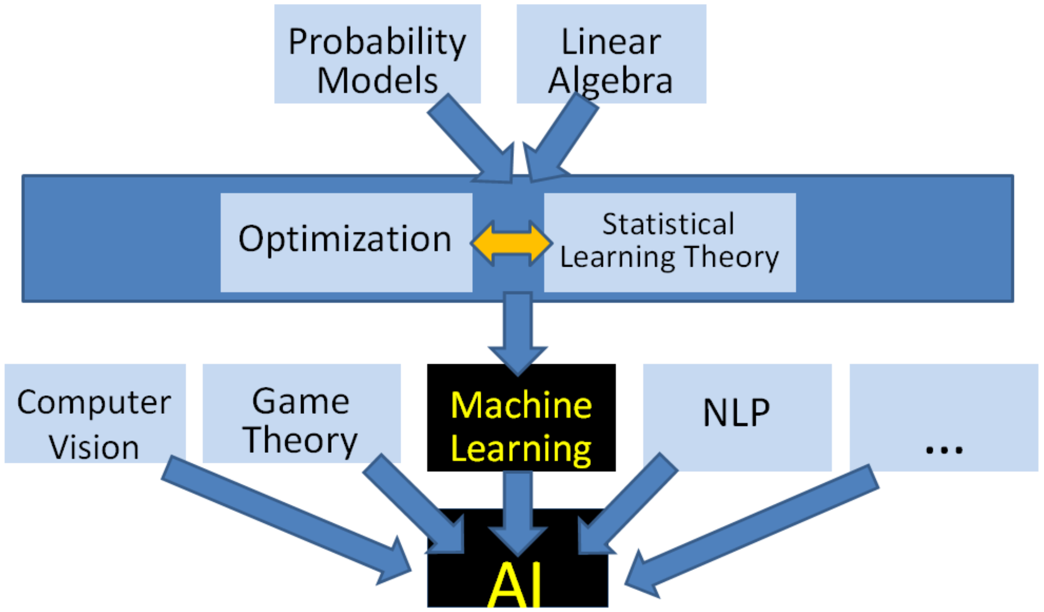
Practical Machine Learning

Non-Linear Models, Naïve Bayes Classification and Logistic Regression



Dr. Nathan Bastian
Northwestern University

Non-Linear Models in Machine Learning

Moving Beyond Linearity

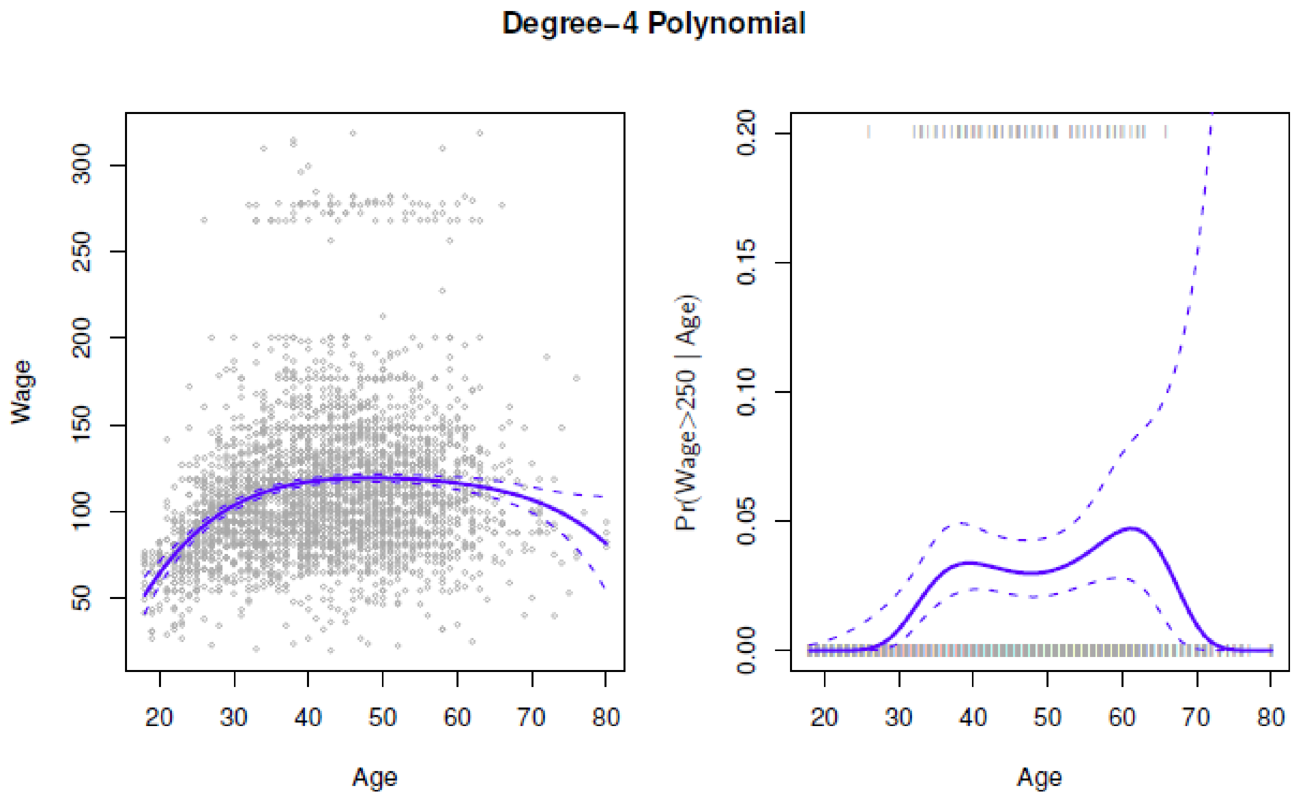
- The linearity assumption is good in many machine learning problems.

- However, there are other methods that offer a lot of flexibility, without losing the ease and interpretability of linear models:
  - Polynomial regression
  - Step functions
  - Regression and smoothing splines
  - Local regression
  - Generalized additive models (GAMs)

Polynomial Regression

- Replace the standard linear model with a polynomial function:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$



Polynomial Regression (cont.)

- For large enough degree *d*, a polynomial regression allows us to produce an extremely non-linear curve.
- We do this by creating new variables $X_1 = X$, $X_2 = X^2$, etc. and then treat as multiple OLS linear regression.
- In general, we are not really interested in the coefficients, but instead the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4$$

Polynomial Regression (cont.)

- It is unusual to use *d* greater than 3 or 4 because for large values of *d*, the polynomial curve can become overly flexible and can take on very strange shapes.

- Note that we can also use cross-validation to choose *d*.

Step Functions

- Using polynomial functions of the features as predictors in a linear model imposes a *global* structure on the non-linear function of X.

- To avoid imposing such a global structure, we can create transformations of a variable by cutting the variable into distinct regions.

