the Mathematics

- Note that $\|\boldsymbol{a}\|^2 = \boldsymbol{a}^{\mathsf{T}}\boldsymbol{a} = \sum_i a_i^2$

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

- Where we have used $\boldsymbol{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} = \boldsymbol{y}^{\mathsf{T}}\mathbf{X}\boldsymbol{w}$, $\sum_{i,j} w_i X_{ji} y_j = \sum_{i,j} y_i X_{ij} w_j$
- Also $\nabla \boldsymbol{w}^{\mathsf{T}}\mathbf{M}\boldsymbol{w} = \mathbf{M}\boldsymbol{w} + \mathbf{M}^{\mathsf{T}}\boldsymbol{w}$
- If $\mathbf{M} = \mathbf{M}^{\mathsf{T}}$ (i.e. $\mathbf{M}$ is symmetric) then $\nabla \boldsymbol{w}^{\mathsf{T}}\mathbf{M}\boldsymbol{w} = 2\mathbf{M}\boldsymbol{w}$
- $(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{\mathsf{T}} = \mathbf{X}^{\mathsf{T}}\mathbf{X}$ so that $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ is symmetric

Computing Gradients

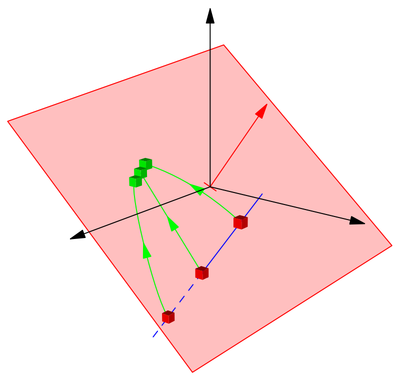
- To understand gradients we sometimes need to go back to components

$$\begin{aligned}\nabla \boldsymbol{w}^{\mathsf{T}}\mathbf{M}\boldsymbol{w} &= \begin{pmatrix}\frac{\partial}{\partial w_1} \\ \frac{\partial}{\partial w_2} \\ \frac{\partial}{\partial w_3} \\ \vdots\end{pmatrix} \sum_{i,j} w_i M_{ij} w_j = \begin{pmatrix}\sum_j M_{1j}w_j + \sum_i w_i M_{i1} \\ \sum_j M_{2j}w_j + \sum_i w_i M_{i2} \\ \sum_j M_{3j}w_j + \sum_i w_i M_{i3} \\ \vdots\end{pmatrix} \\ &= \mathbf{M}\boldsymbol{w} + \mathbf{M}^{\mathsf{T}}\boldsymbol{w}\end{aligned}$$

- It is tedious to compute these things component-wise, but when you need to understand what is going on then go back to the basics

Outline

1. Mappings
2. **Linear Maps**

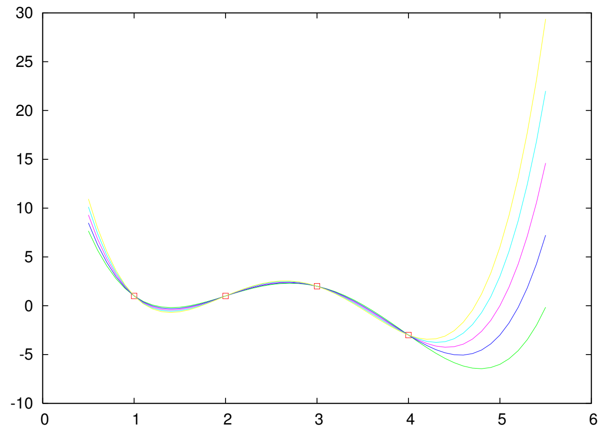


Solving Inverse Problems

- Gauss showed us how to solve **over-constrained** problems (we have more observations than parameters)
- We seek a solution which isn't necessarily exact but minimises an error
- But, what if we have more parameters than observations
- That is, we are **under-constrained**
- Note that in some directions you might be over-constrained and in other directions under-constrained
- This is very typical of most machine learning problems

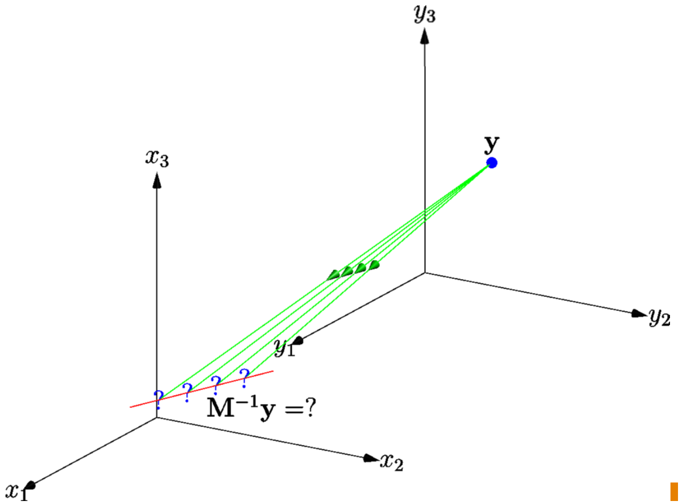
Under Constrained Systems

- If we have less data-points than parameters then there will be multiple solutions



What is the Inverse?

- Many points can map to the same points



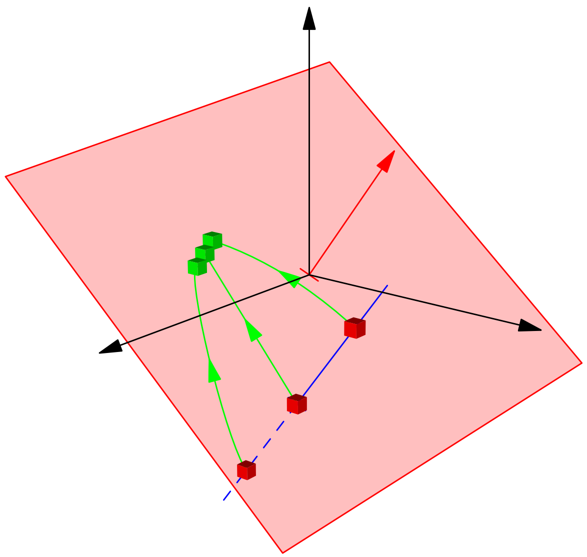
Under-constrained Systems

- The system is **under-constrained**
- We have more unknowns than equations
- The inverse is not unique
- Solving the inverse problem $\left(\boldsymbol{w} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\boldsymbol{y}\right)$ is said to be **ill-posed**
- The inverse $\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}$ doesn't exist
- If we have a complicated learning machine and not sufficient data we often end with an ill-posed inverse problem (there are lots of sets of parameters that explain the data)

Ill-Conditions

- Singular matrices are rare (although they occur when we don't have enough data), but matrices that are close to being singular are common
- If a matrix is close to singular it is ill-conditioned
- Ill-conditioned matrices have some small eigenvalues
- All points get contracted towards a plane
- Large matrices are very often ill conditioned

Ill-Conditioned Matrices



Ill-Conditioning in ML

- Ill-conditioning in machine learning occurs when a very small change in the learning data causes a large change in the predictions of the learning machine
- In linear regression the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ is ill-conditioned when we have as many data points as parameters
- Much of machine learning is concerned with making learning machines better conditioned
- Adding regularisers is one approach to achieve this

Summary

- Linear mappings are commonly used in machine learning algorithms such as regression
- We will often meet the pseudo-inverse involving inverting $\mathbf{X}^\mathsf{T}\mathbf{X}$
- They can be inherently unstable to noise in the inputs