Linear models for regression

Devika Subramanian

Rice University

devika@rice.edu

Outline

Formulating linear regression

Given:

- training data $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \Re^d, y^{(i)} \in \Re\}$ drawn i.i.d. from a fixed, but unknown distribution.
- parametric function class $h_\theta : \Re^{d+1} \rightarrow \Re$

$$h_\theta(x) = \theta^T[1; x] = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d$$

    - model is linear in θ, $\theta \in \Re^{d+1}$
- Empirical loss $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))^2$

Find: $\theta^* = argmin_\theta J(\theta)$ [minimize empirical loss]

An example: predicting home prices

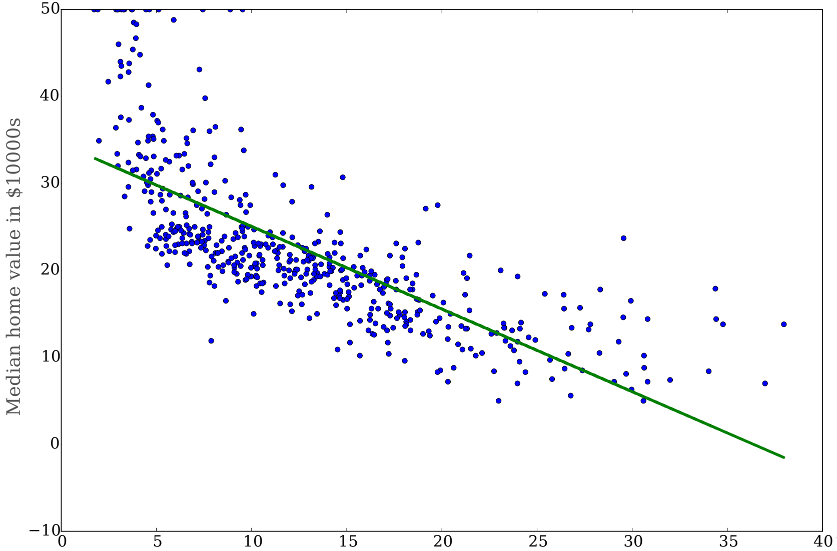
- The goal is to predict the median value of a home in the Boston suburbs based on thirteen characteristics of census tracts in the suburbs. There are 503 census tracts ($m = 503, d = 13$).

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	...	LSTAT
0.00632	18.00	2.31	0.00	0.538	6.575	65.20	4.09	...	4.98
0.02731	00.00	7.07	0.00	0.469	6.42	78.9	4.967	...	9.14
...	...	...	...	...	...	...	...	...	...

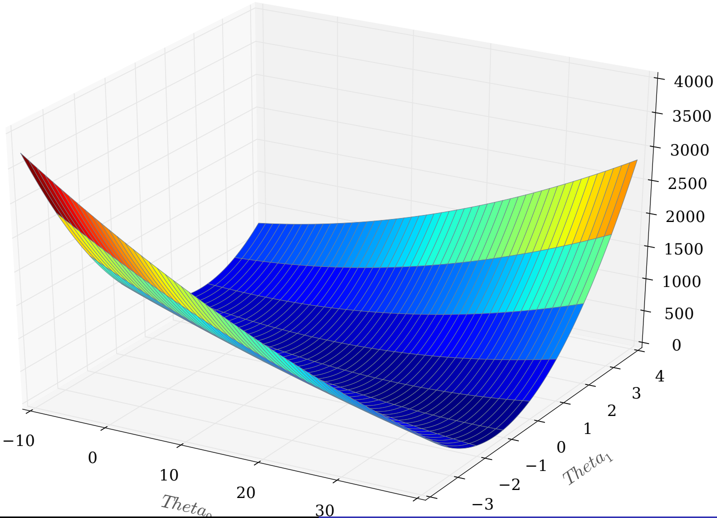
- We will first build a linear model that predicts median home value from the LSTAT (lower socioeconomic status percentage) feature alone.