- In particular, we use *step functions* to break the range of *X* into bins, where we fit a different constant in each bin.

Step Functions (cont.)

- This amounts to converting a continuous variable into an *ordered categorical variable*.
- In greater detail, we create cutpoints (or knots) c_1, c_2,….,c_K in the range of X and then construct K + 1 new variables:

$$\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X < c_2), \\
C_2(X) &= I(c_2 \leq X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}$$

where I(.) is an *indicator function* that returns a 1 if the condition is true and 0 otherwise.

Step Functions (cont.)

- Note that for any value of *X*, $C_0(X) + C_1(X) + \ldots + C_K(X) = 1$, since *X* must be exactly in one of the *K* + 1 intervals.

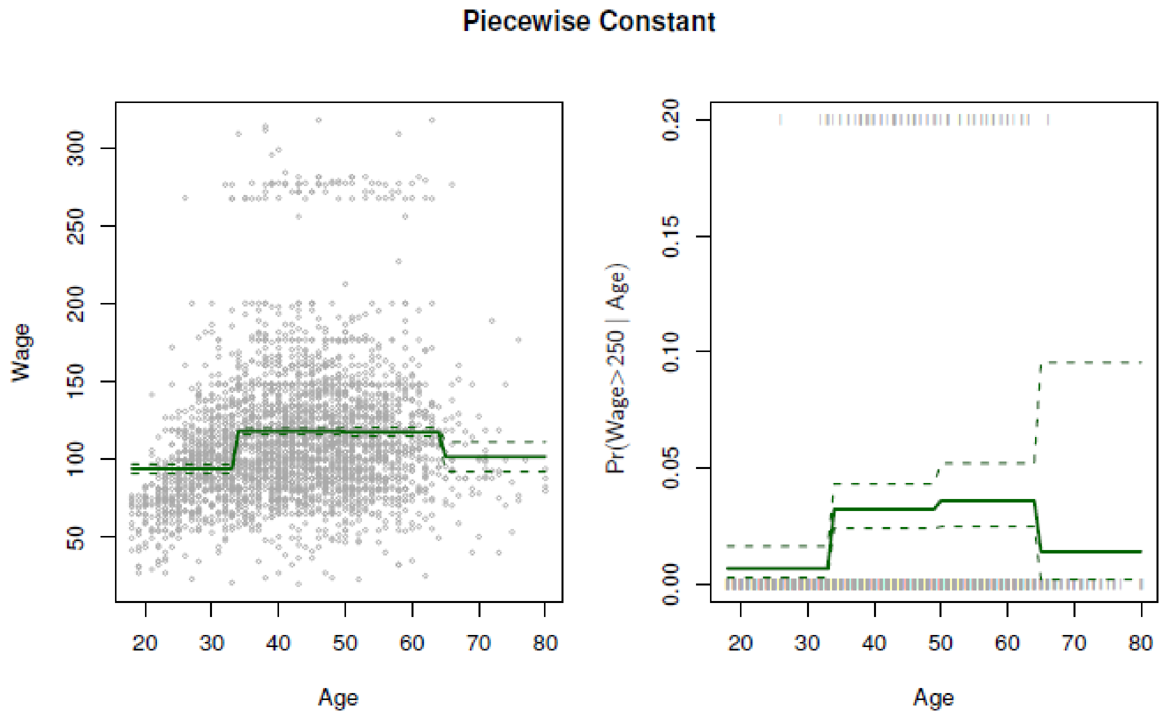
- We then use OLS estimation to fit a linear model using these *K* + 1 new variables.

- For a given value of *X*, at most one of $C_1, C_2, \ldots, C_K$ can be non-zero.

- β_j represents the average increase in the response for *X* in $c_j \leq X \leq c_{j+1}$ relative to $X < c_1$.

Step Functions (cont.)

$$C_1(X) = I(X < 35), \quad C_2(X) = I(35 \leq X < 50), \ldots, C_3(X) = I(X \geq 65)$$



- Unless there are natural breakpoints in the predictors, piece-wise constant functions can miss the action.

Basis Functions

- Polynomial and piece-wise constant regression models are special cases of a *basis function* approach.

- The idea is to have at hand a family of functions or transformations that can be applied to a variable X: $b_1(X),\ldots.,b_K(X)$

- Instead of fitting a linear model in X, we fit the following model:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \ldots + \beta_K b_K(x_i) + \epsilon_i$$

- Note that the basis functions $b_1(.),\ldots.,b_K(.)$ are fixed and known.

Basis Functions (cont.)

- For polynomial regression, the basis functions are $b_j(x_i) = x_i^j$

- For piece-wise constant functions, the basis functions are

$$b_j(x_i) = I(c_j \leq x_i \leq c_{j+1})$$

- Note that we can use OLS to estimate the unknown regression coefficients.

- Thus, all of the inference tools for linear models (standard errors, F-statistics, etc.) are available in this setting.

Regression Splines

- Regression splines are a flexible class of basis functions that extend upon the polynomial regressions and piece-wise constant regression approaches.
- They involve dividing the range of *X* into *K* distinct regions; within each region, a polynomial function is fit to the data.
- These polynomials are constrained so that they join *smoothly* at the region boundaries (or *knots*).
- Provided that the interval is divided into enough regions, this can produce an extremely flexible it.

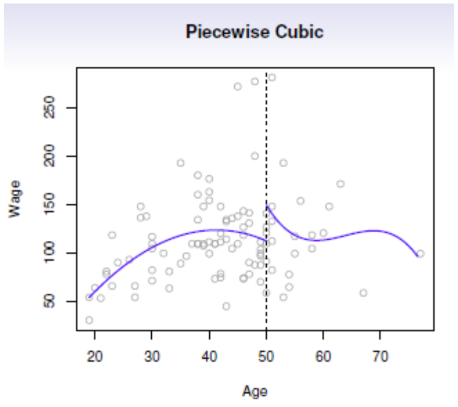
Piecewise Polynomials

- Instead of fitting a high-degree polynomial over the entire range of *X*, *piece-wise polynomial regression* involves fitting separate low-degree polynomials over different regions of *X*.

- Here, the beta coefficients differ in different parts of the range of *X*; the points where the coefficients change are called *knots*.

- Example: A piecewise cubic polynomial with a single knot at a point *c* takes the following form:

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$



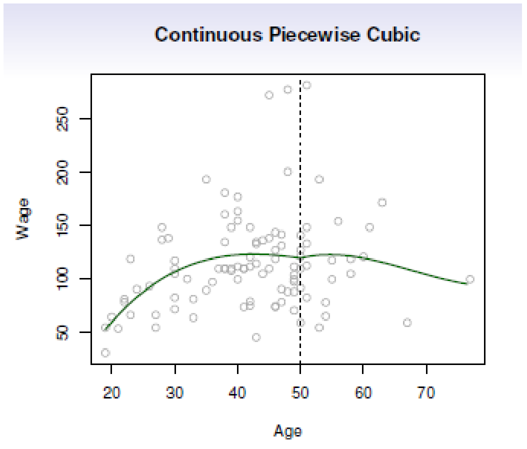
Piecewise Polynomials (cont.)

