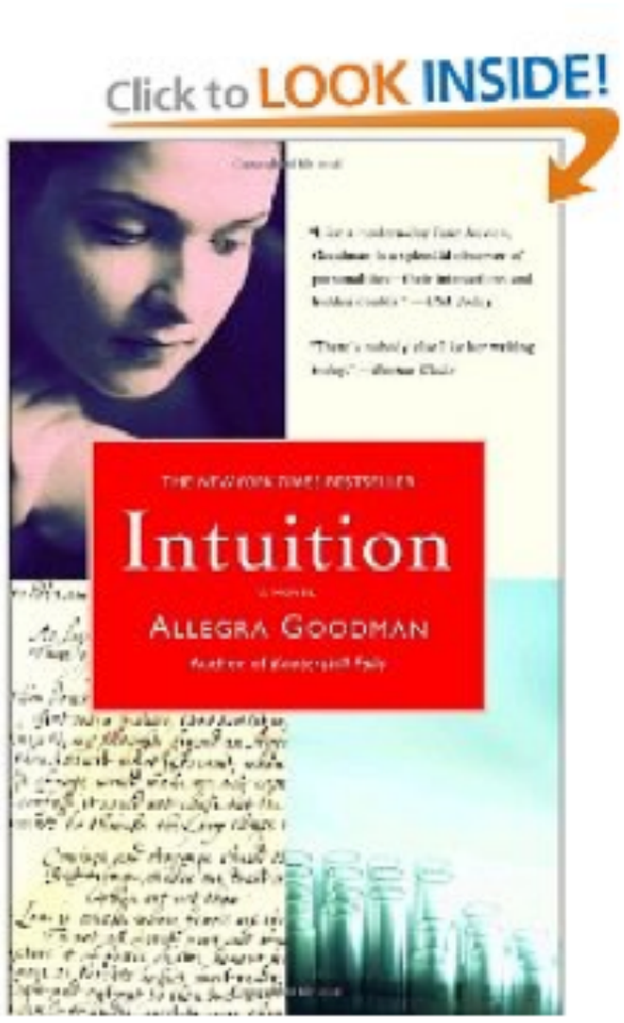
Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Regression
- Dimensionality Reduction
  - Feature selection



Recall the collaborative filtering problem, e.g. recommending books or movies based on similar users.

Intuition [Paperback]

Allegra Goodman (Author)

(69 customer reviews) | Like (4)

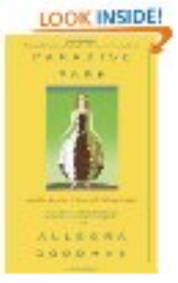
Customers Who Bought This Item Also Bought

Page 1 of 25

Kaaterskill Falls by Allegra Goodman

(62)

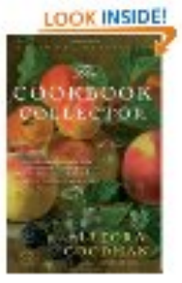
$10.40



Paradise Park by Allegra Goodman

(36)

$11.12



The Cookbook Collector: A Novel by Allegra Goodman

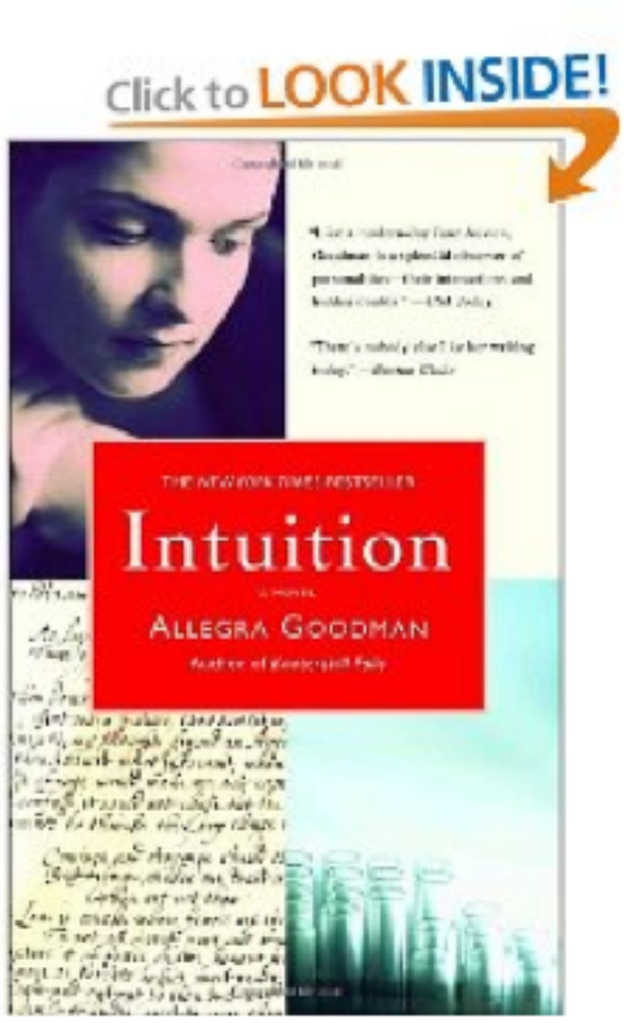
(135)

$10.20

Statistics for the Life Sciences (4th Edition) by Myra L. Samuels

(20)

$113.99



Suppose the goal (of the learning algorithm) is to predict my score, a real value between 0 and 5, on a book.