$$h_\theta(LSTAT) = \theta_0 + \theta_1 LSTAT$$

Visualization of the regression problem



Visualization of the empirical loss $J(\theta)$



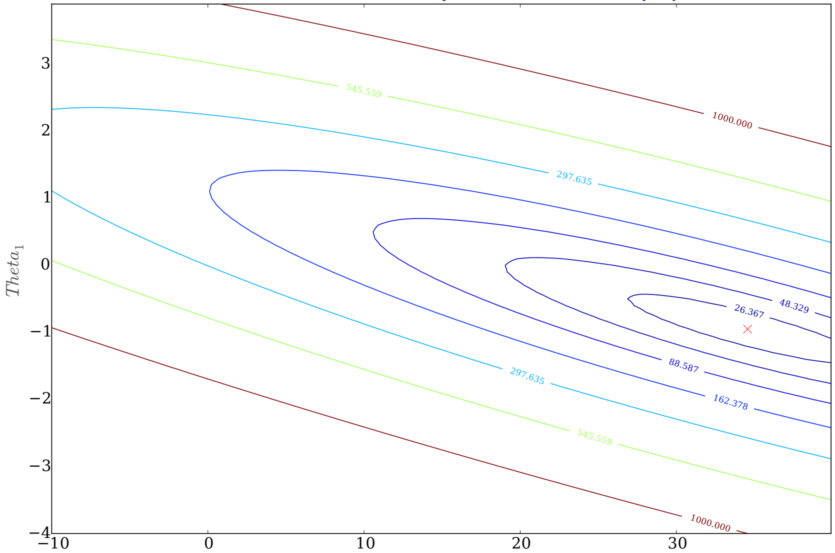
Gradient descent to estimate θ

- Empirical loss $J(\theta)$ is convex and has a unique global minimum (quadratic in θ).
- A gradient descent algorithm can find θ^* which minimizes $J(\theta)$.
    - Start with a random guess for θ (e.g., the all zeros vector).
    - Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.

$$\begin{aligned}\theta_j &\leftarrow \theta_j - \alpha\frac{\partial J}{\partial \theta_j}; \quad 0 \leq j \leq d \\ &\leftarrow \theta_j + \frac{\alpha}{m}\sum_{i=1}^{m}(y^{(i)} - \theta^T[1; x^{(i)}])x_j^{(i)}\end{aligned}$$

until θ converges. $0 < \alpha \leq 1$ is the learning rate and is typically chosen to a small number, e.g., 0.05. α needs to be chosen carefully to ensure convergence.

Another visualization of the empirical loss $J(\theta)$



Stochastic gradient descent

Standard gradient descent computes the gradient of J with respect to all examples in $\mathcal{D}$. Stochastic gradient descent (SGD) updates θ on each example. SGD causes faster convergence, especially when m is large.

- Start with a random guess for θ (e.g., the all zeros vector).
- Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.

  Loop

  for i = 1 to m do:

$$\theta_j \quad \leftarrow \quad \theta_j + \frac{\alpha}{m}(y^{(i)} - \theta^T[1; x^{(i)}])x_j^{(i)}$$

  until θ converges.

Making gradient descent work in practice

- Need to choose learning rate α very carefully.
- Need to scale the features in X so that coefficients θ_j are more or less of the same order. Example of scaling: zero mean- unit variance scaling of each column of X.

Closed-form solution for θ

$$\begin{aligned} J(\theta) &= \frac{1}{2m}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2 \\ &= \frac{1}{2m}(X\theta - y)^T(X\theta - y) \end{aligned}$$

where

$$X = \begin{bmatrix} 1 & \text{———} {x^{(1)}}^T \text{———} \\ 1 & \text{———} {x^{(2)}}^T \text{———} \\ \dots & \\ 1 & \text{———} {x^{(m)}}^T \text{———} \end{bmatrix}, y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix} \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_d \end{bmatrix}$$

By solving for $\triangledown_\theta J(\theta) = 0$, we get the normal equation

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

Vectorized form of gradient descent

- Start with a random guess for θ (e.g., the all zeros vector).
- Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.

$$\begin{aligned}\theta &\leftarrow \theta - \alpha \bigtriangledown_{\theta} J(\theta) \\ &\leftarrow \theta - \frac{\alpha}{m} X^T (X\theta - y)\end{aligned}$$

until θ converges.

This algorithm works not just for $\theta \in \Re^2$, but for general $\theta \in \Re^{d+1}$.