- Each of the polynomial functions can be fit using OLS applied to simple functions of the original predictor.

- Using more knots leads to a more flexible piecewise polynomial.

- If general, if we place *K* different knots through the range of *X*, then we end up fitting *K* + 1 different polynomials.

- It is better to add *constraints* to the polynomials (e.g., continuity). *Splines* have the maximum amount of continuity.

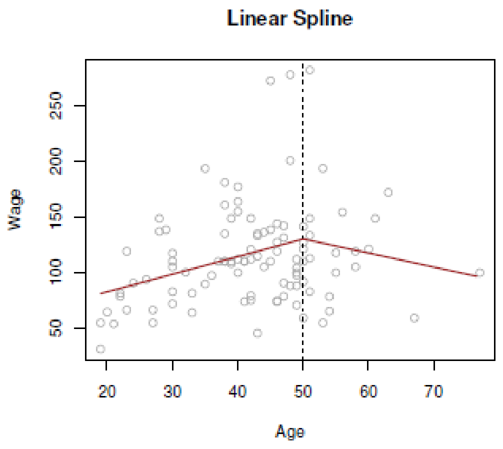


Regression Splines (cont.)

- Each constraint that we impose effectively frees up one degree of freedom, by reducing the complexity of the resulting piecewise polynomial fit.

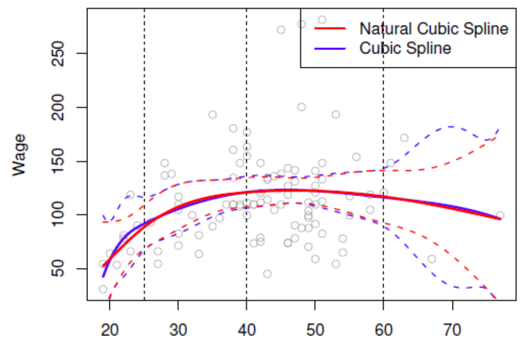
- The general definition of a degree-*d* spline is that it is a piecewise degree-*d* polynomial, with continuity in derivatives up to degree *d* – 1 at each knot.

- Thus, a **linear spline** is obtained by fitting a line in each region of the predictor space defined by the knots, requiring continuity at each knot.



Regression Splines (cont.)

- A *natural spline* is a regression spline with additional boundary constraints.
- The function is required to be linear at the boundary (in the region where *X* is smaller than the smallest knot, or larger than the largest knot).
- This additional constraint means that natural splines generally produce more stable estimates at the boundaries.
- A *natural cubic spline* extrapolates linearly beyond the knots.



Regression Splines (cont.)

Choosing the Location of Knots

- The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly.
- One option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable.
- In practice, it is common to place knots in a uniform fashion. For example, one strategy is to decide *K*, the number of knots, and then place them at appropriate quantiles of the observed *X*.

Regression Splines (cont.)

Choosing the Number of Knots

- One option is to try out different numbers of knots and see which produces the best looking curve.
- However, a more objective approach is to use cross-validation.
- The procedure is repeated for different number of knots *K*; then the value of *K* giving the smallest RSS is chosen.
- Splines allow us to place more knots, and hence flexibility, over regions where the function *f* seems to be changing rapidly, and fewer knots where *f* appears more stable.

Nonparametric Regression

- The nonparametric regression approach is to choose a function f from some smooth family of functions.
- We do need to make some assumptions about f – that it has some degree of smoothness and continuity. However, these restrictions are far less limiting than the parametric way.
- Unlike parametric models, nonparametric models do not have a formulaic way of describing the relationship between the predictors and the response; this often needs to be done graphically.
- However, the nonparametric approach is more flexible, assumes far less (and so is less liable to make bad mistakes), and is particularly useful when little past experience is available to know the appropriate form for a parametric model.

Smoothing Splines

- We create regression splines by specifying a set of knots, producing a sequence of basis functions, and then use OLS to estimate the spline coefficients.

- What we really want is a function *g* that makes RSS small and *smooth*. Thus, consider the following criterion for fitting a smooth function *g*(*x*) to some data (known as a *smoothing spline*):

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

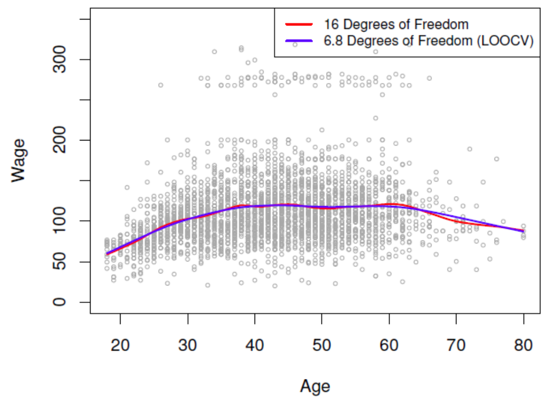
where λ is a nonnegative tuning parameter.

Smoothing Splines (cont.)

- The first term is a loss function (RSS), which tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is; this is modulated by the tuning parameter λ.
- The larger the value of λ, the smoother g will be. The smaller the value of λ, the more wiggly the function.
- As $\lambda \rightarrow \infty$, the function $g(x)$ becomes linear.

Smoothing Splines (cont.)

- It turns out that the solution is a natural cubic spline, with a knot at every unique value of x_i.
- The tuning parameter λ controls the level of roughness (i.e., the effective degrees of freedom).
- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- We use LOOCV to find λ!!



Local Regression

- *Local regression* is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point x_0 using only the nearby training observations.