Intuition [Paperback]

Allegra Goodman (Author)

(69 customer reviews) | Like (4)

Customers Who Bought This Item Also Bought

Page 1 of 25

Kaaterskill Falls by Allegra Goodman

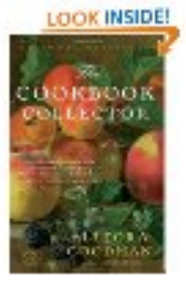
(62)

$10.40

Paradise Park by Allegra Goodman

(36)

$11.12



The Cookbook Collector: A Novel by Allegra Goodman

(135)

$10.20

Statistics for the Life Sciences (4th Edition) by Myra L. Samuels

(20)

$113.99

Nearest-neighbor regression

- Given *k*, compute a *k*-nearest neighbor data structure on the training set
- On a test example, *q*, find its *k*-nearest neighbors, *N(q)*, in the training set
- Predict:

$$\hat{y} = \frac{1}{k} \sum_{i \in N(q)} y_i$$

- Notice: this is a non-parametric method.

Regression

- We seek to learn a mapping from inputs to continuous valued outputs (e.g., price, temperature)
- A prediction problem in which the label, *y*, is from a continuous space (e.g. $\mathbb{R}$), is known as a regression problem.
- E.g. supervised learning on examples, *(x, y)*, where

$$x \in \mathbb{R}^d, y \in \mathbb{R}$$

- The goal is to learn a regressor or regression function (instead of a classifier): $f : X \to Y$

A note of clarification

- We've already seen Logistic Regression which has "regression" in its name. Note however that this is typically used to learn a classifier, (**w**, b).
  - You can also use the **w**, b parameters learned to model the distribution P(y | x) using the logistic function.
- In contrast, in regression, E[y | x] is the regressor.

Linear regression

- A very simple model for regression is called linear regression. For x $\in \mathbb{R}^d$, given parameters $\underline{\theta}$ we'll make the following modeling assumption:

$$E[y|x;\theta] = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

- We then model the conditional distribution of y given x as a 1-D Gaussian with mean $\mu = \underline{\theta} \cdot \underline{x} + \theta_0$

- Via feature mappings, $\underline{\phi}(\underline{x})$,we can (later) allow this model to represent more complex regressors.

Linear regression

We model the conditional distribution of y as a Gaussian with mean $\mu = \underline{\theta} \cdot \underline{x} + \theta_0$, and variance σ^2:

$$P(y|\mathbf{x}, \theta, \theta_0) = N(y;\, \theta^T \mathbf{x} + \theta_0,\, \sigma^2)$$

where

$$N(y;\, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right)$$

Equivalently: $y = \underline{\theta} \cdot \underline{x} + \theta_0 + \epsilon$ where:

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

Linear regression

We model the conditional distribution of y as a Gaussian:

$y = \underline{\theta} \cdot \underline{x} + \theta_0 + \epsilon$ where: $\epsilon \sim N(0, \sigma^2)$

So we model the training data as:

$$y_t = \theta^T \mathbf{x}_t + \theta_0 + \epsilon_t, \quad t = 1, \ldots, n$$

where the values $\epsilon_t \sim N(0, \sigma^2)$ are drawn independently.

Linear regression

We can then solve for the parameters to maximize the conditional likelihood of y given x, over the training data:

$$L(\theta, \theta_0, \sigma^2) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_t - \theta^T \mathbf{x}_t - \theta_0)^2\right)$$

$$= \sum_{t=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_t - \theta^T \mathbf{x}_t - \theta_0)^2\right)\right]$$

$$= \sum_{t=1}^{n} \left[-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(y_t - \theta^T \mathbf{x}_t - \theta_0)^2\right]$$

$$= \text{const.} - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{t=1}^{n}(y_t - \theta^T \mathbf{x}_t - \theta_0)^2$$

Linear regression: LMS

$$L(\theta, \theta_0, \sigma^2) = \text{const.} - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{t=1}^{n}(y_t - \theta^T\mathbf{x}_t - \theta_0)^2$$

First we solve for $\hat{\theta}, \hat{\theta}_0$ by minimizing $\sum_{t=1}^{n}(y_t - \theta^T\mathbf{x}_t - \theta_0)^2$

Then $\hat{\sigma}^2 = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{\theta}^T\mathbf{x}_t - \hat{\theta}_0)^2$

This is the (Linear) Least Mean Squares problem (a.k.a. Least Squares) and its solution is:

$$\begin{bmatrix} \hat{\theta} \\ \hat{\theta}_0 \end{bmatrix} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^T$

and X is a matrix with rows: $[\mathbf{x}_t^T, 1]$

Linear regression: Ridge Regression

- Allow feature mappings, $\underline{\phi}(\underline{x})$. Then the predicted output is given by:

$$\hat{y}(\underline{x}) = \underline{\theta} \cdot \underline{\phi}(\underline{x})$$