Probabilistic interpretation of linear regression

We will assume a *generative model* for the data:

$$y^{(i)} = \theta^{(i)} x^{(i)} + \epsilon^{(i)}, \epsilon^{(i)} \sim N(0, \sigma^2)$$

For simplicity, assume $x^{(i)}, y^{(i)} \in \Re$. Then,

$$\begin{aligned} p(\epsilon^{(i)} \mid \theta, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{{\epsilon^{(i)}}^2}{2\sigma^2}\right) \\ p((x^{(i)}, y^{(i)}) \mid \theta, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \end{aligned}$$

Maximum likelihood estimation of linear regression parameters

Given $\mathcal{D} = \{(x^{(i)}, y^{(i)}) \mid 1 \leq i \leq m, x^{(i)} \in \Re, y^{(i)} \in \Re\}$, and assuming that the data in $\mathcal{D}$ are drawn i.i.d.,

$$\begin{aligned} P(\mathcal{D} \mid \theta, \sigma^2) &= \prod_{i=1}^{m} p((x^{(i)}, y^{(i)}) \mid \theta, \sigma^2) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Principle of maximum likelihood:

$$\theta^*_{MLE} = argmax_\theta P(\mathcal{D} \mid \theta)$$

Maximum likelihood estimation of linear regression parameters (contd.)

$$\begin{aligned}\theta^*_{MLE} &= argmax_\theta logP(\mathcal{D} \mid \theta) \\ &= argmin_\theta - logP(\mathcal{D} \mid \theta)\end{aligned}$$

The quantity $-logP(\mathcal{D} \mid \theta)$ is called the negative log likelihood function (NLL).

$$\begin{aligned}logP(\mathcal{D} \mid \theta) &= log\left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{m}\left(log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)\end{aligned}$$

Maximum likelihood estimation of linear regression parameters (contd.)

$$NLL(\theta) \;\;=\;\; -logP(\mathcal{D} \mid \theta) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

Now we see that the squared error loss function arises out of maximum likelihood estimation of θ based on a linear generative model for the data.

Basis function expansion: improving expressiveness

- The goal is to predict the median value of a home in the Boston suburbs based on thirteen characteristics of census tracts in the suburbs.

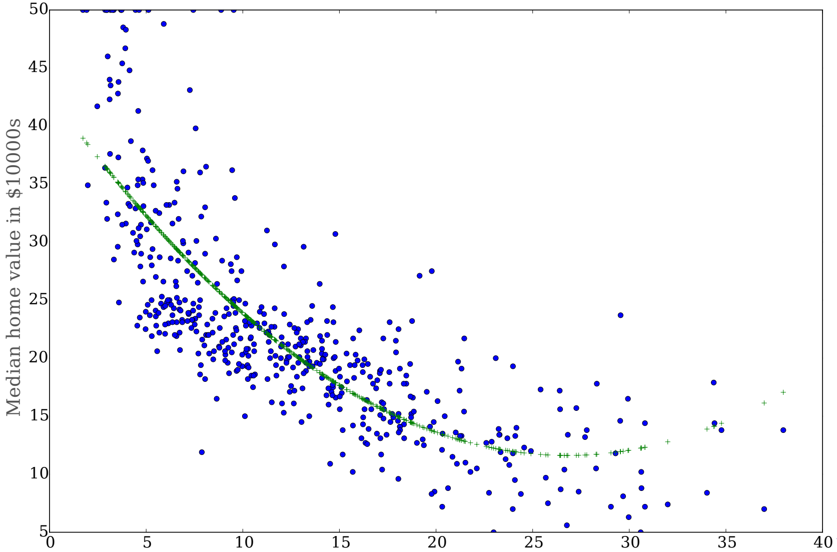
CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	...	LSTAT
0.00632	18.00	2.31	0.00	0.538	6.575	65.20	4.09	...	4.98
0.02731	00.00	7.07	0.00	0.469	6.42	78.9	4.967	...	9.14
...	...	...	...	...	...	...	...	...	...

- We will now build a model that predicts median home value from the LSTAT (lower socioeconomic status percentage) feature alone.

$$h_\theta(LSTAT) = \theta_0 + \theta_1 LSTAT + \theta_2 LSTAT^2$$

Note that while our model is non-linear in LSTAT, it is still linear in θ. Our algorithms for finding θ to minimize $J(\theta)$ can be used here! We need to scale the features $LSTAT$ and $LSTAT^2$ so that the θ components are more or less of the same order.

Quadratic basis function example



Basis function expansion

Suppose $x \in \Re$ and $y \in \Re$. We can build function classes h_θ that are all linear in θ.