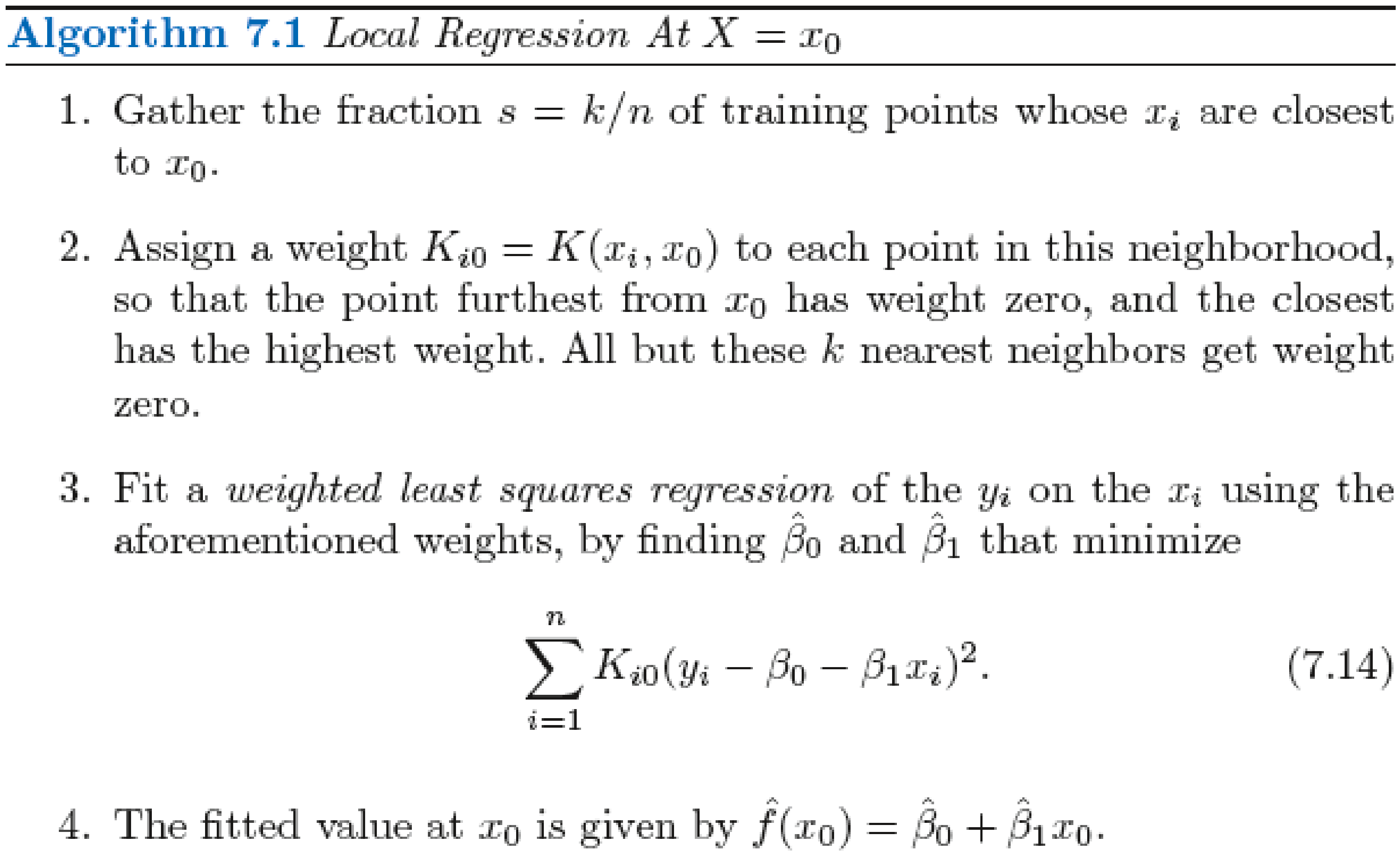
- It is a *memory-based* procedure because we need all the training data each time we wish to compute a prediction.

- The *span* plays a role like that of the tuning parameter λ in smoothing splines; it controls the flexibility of the non-linear fit.

Local Regression (cont.)

- The smaller the value of the span *s*, the more *local* and wiggly will be our fit.
- A very large value of *s* will lead to a global fit to the data using all of the training observations.
- We can use cross-validation to choose *s* or specify it directly.
- Another choice to be made includes how to define the weighting function *K*, and whether to fit a linear, constant, or quadratic regression.

Local Regression (cont.)



Algorithm 7.1 *Local Regression At $X = x_0$*

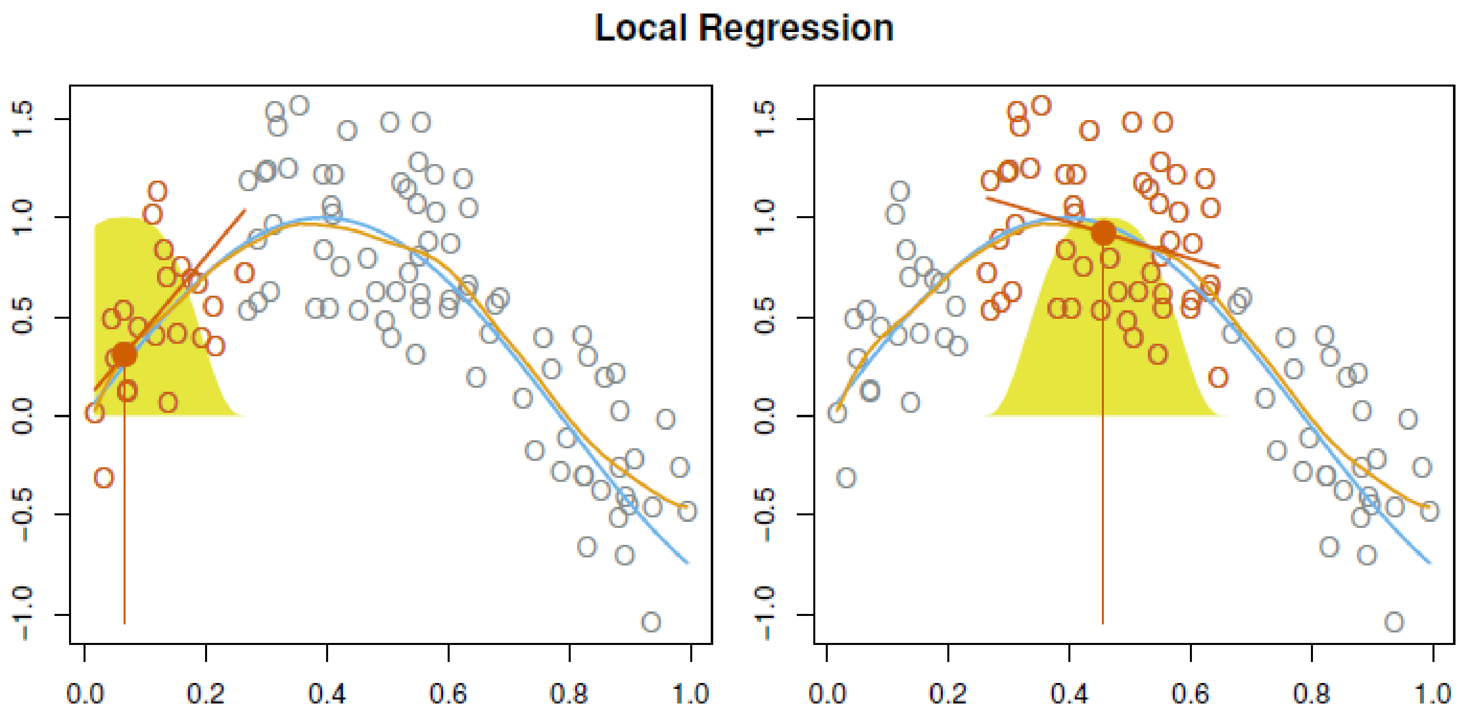
1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.
3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \qquad (7.14)$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Local Regression (cont.)