- Often, when the training data is small compared to dimension of the feature space, we need to regularize.
- Assuming that the noise in the observed labels is additive zero mean Gaussian, we can obtain the parameters from training samples by minimizing

$$J(\underline{\theta}) = \frac{1}{2}\sum_{t=1}^{n}\left(y_t - \underline{\theta} \cdot \underline{\phi}(\underline{x}_t)\right)^2 + \frac{\lambda}{2}\|\underline{\theta}\|^2$$

sum over the training examples

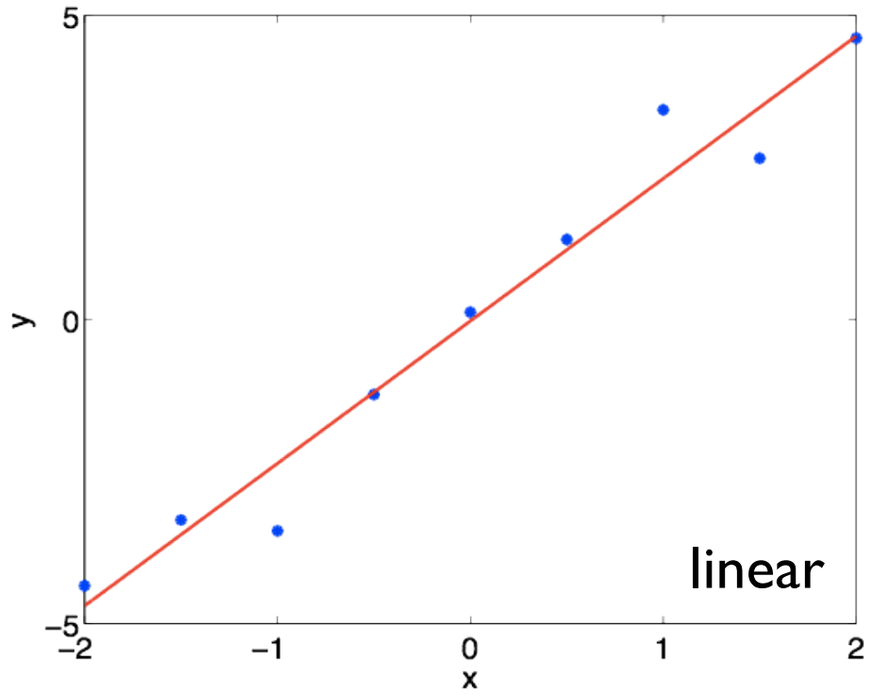
squared prediction loss on the example

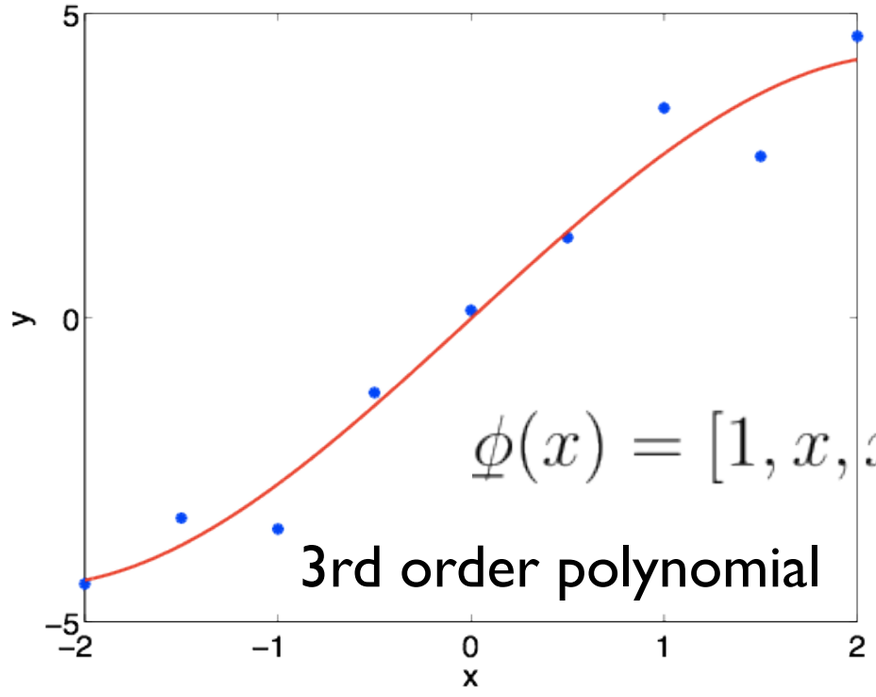
regularization term

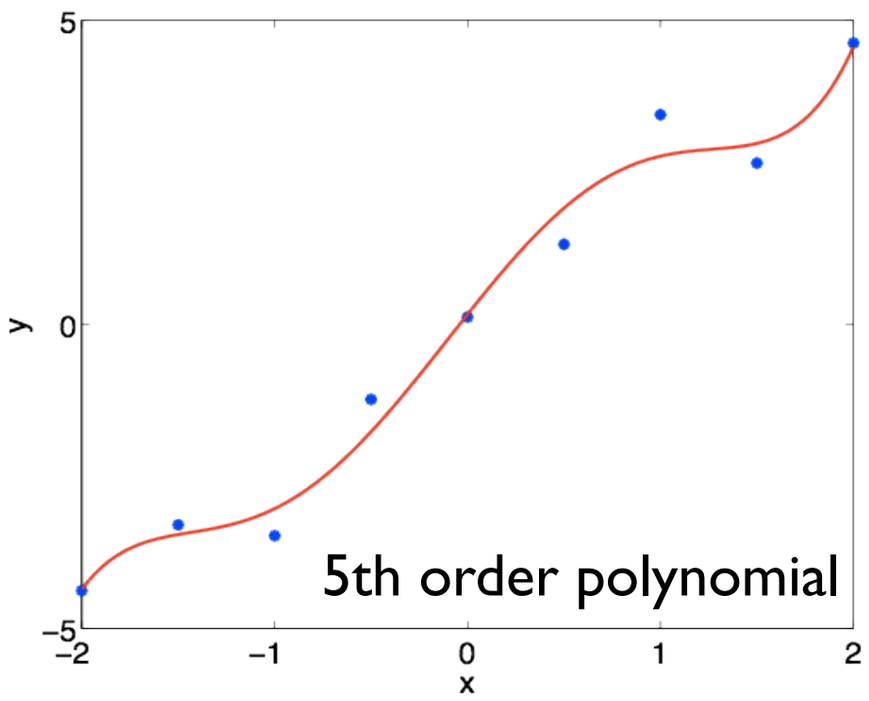
- This is Ridge Regression (a.k.a. Shrinkage). L-2 regularization term shrinks the values of the parameters, pushing unused ones to 0.

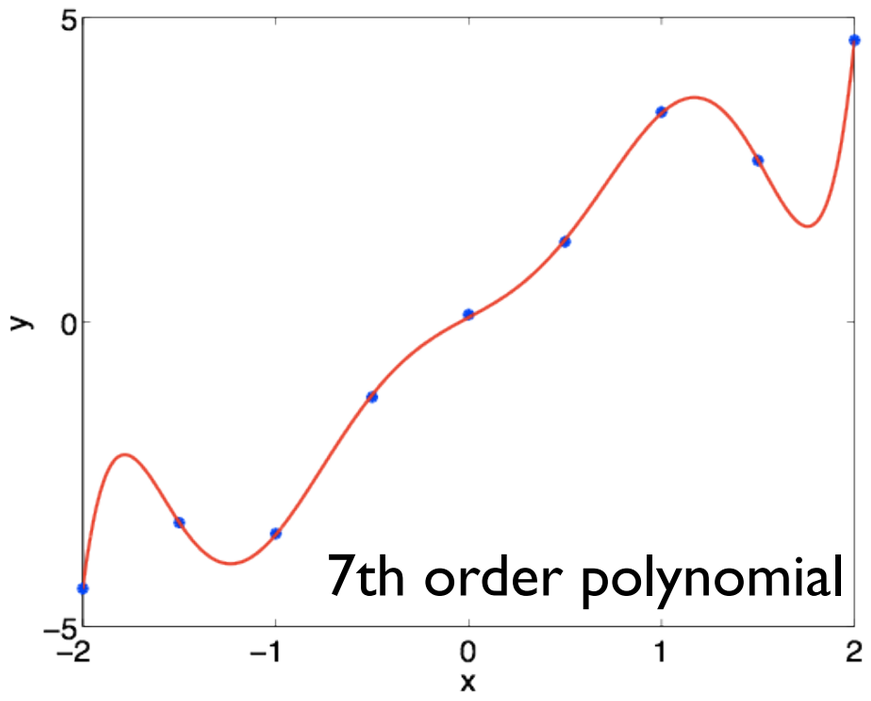
Linear regression

- We can easily obtain non-linear regression functions by considering different feature mappings









Linear regression solution

$$J(\underline{\theta}) = \frac{1}{2}\sum_{t=1}^{n}\left(y_t - \underline{\theta}\cdot\underline{\phi}(\underline{x}_t)\right)^2 + \frac{\lambda}{2}\|\underline{\theta}\|^2$$

$$\frac{d}{d\underline{\theta}}J(\underline{\theta}) = \sum_{t=1}^{n} -\overbrace{\left(y_t - \underline{\theta}\cdot\underline{\phi}(\underline{x}_t)\right)}^{\alpha_t}\underline{\phi}(\underline{x}_t) + \lambda\underline{\theta} = 0$$

$$\Rightarrow \underline{\theta}(\alpha) = \frac{1}{\lambda}\sum_{t=1}^{n}\alpha_t\underline{\phi}(\underline{x}_t)$$

- The unique solution lies in the span of the feature vectors (this is due to the regularization term), i.e. parameters θ_j for any dimensions ϕ_j that are unused in the training set, are pushed to 0 via the regularization.

