Nonparametric Smoothing 2

-Quantifying the World-

Lecturer: Darren Homrighausen, PhD

Curse of dimensionality and local averaging

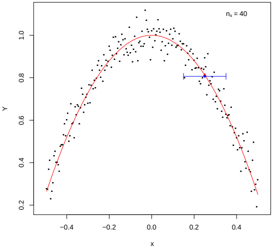
Prediction via local averaging: Loess

From the lectures, the Loess fit looks to minimize:

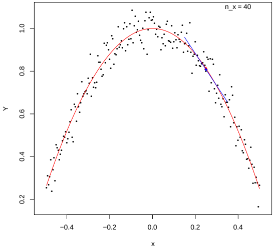
$$\sum_{i=1}^{n}(Y_i - \beta_0 - \beta X_i)^2 \mathrm{Near}(X_i, X) = \sum_{i \in N_K(X)} (Y_i - \beta_0 - \beta X_i)^2 W\left(\frac{|X - X_i|}{\Delta_X}\right)$$

and

- $N_K(X)$ are the indices of the K nearest neighbors to X
- $\Delta_X = \max_{i \in N_K(X)} |X_i - X|$, which plays the role of t

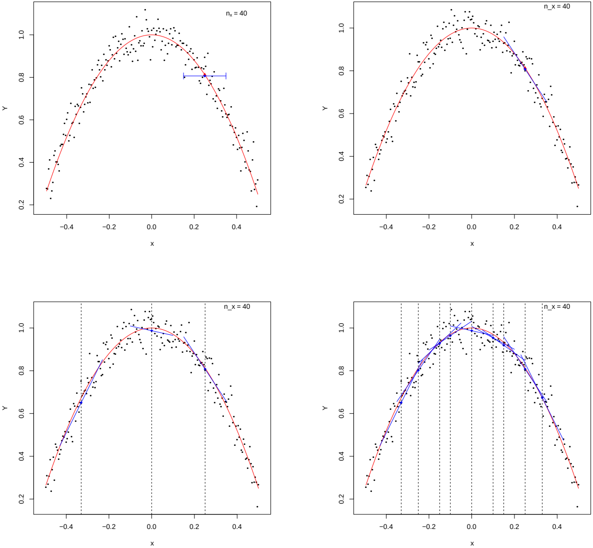


$\beta = 0$



$\beta \neq 0$

Loess



From linear to nonlinear models

Question: Why don't we always fit such a flexible model?

Answer: This works great if p is small

(and the specification of nearness is good)

However, as p gets large

- nothing is nearby
- all points are on the boundary

  (Hence, predictions are generally extrapolations)

These aspects make up (part) of the curse of dimensionality

Curse of dimensionality

Fix the dimension p

(Assume p is even to ignore unimportant digressions)

Let S be a hypersphere with radius r

Let C be a hypercube with side length $2r$

Then, the volume of S and C are, respectively

$$V_S = \frac{r^p \pi^{p/2}}{(p/2)!} \text{ and } V_C = (2r)^p$$

(Interesting observation: this means for $r < 1/2$ the volume of the hypercube goes to 0, but the diagonal length is always $\propto \sqrt{p}$. Hence, the hypercube gets quite 'spiky' and is actually horribly jagged. Regardless of radius, the hypersphere's volume goes to zero quickly.)

Curse of dimensionality

Hence, the ratio of the volumes of a circumscribed hypersphere by a hypercube is

$$\frac{V_C}{V_S} = \frac{(2r)^p \cdot (p/2)!}{r^p \pi^{p/2}} = \frac{2^p \cdot (p/2)!}{\pi^{p/2}} = \left(\frac{4}{\pi}\right)^d d!$$

where $d = p/2$

Observation: This ratio of volumes is increasing really fast. This means that all of the volume of a hypercube is near the corners. Also, this is independent of the radius.