- With a sliding weight function, we fit separate linear fits over the range of *X* by weighted least squares.



Generalized Additive Models

- *Generalized additive models* (GAMs) allow for flexible nonlinearities in several variables, but retains the additive structure of linear models.

- GAMs can be applied with both quantitative and qualitative responses.

- In particular, we replace each linear component of the multiple linear regression model with a (smooth) non-linear function.

Generalized Additive Models (cont.)

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned}$$

- It is called an *additive* model because we calculate a separate f_j for each X_j, and then add together all of their contributions.

- We can use any of the previously discussed methods (smoothing splines, local regression, polynomial regression, etc.) as building blocks for fitting an additive model.

Generalized Additive Models (cont.)

- GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss.
- This means that we do not need to manually try out many different transformations on each variable individually.
- The non-linear fits can potentially make more accurate predictions for the response Y.

Generalized Additive Models (cont.)

- Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed.

- Thus, if we are interested in inference, GAMs provide a useful representation.

- The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom.

Generalized Additive Models (cont.)

- The main limitation of GAMs is that the model is restricted to be additive.
- With many variables, important interactions can be missed. However, we can manually add interaction terms to the GAM model by including additional predictors of the form X_j x X_k.
- Although we have not yet covered classification problems, note that GAMs can also be used in situations where *Y* is qualitative.

Naïve Bayes Classification

Classification

- Predicting a *qualitative* response for an observation can be referred to as *classifying* that observation, since it involves assigning the observation to a category, or class.

- Thus, *classification models* are supervised learning methods for which the true class labels for the data points are given in the training data.

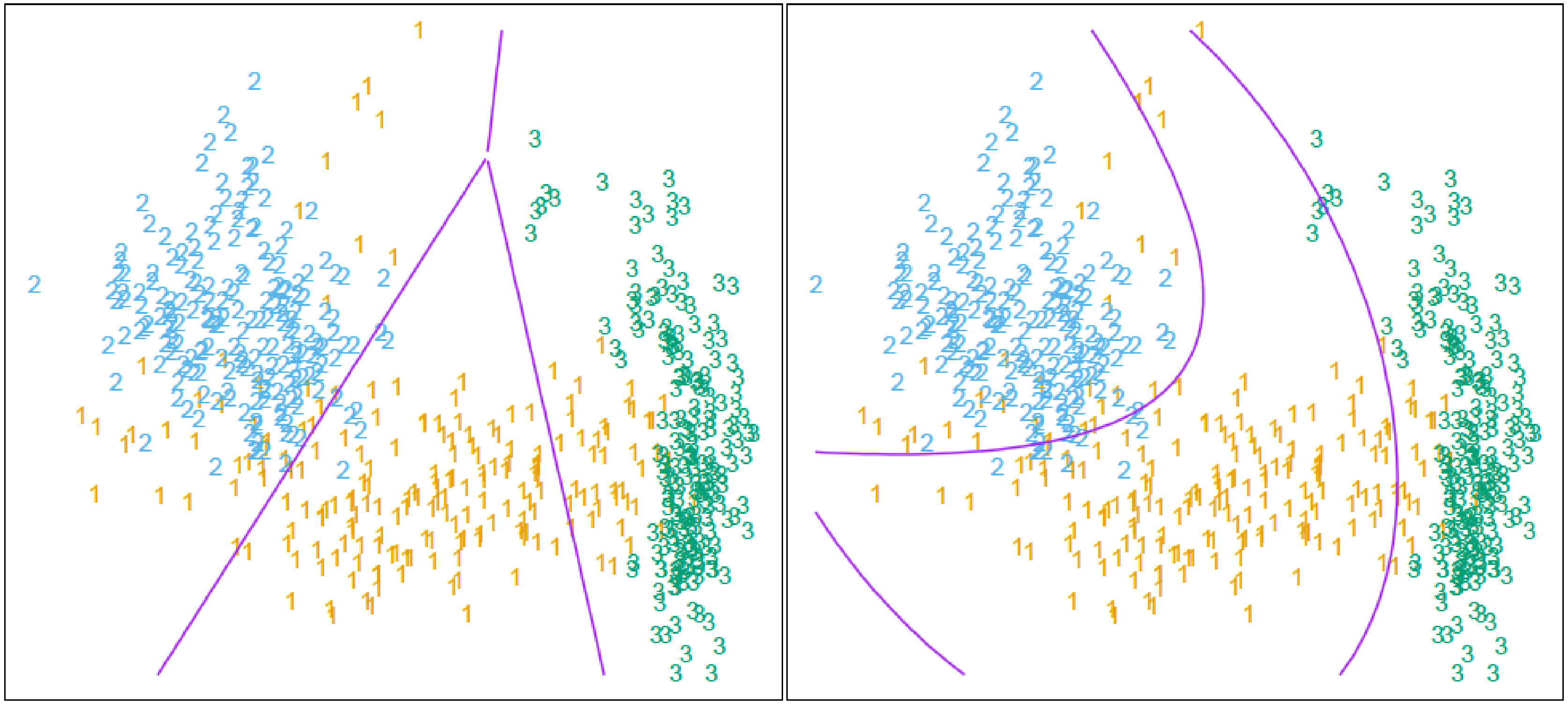
- The methods used for classification often predict the probability of each of the categories of a qualitative variable as the basis for making the classification decision.

Classification Setting

- Training data: $\{(x_1,g_1), (x_2,g_2),\ldots,(x_N,g_N)\}$
- The feature vector $X = (X_1, X_2,\ldots,X_p)$, where each X_j is quantitative.
- The response variable G is categorical s.t. $G \in G = \{1,2,\ldots,K\}$
- Form a predictor $G(x)$ to predict G based on X.
- Note that $G(x)$ divides the input space (feature vector space) into a collection of regions, each labeled by one class.

Classification Setting (cont.)

- For each plot, the feature vector space is divided into three pieces, each assigned with a particular class.



Classification Error Rate

- The *classification error rate* is the number of observations that are misclassified over the sample size:

$$\frac{1}{n}\sum_{i=1}^{n} I(\hat{Y}_i \neq Y_i)$$

where $I\left(\hat{Y}_i \neq Y_i\right) = 1$ if $\hat{Y}_i \neq Y_i$ and 0 otherwise.

- For binary classification let $\hat{Y}$ be a 0-1 vector of the predicted class labels and *Y* be a 0-1 vector of the observed class labels.