Dual linear regression

- The dual parameters are obtained as the solution to a linear equation

$$\begin{aligned}\alpha_t &= y_t - \underline{\theta}(\alpha)\cdot\underline{\phi}(\underline{x}_t) \\ &= y_t - \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_i[\underline{\phi}(\underline{x}_i)\cdot\underline{\phi}(x_t)]\end{aligned}$$

kernel $K(\underline{x}_i, \underline{x}_t)$

$$\underset{n\times 1}{\underline{\alpha}} = \underset{n\times 1}{\underline{y}} - \frac{1}{\lambda}K\underline{\alpha}$$

Gram matrix $n \times n$

$$\Rightarrow \alpha^* = (I + \frac{1}{\lambda}K)^{-1}y$$

- Predicted output for a new input is given by

$$\hat{y}(\underline{x}) = \underline{\theta}(\alpha^*)\cdot\underline{\phi}(\underline{x}) = \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_i^*[\underline{\phi}(\underline{x}_i)\cdot\underline{\phi}(\underline{x})]$$

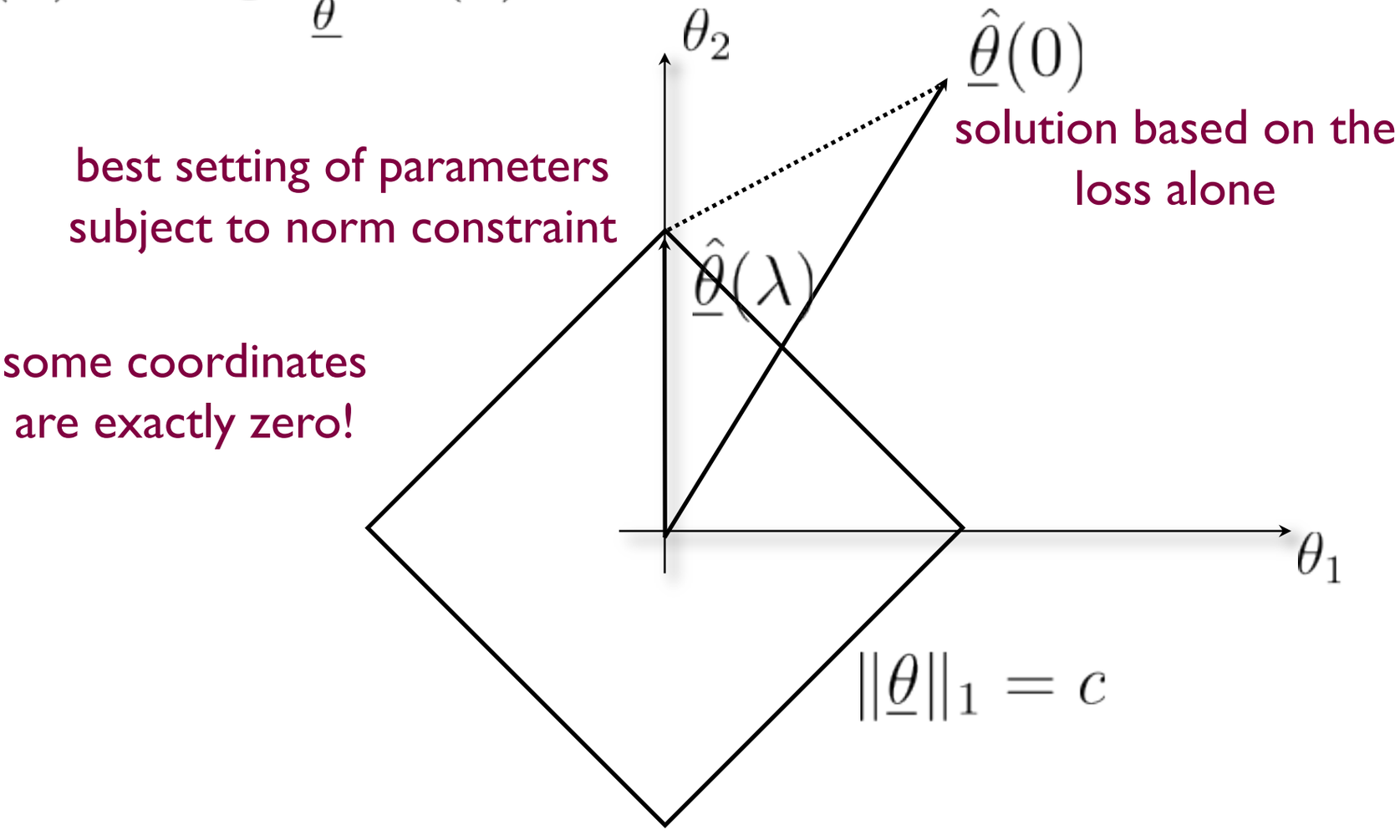
kernel $K(\underline{x}_i, \underline{x})$

LASSO: Regression with L1 regularization

- By using a 1-norm regularizer we will cause some of parameters to be set exactly to zero. → Feature selection

$$J(\underline{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_t - \underline{\theta}\cdot\underline{\phi}(\underline{x}_i)\right)^2 + \lambda\|\underline{\theta}\|_1$$

$$\hat{\underline{\theta}}(\lambda) = \underset{\underline{\theta}}{\operatorname{argmin}}\, J(\underline{\theta})$$