Additive models

(ISL 7.7, 7.8.3)

Additive models

Write

$$f(X) = f_1(x_1) + \cdots + f_p(x_p) = \sum_{j=1}^{p} f_j(x_j)$$

This is

- more general than the linear fit

$$f(X) = \sum_{j=1}^{p} f_j(x_j) \underbrace{=}_{\text{multiple regression}} \sum_{j=1}^{p} \beta_j x_j$$

- less general than the fully nonparametric one

Additive models

We can find a combination of linear models and nonlinear models that provides flexibility while shielding us somewhat from the dimension problem

Estimation of such a function is not much more complicated than a fully linear model (as all inputs enter separately)

The algorithmic approach is known as backfitting

Additive models (for regression)

Additive models are fit with an iterative algorithm

(Known as the Gauss-Seidel method, an iterative scheme for solving least squares)

This is for $j = 1, \ldots, p, 1, \ldots, p, 1 \ldots$:

$$f_j(x_j) \leftarrow \mathbb{E}\left[Y - \sum_{k \neq j} f_k(x_k) | x_j \right]$$

Under fairly general conditions, this converges to $\mathbb{E}[Y|X]$

(That is the best, but unknown, prediction of Y at X)

Additive models (for regression)

Backfitting for additive models is roughly as follows:

Choose a univariate nonparametric smoother $\mathcal{S}$ and form all marginal fits $\hat{f}_j$

(Commonly a smoothing spline)

Iterate over j until convergence:

1. Define the residuals $R_i = Y_i - \sum_{k \neq j} \hat{f}_k(X_{ik})$
2. Smooth the residuals $\hat{f}_j = \mathcal{S}(R)$
3. Center $\hat{f}_j \leftarrow \hat{f}_j - n^{-1} \sum_{i=1}^n \hat{f}_j(X_{ij})$

Report

$$\hat{f}(X) = \overline{Y} + \hat{f}_1(x_1) + \cdot + \hat{f}_p(x_p)$$

Logistic regression

Logistic regression

As squared error isn't quite right for classification, additive logistic regression is a popular approach

Suppose $Y \in \{-1, 1\}$

Then, we can fit a logistic regression

$$\mathrm{logit}(\mathbb{P}(Y = 1|X)) = \log\left(\frac{\mathbb{P}(Y = 1|X)}{\mathbb{P}(Y = -1|X)}\right) = \sum_{j=1}^{p} \beta_j x_j$$

(That is, model the log odds as a linear function of the features)

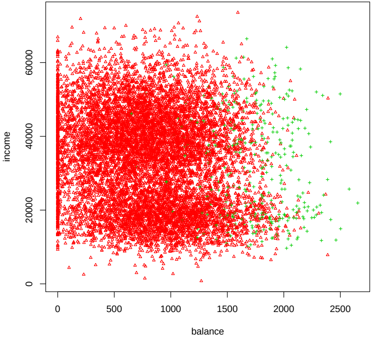
Writing $\mathbb{P}(Y = 1|X) = \pi(X)$,

$$\pi(X) = \mathbb{P}(Y = 1|X) = \frac{e^{\sum_{j=1}^{p} \beta_j x_j}}{1 + e^{\sum_{j=1}^{p} \beta_j x_j}}$$

Logistic regression: Example

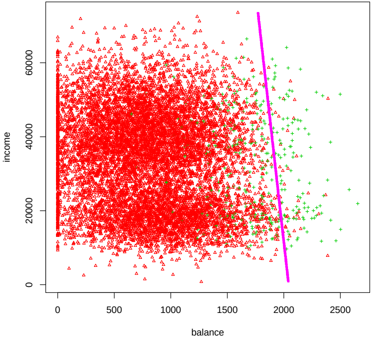
Let's look at the default data in ISL

In particular, we will look at default status as a function of balance and income



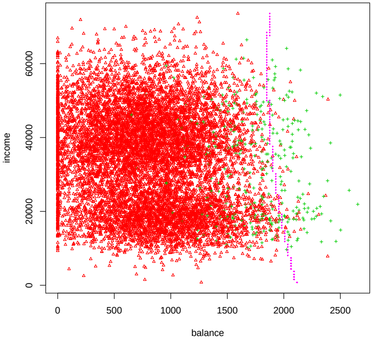
Logistic regression: transformations

```
out.glm  = glm(default~balance + income,family='binomial')
```



Logistic regression: transformations

```
out.glm  = glm(default~balance + income +
               I(income^2),family='binomial')
```