Review of Probability Basics

- Prior, conditional and joint probability for random variables
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 | X_2), P(X_2 | X_1)$
  - Joint probability: $\mathbf{X} = (X_1, X_2), P(\mathbf{X}) = P(X_1, X_2)$
  - Relationship: $P(X_1, X_2) = P(X_2 | X_1)P(X_1) = P(X_1 | X_2)P(X_2)$
  - Independence: $P(X_2 | X_1) = P(X_2), P(X_1 | X_2) = P(X_1), P(X_1, X_2) = P(X_1)P(X_2)$

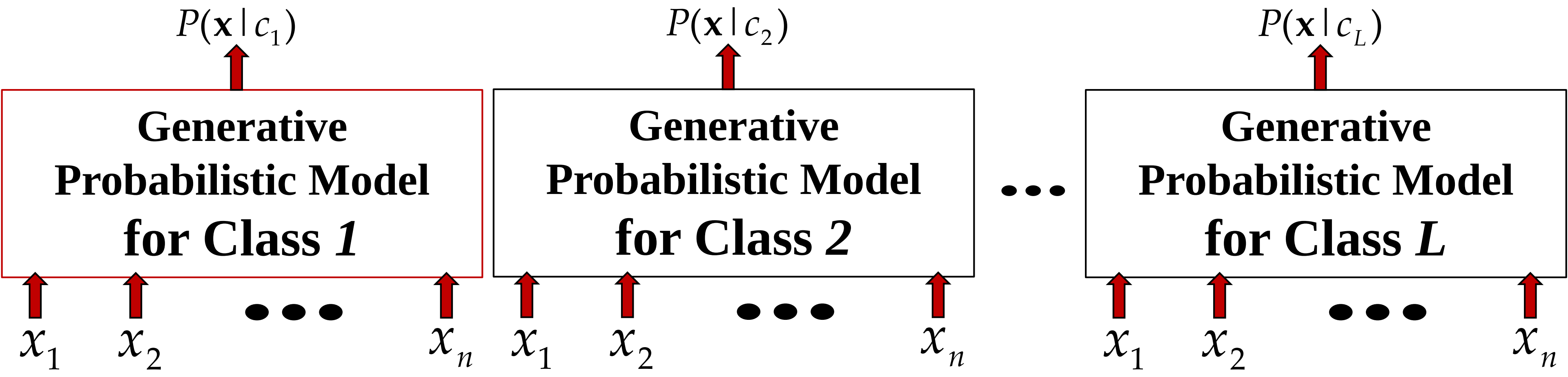
- **Bayes' Rule**

$$P(C | \mathbf{X}) = \frac{P(\mathbf{X} | C)P(C)}{P(\mathbf{X})} \quad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Probabilistic Classification

- Establishing a probabilistic model for classification
  - **Generative model**

$$P(\mathbf{X}|C) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$


$P(\mathbf{x}|c_1)$
$P(\mathbf{x}|c_2)$
$P(\mathbf{x}|c_L)$
Generative Probabilistic Model for Class *1*
Generative Probabilistic Model for Class *2*
Generative Probabilistic Model for Class *L*
x_1
x_2
x_n

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

Probabilistic Classification (cont.)

- MAP classification rule
  - **MAP**: **M**aximum **A** **P**osterior
  - Assign ***x*** to *c** if

$$P(C=c^* \mid \mathbf{X}=\mathbf{x}) > P(C=c \mid \mathbf{X}=\mathbf{x}) \quad c \neq c^*,\ c=c_1,\cdots,c_L$$

- Generative classification with the MAP rule
  - Apply Bayes' rule to convert them into posterior probabilities

$$P(C=c_i \mid \mathbf{X}=\mathbf{x}) = \frac{P(\mathbf{X}=\mathbf{x} \mid C=c_i)P(C=c_i)}{P(\mathbf{X}=\mathbf{x})}$$
$$\propto P(\mathbf{X}=\mathbf{x} \mid C=c_i)P(C=c_i)$$
$$\text{for } i=1,2,\cdots,L$$

  - Then apply the MAP rule

Naïve Bayes Classifier

- Bayes classification

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C)P(C) = P(X_1, \cdots, X_n \mid C)P(C)$$

Difficulty: learning the joint probability $P(X_1, \cdots, X_n \mid C)$

- Naïve Bayes classification
  - Assume that all input features are conditionally independent!

$$\begin{aligned} P(X_1, X_2, \cdots, X_n \mid C) &= P(X_1 \mid X_2, \cdots, X_n, C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2 \mid C) \cdots P(X_n \mid C) \end{aligned}$$

  - MAP classification rule: for $\mathbf{x} = (x_1, x_2, \cdots, x_n)$

$$[P(x_1 \mid c^*) \cdots P(x_n \mid c^*)]P(c^*) > [P(x_1 \mid c) \cdots P(x_n \mid c)]P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Naïve Bayes Classifier (cont.)

- Algorithm: Discrete-Valued Features
  - Learning Phase: Given a training set **S** of F features and L classes,

    For each target value of $c_i \; (c_i = c_1, \cdots, c_L\,)$

    $\hat{P}(C = c_i) \leftarrow$ estimate $P(C = c_i)$ with examples in **S**;

    For every feature value x_{jk} of each feature $X_j \; (j = 1, \cdots, F; k = 1, \cdots, N_j)$

    $\hat{P}(X_j = x_{jk} \mid C = c_i) \leftarrow$ estimate $P(X_j = x_{jk} \mid C = c_i)$ with examples in **S**;

    Output: $F * L$ conditional probabilistic (generative) models
  - Test Phase: Given an unknown instance $\mathbf{X}' = (a'_1, \cdots, a'_n)$

    "Look up tables" to assign the label c^* to $\mathbf{X}'$ if

$$[\hat{P}(a'_1 \mid c^*) \cdots \hat{P}(a'_n \mid c^*)]\hat{P}(c^*) > [\hat{P}(a'_1 \mid c) \cdots \hat{P}(a'_n \mid c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example (cont.)

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14 \qquad P(\text{Play}=No) = 5/14$

Example (cont.)

- Test Phase
  - **Given a new instance, predict its label**

    x'=(Outlook=*Sunny,* Temperature=*Cool,* Humidity=*High,* Wind=*Strong*)

  - **Look up tables achieved in the learning phrase**

P(Outlook=*Sunny*|Play=*Yes*) = 2/9	P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play=*Yes*) = 3/9	P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*Yes*) = 3/9	P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*Yes*) = 3/9	P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*Yes*) = 9/14	P(Play=*No*) = 5/14

  - **Decision making with the MAP rule**

P(*Yes*|**x**') ≈ [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|**x**') ≈ [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|**x**') < P(*No*|**x**'), we label **x**' to be "*No*".

Logistic Regression

Linear Methods for Classification

- *Decision boundaries* are linear.

- Two class problem:
  - The decision boundary between the two classes is a *hyperplane* in the feature vector space.

  - A hyperplane in the *p* dimensional input space is the set:

$$\{x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j = 0\}$$

Linear Methods for Classification (cont.)