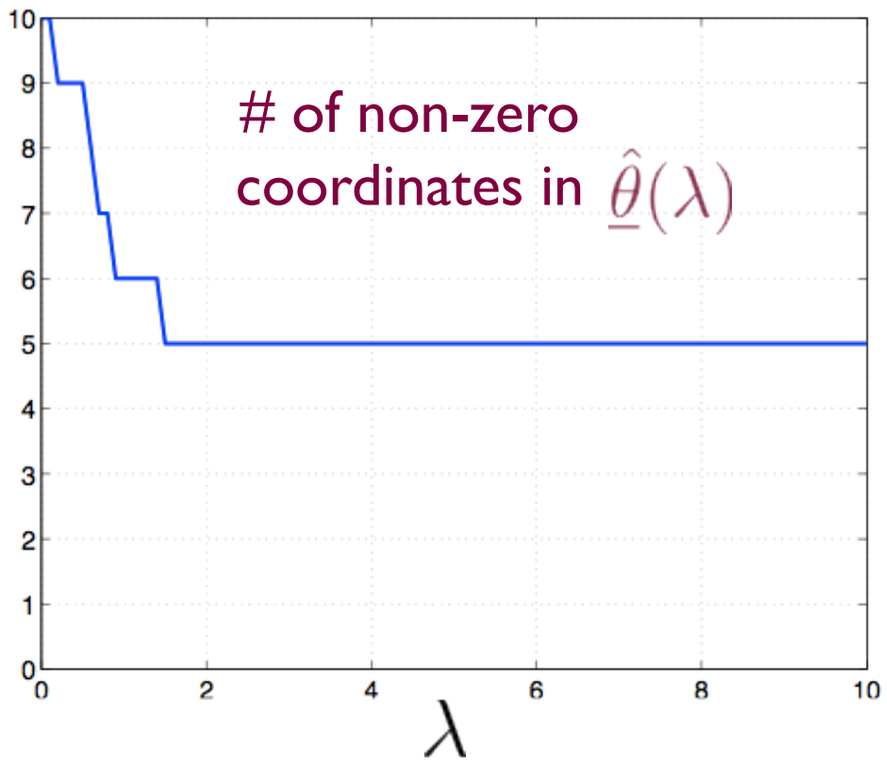
Regularization example

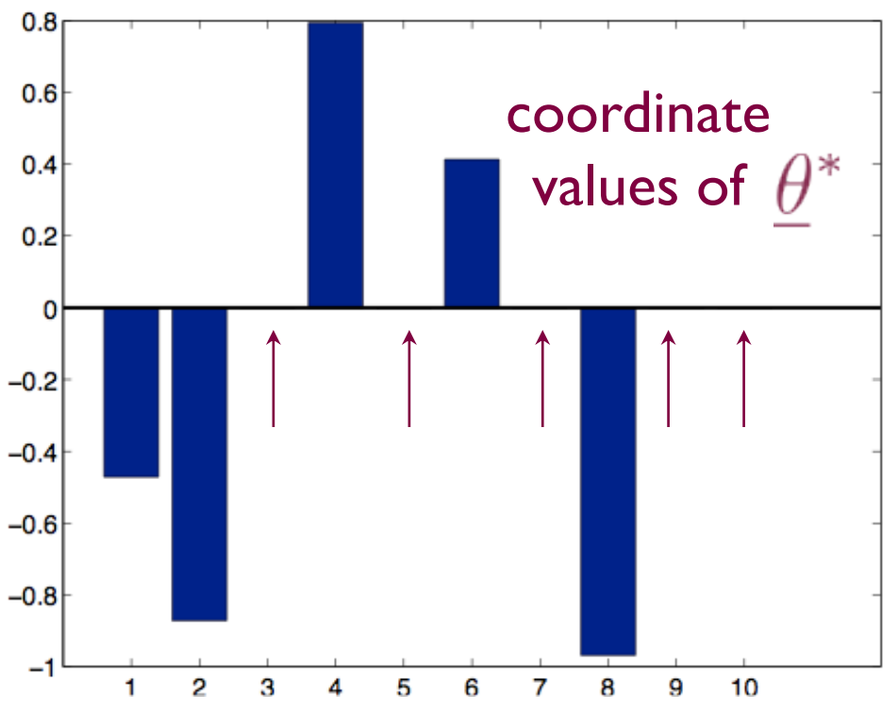
- n=100 samples, d=10, training outputs generated from

$$y_t \sim N\left(\underline{\theta}^* \cdot \underline{x}_t,\, \sigma^2\right),\ t = 1, \ldots, n$$

- If we increase the regularization penalty, λ, we get fewer non-zero parameters in the solution to

$$J(\underline{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_t - \underline{\theta} \cdot \underline{\phi}(\underline{x}_i)\right)^2 + \lambda\|\underline{\theta}\|_1$$