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x \\ h_\theta(x) &= \theta_0 + \theta_1 x + \theta_p x^2 \\ \ldots \\ h_\theta(x) &= \theta_0 + \theta_1 x + \ldots + \theta_x x^p \end{aligned}$$

We can model basis function expansion with $\phi : \Re \rightarrow \Re^p$ which maps $x \in \Re \rightarrow \phi(x) = (x, x^2, \ldots, x^p) \in \Re^p$. In our algorithms, we replace X by $\phi(X)$.

$$X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \ldots & \ldots \\ 1 & x^{(m)} \end{bmatrix}, \phi(X) = \begin{bmatrix} 1 & x^{(1)} & {x^{(1)}}^2 & \ldots & {x^{(1)}}^p \\ 1 & x^{(2)} & {x^{(2)}}^2 & \ldots & {x^{(2)}}^p \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x^{(m)} & {x^{(m)}}^2 & \ldots & {x^{(m)}}^p \end{bmatrix}$$

Estimating θ with expanded basis functions

- Batch gradient descent

$$\theta \leftarrow \theta - \frac{\alpha}{m}\phi(X)^T(\phi(X)\theta - y)$$

- Stochastic gradient descent

$$\theta \leftarrow \theta - \frac{\alpha}{m}(\theta^T\phi(x^{(i)}) - y^{(i)})\phi(x^{(i)})$$

- Closed form solution

$$\theta^* = (\phi^T\phi)^{-1}\phi^T y$$

More on basis functions

ϕ is not limited to one dimensional x's. In fact, we can construct basis function expansions of the form $\phi : \Re^d \to \Re^p$ where $p > d$. For example

$$(x_1, x_2) \to (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

Models will be of the form

$$h_\theta((x_1, x_2)) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2$$

We can use gradient descent of θ with x replaced by $\phi(x)$.

Controlling model complexity

- Lower values of p yield stabler estimates of θ, model could potentially underfit.
- Higher values of p yield higher variance in estimates of θ, model could potentially overfit.
- One way to automatically control model complexity is to penalize large θ_j. This idea is called regularization.

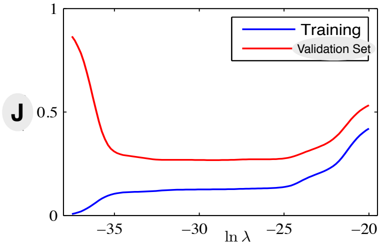
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T\phi(x^{(i)})\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{d}\theta_j^2$$

This cost function is an L2 regularizer of θ and estimates of θ which optimize this measure are called ridge regression estimates. Note that θ_0 is not penalized. λ is a regularization parameter whose value is estimated using a validation set.

Ridge regression or L2 penalties on θ

- We add a term to the cost function $\frac{\lambda}{m}\sum_{j=1}^{d}\theta_j^2$ which has the effect of driving the θ components to zero. The λ term determines the relative importance of the empirical loss term and the penalty term on θ.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T\phi(x^{(i)})\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{d}\theta_j^2$$



Ridge regression estimators

- Batch gradient descent

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} \phi(X)^T (\phi(X)\theta - y) + \frac{\lambda}{m}\theta_j \right] \quad 1 \leq j \leq d$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} \phi(X)^T (\phi(X)\theta - y) \right] \quad j = 0$$

- Stochastic gradient descent

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} (\theta^T \phi(x^{(i)}) - y^{(i)}) \phi(x^{(i)}) + \frac{\lambda}{m}\theta_j \right] \quad 1 \leq j \leq d0$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} (\theta^T \phi(x^{(i)}) - y^{(i)}) \phi(x^{(i)}) \right] \quad j = 0$$

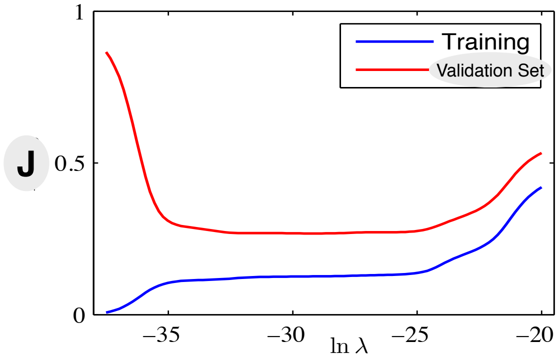
- Closed form solution

$$\theta^* = (\phi^T \phi + \lambda I)^{-1} \phi^T y$$

Regularization

Regularization allows complex models to be trained on data sets of limited size without severe overfitting, essentially by limiting effective model complexity,