- The two regions separated by a hyperplane:

$$\{x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j > 0\} \qquad \{x : \alpha_o + \sum_{j=1}^{p} \alpha_j x_j < 0\}$$

- For more than two classes, the decision boundary between any pair of classes *k* and *l* is a hyperplane.

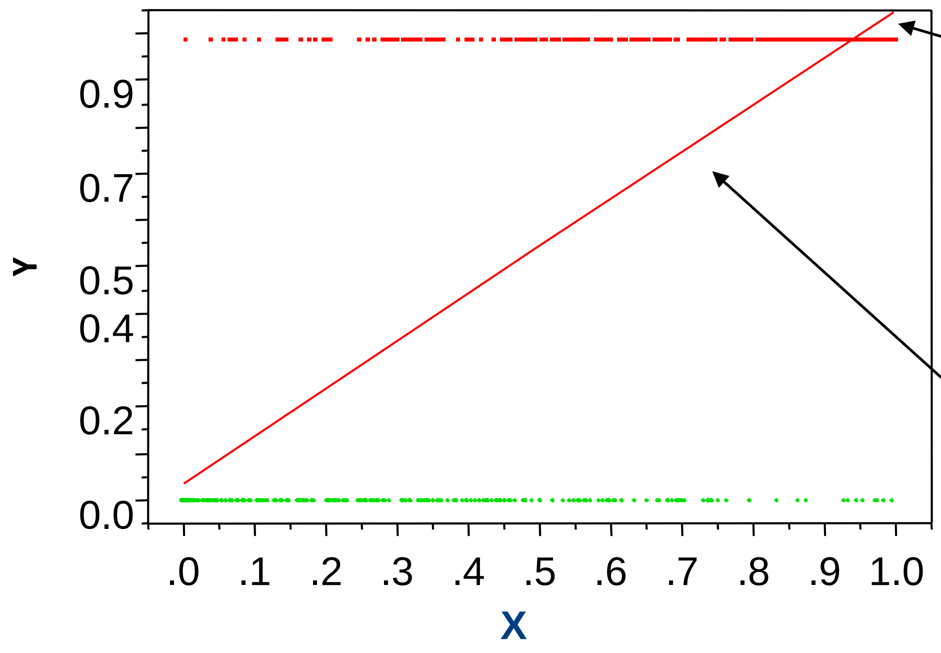
- Example method for deciding the hyperplane:
  - Logistic regression

Can we use Linear Regression?

- One rather formal justification is to view the linear regression as an estimate of conditional expectation.
- However, how good an approximation to conditional expectation is the rather rigid linear regression model?
- The key issue is that the fitted values can be negative or greater than 1.
- This is a consequence of the rigid nature of linear regression, especially if we make predictions outside the domain of the training data.

Can we use Linear Regression?

- When *Y* only takes on values of 0 and 1, we can see why is OLS linear regression is often not appropriate.



How do we interpret values greater than 1?

How do we interpret values of *Y* between 0 and 1?

Linear vs. Logistic Regression

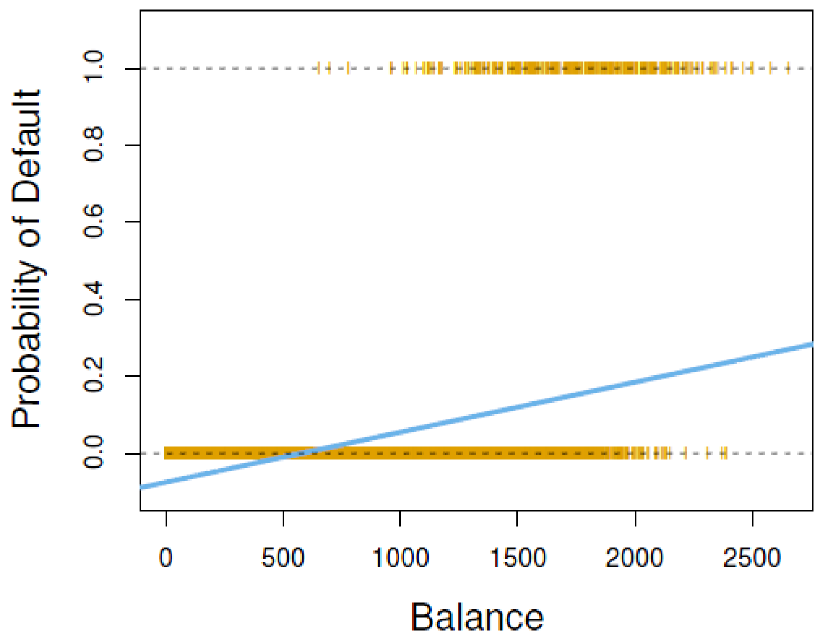
- Linear regression might produce probabilities less than zero or bigger than one because the regression line can take on any value between negative and positive infinity.

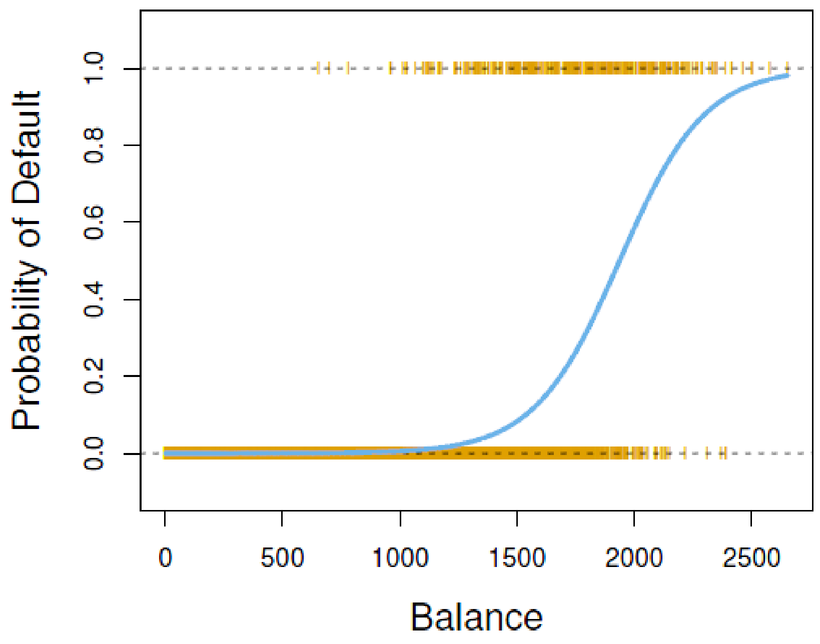
- Thus, the regression line almost always predicts the wrong value for *Y* in classification problems.

- *Logistic regression* is more appropriate, especially because we are interested in estimating the *probabilities* that *X* belong to each category, or class.

Linear vs. Logistic Regression (cont.)