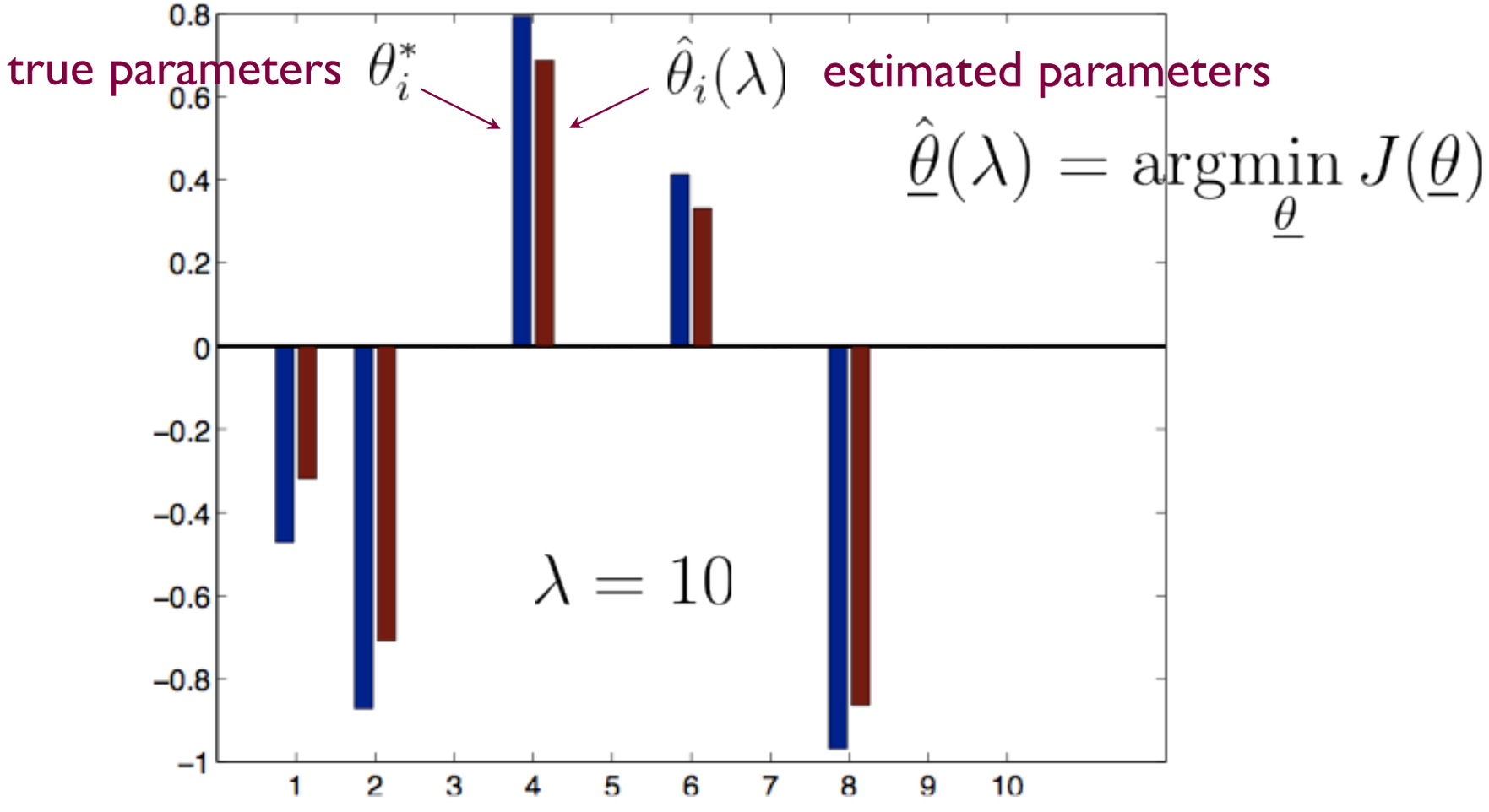




Regularization example

- The 1-norm penalty term controls the number of non-zero coordinates but also reduces the magnitude of the coordinates

$$J(\underline{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_t - \underline{\theta}\cdot\underline{\phi}(\underline{x}_i)\right)^2 + \lambda\|\underline{\theta}\|_1$$



Dimensionality reduction

- How to handle the big data setting?
- What if some of the features are redundant (e.g. deterministic functions of other features, or dependent random variables), or simply not helpful for classification?

- Dimensionality reduction is the study of how to address these issues, as well as more general issues:
  - Perhaps the relevant data lives in a lower dimensional subspace. This lower dimension is known as the intrinsic dimension of the data.
  - This could be a subset of the dimensions (or features) of the input data, or perhaps some manifold of low intrinsic dimension.
  - Meanwhile, to reduce the complexity of learning a classifier, you might want to compute in a lower dimension.

Dimensionality reduction

- One powerful technique is to take a set of random projections.

- Training:
  - Project the data set onto a random, lower dimensional subspace
  - Learn a classifier
  - Repeat multiple times (re-randomizing each time).

- Prediction:
  - On a new unlabeled example, compute its projections as above (do not re-randomize)
  - Classify under each classifier learned above.
  - Need to specify (per classification task) how to form the aggregate classification.

Dimensionality reduction

Many techniques, including:

- Random projection
- Spectral embedding
- …
- Feature selection: dimensionality reduction techniques that select a subset of the input dimensions

Feature selection

- Techniques to find a relevant subset of the dimensions or features are called Feature selection.