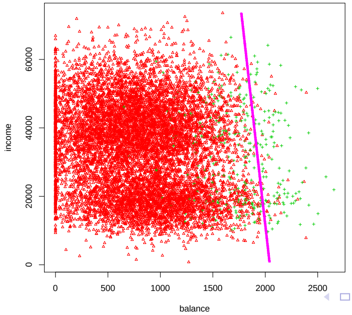
Conclusion: Linear rules in a transformed space can have nonlinear decisions in original features

Logistic regression: transformations

Reminder: The logistic model: untransformed

$$\begin{aligned}\text{logit}(\mathbb{P}(Y=1|X)) &= \beta_0 + \beta^\top X \\ &= \beta_0 + \beta_1 \text{balance} + \beta_2 \text{income}\end{aligned}$$

The decision boundary is linear in the feature space

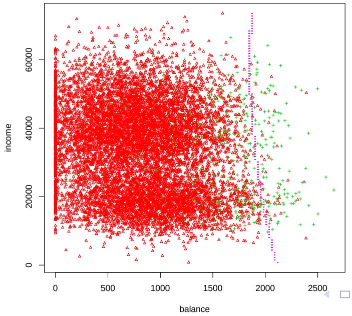


Logistic regression: transformations

Adding the polynomial transformation $\Phi(X) = (x_1, x_2, x_2^2)$:

$$\begin{aligned}\text{logit}(\mathbb{P}(Y = 1|X)) &= \beta_0 + \beta^\top \Phi(X) \\ &= \beta_0 + \beta_1 \text{balance} + \beta_2 \text{income} + \beta_3 \text{income}^2\end{aligned}$$

Decision boundary is nonlinear in the feature space!



Additive logistic regression

Additive models (for classification)

This gets inverted in the usual way to acquire a probability estimate

$$\pi(X) = \mathbb{P}(Y = 1|X) = \frac{e^{f(X)}}{1 + e^{f(X)}}$$

($f(X) = X^\top \beta$ gives us (linear) logistic regression)

These models are usually fit by numerically maximizing the binomial likelihood, and hence enjoy all the asymptotic optimality features of MLEs

Additive models (for classification)

Example: In R, this can be fit with the package gam

In the gam package there is a dataset kyphosis

This dataset examines a disorder of the spine

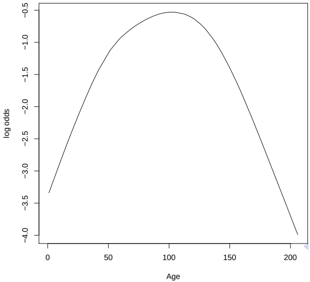
Let's look at two possible covariates Age and Number

(Number refers to the number of vertebrae that were involved in a surgery)

Additive models (for classification)

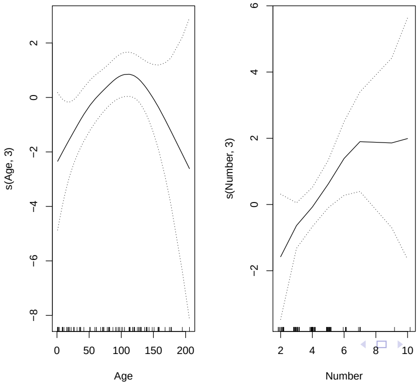
```
library(gam)
data(kyphosis)

out = gam(Kyphosis~s(Age,3),family=binomial,data=kyphosis)
out.pred = predict(out)
plot(sort(kyphosis$Age),out.pred[order(kyphosis$Age)],
      type='l',xlab='Age',ylab='log odds')
```



Additive models (for classification)

```
out = gam(Kyphosis ~ s(Age,3) + s(Number,3),
        family = binomial, data=kyphosis)
par(mfrow=c(1,2))
plot(out,se=T)
```



More complex additive models

More generally, we can consider each function in the sum to be a function of all input variables

A prominent approach here is called boosting

The core idea is that boosting is a greedy approach to fitting the more complex additive model

Unlike gam, boosting fits an additive model using a base learner for ϕ

(Often, trees are used as a base learner, though many procedure can be boosted)

Class Exercise

Class exercise

1. Take the python code `unit8inClass.py` and finish the backfitting implementation
2. Using `kyphosis_gam.R`, fit an additive model and get the predicted probability of Kyphosis for someone at Age $=$ 10 and Number $=$ 4