Choosing regularization parameter λ



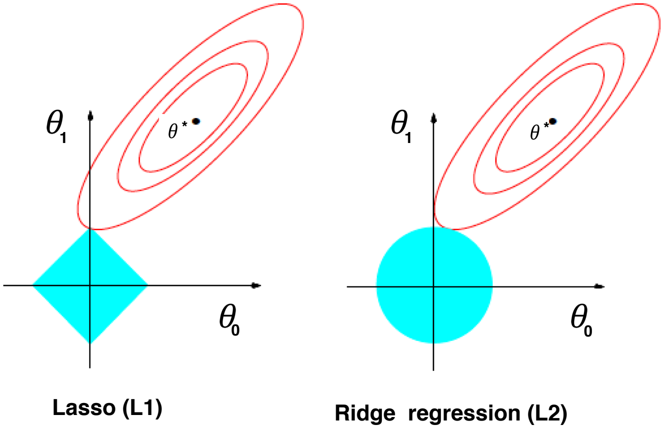
We sweep the λ parameter space on the log scale with a set-aside portion of the training data called the validation set. The training data minus the validation data is used for parameter estimation. The set-aside test data is used for final model evaluation.

Lasso regularization

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T\phi(x^{(i)})\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{d}|\theta_j|$$

- The penalty term on θ is an L1 norm, unlike the L2 norm used in ridge regression.
- Lasso regularization also drives θ components to zero, just like ridge regression. However, the L1 penalty term drives many θ components to exactly zero, especially when the regularization parameter λ is large.
- Lasso yields sparse models that only involve a subset of the features of X. Thus, lasso performs automatic feature selection.

Ridge regression and lasso



It is hard to know *a priori* which regularization technique will yield better models – so best to run both and compare using cross-validation.

Maximum A Priori (MAP) estimation of θ

$$P(\theta|\mathcal{D}) \propto P(\mathcal{D}|\theta)P(\theta)$$

- $P(\mathcal{D}|\theta)$ is the likelihood of the data $\mathcal{D}$ given the parameter θ
- $P(\theta)$ is the prior distribution on the parameter θ
- MAP estimation of θ is a Bayesian update on the prior distribution of θ using the data $\mathcal{D}$.

MAP estimation of θ

To calculate $P(\mathcal{D}|\theta)$, we assume a generative model of the data

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, \epsilon^{(i)} \sim N(0, \sigma^2)$$

Then,

$$P(\mathcal{D}|\theta; \sigma^2) \quad = \quad \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp \left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)$$

The prior distribution on θ is a multivariate Gaussian – this is a conjugate prior.

$$P(\theta) \quad = \quad N(\theta|\mathbf{0}, \alpha^2 I)$$

MAP estimation of θ (contd.)

$$\begin{aligned}
P(\theta|\mathcal{D}) &\propto P(\mathcal{D}|\theta)P(\theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \\
&\quad \left(\frac{1}{2\pi\alpha^2}\right)^{\frac{\alpha+1}{2}} exp\left(-\frac{\theta^T\theta}{2\alpha^2}\right)
\end{aligned}$$

$$\begin{aligned}
\theta_{MAP} &= argmax_\theta P(\theta|\mathcal{D}) \\
&= argmax_\theta log P(\theta|\mathcal{D})
\end{aligned}$$

Locally weighted linear regression

Consider a variation on $J(\theta)$ which weights training data points based on distance to a point x

$$J_{local}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} w^{(i)} (y^{(i)} - \theta^T \phi(x^{(i)}))^2$$

where

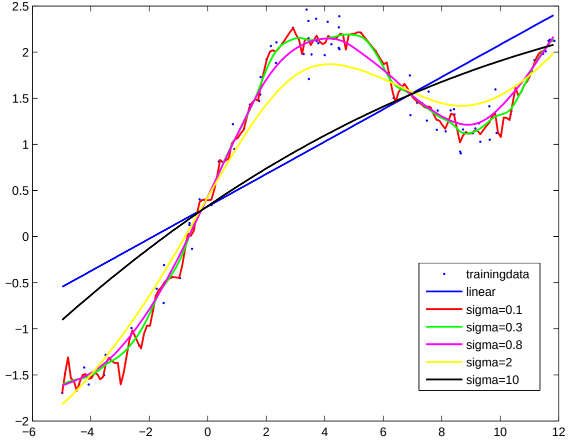
$$w^{(i)} = exp\left(-\frac{(x - x^{(i)})^T (x - x^{(i)})}{2\sigma^2} \right)$$

σ is a bandwidth parameter which controls the sphere of influence around $x^{(i)}$.

Consider the following procedure for predicting y for a new x.

- Calculate $w^{(i)}$ for $1 \leq i \leq m$.
- Solve for θ which minimizes $J_{local}(\theta)$.
- Predict $y = \theta^T x$.

Locally weighted regression



Questions

- Is locally weighted regression a parametric method or a non-parametric method?
- How can you control overfitting in locally weighted regression?
- When is it appropriate to use locally weighted regression?