E.g. Linear models:

classification

$$y = \mathrm{sign}\big(\underline{\theta} \cdot \underline{\phi}(\underline{x})\big) \in \{-1, 1\}$$

regression

$$y = \underline{\theta} \cdot \underline{\phi}(\underline{x}) \in \mathcal{R}$$

etc.

$$\underline{\phi}(\underline{x}) = \begin{bmatrix} \phi_1(\underline{x}) \\ \cdots \\ \phi_d(\underline{x}) \end{bmatrix}$$

feature coordinate

- We seek to identify a few feature coordinates that the class label (or regression output) primarily depends on
- This is often advantageous in order to improve generalization (as feature selection exerts additional complexity control, or regularization) or to gain interpretability.

Simple feature selection

- There are a many approaches to dimensionality reduction. Some common feature selection techniques:
- Information analysis
  - rank individual features according to their mutual information with the class label
  - limited to discrete variables
- 1-norm regularization
  - replaces the L2-norm regularizer with the L1-norm: encourages some of the coordinates to be set exactly to zero
  - E.g. LASSO, we saw in Regression unit.
- Iterative subset selection
  - iteratively add (or prune) coordinates based on their impact on the (training) error

Information analysis

- Suppose the feature vector is just the input vector x whose coordinates take values in {1,...,k}
- Given a training set of size n, we can evaluate an empirical estimate of the mutual information between the coordinate values and the binary label

$$\hat{I}(Y, X_i) = \sum_{y \in \{-1,1\}} \sum_{x_i=1}^{k} \hat{P}(y, x_i) \log \frac{\hat{P}(y, x_i)}{\hat{P}(y)\hat{P}(x_i)}$$

empirical estimates

$$\hat{P}(y, x_i) = \frac{1}{n} \sum_{t=1}^{n} \delta(y, y_t)\delta(x_i, x_{ti})$$

$$\hat{P}(y) = \frac{1}{n} \sum_{t=1}^{n} \delta(y, y_t)$$

$$\hat{P}(x_i) = \frac{1}{n} \sum_{t=1}^{n} \delta(x_i, x_{it})$$

$$\begin{cases} 1, & \text{if } x_i = x_{it} \\ 0, & \text{otherwise} \end{cases}$$

Information analysis

- Suppose the feature vector is just the input vector x whose coordinates take values in {1,...,k}
- Given a training set of size n, we can evaluate an empirical estimate of the mutual information between the coordinate values and the binary label

$$\hat{I}(Y, X_i) = \sum_{y \in \{-1,1\}} \sum_{x_i=1}^{k} \hat{P}(y, x_i) \log \frac{\hat{P}(y, x_i)}{\hat{P}(y)\hat{P}(x_i)}$$

- Pro: provides a ranking of the features to include
- Con: weights redundant features equally (includes neither or both)
- Filter method, i.e. you can run it before doing learning, so it's not tied to the particular classifier being learned

  → However, may select features that the particular classifier cannot use, or omit combinations of features particularly useful for the classifier

Subset selection

- An algorithm for finding a good subset of feature coordinates (feature functions) to use

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

for each subset S, $|S| = k$, evaluate

$$J(S) = \min_{\underline{\theta}_S} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\theta}_S \cdot \underline{\phi}_S(\underline{x}_t)\right)^2$$

each feature subset is assessed on the basis of the resulting training error

$$\hat{S} = \operatorname*{argmin}_S J(S)$$

find the best subset

- k is used for (statistical) complexity control
- computationally hard (exponential in k)
- This is a wrapper method, as opposed to a filter method, e.g. you need to learn the classifier (or regressor) to evaluate this method.

Greedy subset selection

- A greedy algorithm for finding a good subset of feature coordinates (feature functions) to use

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

$$S = \emptyset$$

for each $j \notin S$ evaluate — try each new coordinate

$$J(S \cup j) = \min_{\underline{\theta}_{S \cup j}} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\theta}_{S \cup j} \cdot \underline{\phi}_{S \cup j}(\underline{x}_t) \right)^2$$

re-estimate all the parameters in the context of the new coordinate

$$\hat{j} = \operatorname*{argmin}_{j \notin S} J(S \cup j)$$

find the best coordinate to include

$$S \leftarrow S \cup \{\hat{j}\}$$

- each new feature is assessed in the context of those already included
- the method is not guaranteed to find the optimal subset

Forward-fitting

- We can also choose feature coordinates without re-estimating the parameters associated with already included coordinates

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

$$S = \emptyset, \quad \underline{\hat{\theta}}_\emptyset = 0$$

for each j evaluate

fixed at this stage

$$J(\underline{\hat{\theta}}_S, j) = \min_{\theta_j} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\hat{\theta}}_S \cdot \underline{\phi}_S(\underline{x}_t) - \theta_j \phi_j(\underline{x}_t) \right)^2$$

re-estimate only the parameter associated with the new coordinate

$$\hat{j} = \operatorname*{argmin}_j J(\underline{\hat{\theta}}_S, j)$$

select the best new feature

$$\underline{\hat{\theta}}_{S \cup j} = \{\underline{\hat{\theta}}_S, \hat{\theta}_{\hat{j}}\}, \quad S \leftarrow S \cup \{\hat{j}\},$$

- same feature may be included more than once