- Below, the orange marks indicate the response *Y*, either 0 or 1. Linear regression does not estimate Pr(*Y* = 1|*X*) well, so logistic regression seems well suited to the task.





Logistic Regression

- There are two big branches of methods for classification. One is called *generative* modeling (which we saw with Naïve Bayes classification), and the other is called *discriminative* modeling.
  - A **generative** model learns the joint probability distribution.
  - A **discriminative** model learns the conditional probability distribution.

- *Logistic regression* for classification is a discriminative modeling approach, where we estimate the *posterior probabilities* of classes given *X* directly without assuming the marginal distribution on *X*.

- As a result, this method preserves linear classification boundaries.

Logistic Regression (cont.)

- A review of Bayes rule showed us that when we use 0-1 loss, we pick the class *k* that has the maximum posterior probability:

$$\hat{G}(x) = \arg \underset{k}{max} Pr(G = k|X = x)$$

- The decision boundary between classes *k* and *l* is determined by:

$$Pr(G = k|X = x) = Pr(G = l|X = x)$$

- That is, the *x*'s at which the two posterior probabilities of *k* and *l* are equal.

Logistic Regression (cont.)

- If we divide both sides by *Pr*(*G = l* | *X = x*) and take the log of this ratio, the previous equation is equivalent to:

$$\log \frac{Pr(G=k|X=x)}{Pr(G=l|X=x)} = 0$$

- Since we want to enforce a linear classification boundary, we assume the function above is linear:

$$\log \frac{Pr(G=k|X=x)}{Pr(G=l|X=x)} = a_0^{(k,l)} + \sum_{j=1}^{p} a_j^{(k,l)} x_j$$

- This is the basic assumption of logistic regression.

Logistic Regression (cont.)

- The log ratios of posterior probabilities are called *log-odds* or *logit* transformations.

- The posterior probabilities are given by the following two equations:

$$Pr(G = k|X = x) = \frac{\exp\left(\beta_{k0} + \beta_k^T x\right)}{1 + \sum_{l=1}^{K-1} \exp\left(\beta_{l0} + \beta_l^T x\right)} \text{ for } k = 1, \ldots, K-1$$

$$Pr(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp\left(\beta_{l0} + \beta_l^T x\right)}$$

Logistic Regression (cont.)

- For Pr(*G = k* | *X = x*) given previously:
  - These must sum to 1: $\sum_{k=1}^{K} Pr(G=k|X=x)=1$

- Similarities with linear regression on indicators:
  - Both attempt to estimate Pr(*G = k* | *X = x*), both have linear classification boundaries, and the posterior probabilities sum to 1 across classes

- Differences with linear regression on indicators:
  - For linear regression, approximate Pr(*G = k* | *X = x*) by a linear function of *x*; it is not guaranteed to fall between 0 and 1.
  - For logistic regression, Pr(*G = k* | *X = x*) is a nonlinear (sigmoid) function of *x*; it is guaranteed to range from 0 to 1.

Logistic Regression (cont.)

- How do we estimate the parameters and how do we fit a logistic regression model?
- What we want to do is find parameters that *maximize* the conditional *likelihood* of class labels *G* given *X* using the training data.
- We use the **Newton Raphson Algorithm** (a gradient descent method).

Logistic Regression (cont.)

- Starting with β^{old}, a single Newton-Raphson update is given by this matrix formula:

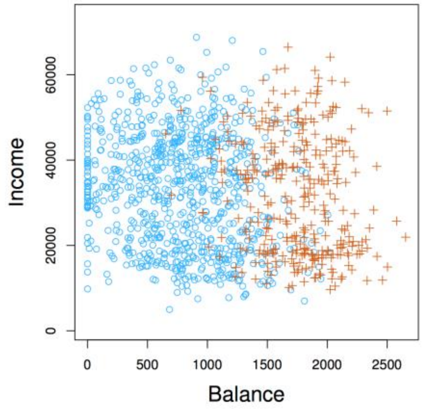
$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial l(\beta)}{\partial\beta}$$

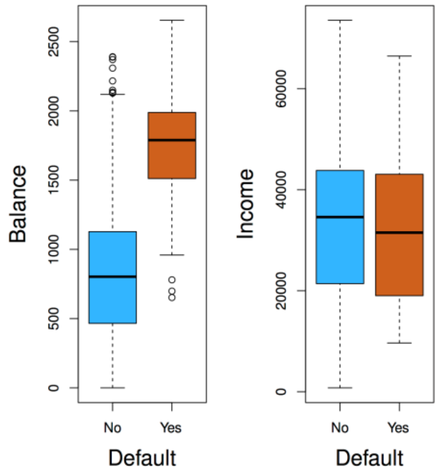
where the derivatives are evaluated at β^{old}.

- If given an old set of parameters, we update the new set of parameters by taking β^{old} minus the inverse of the Hessian matrix times the first-order derivative vector.

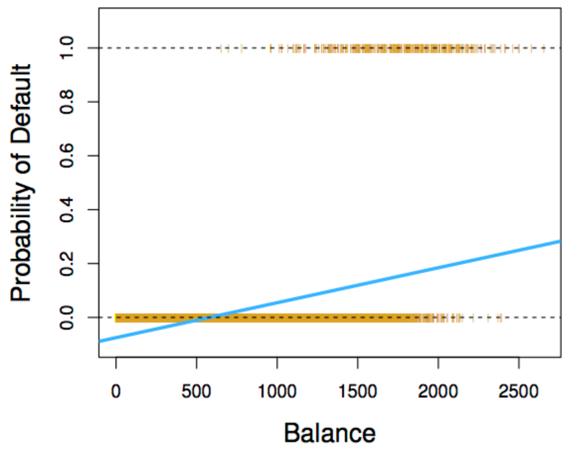
Logistic Regression: Credit Card Default Example

- We would like to be able to predict customers that are likely to default on their credit card.
- Possible *X* variables:
  - Annual Income
  - Monthly Credit Card Balance
- The *Y* variable (Default) is categorical: Yes (1) or No (0)

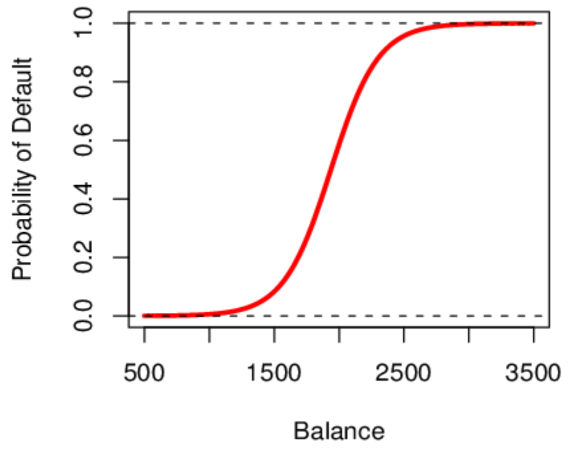




Logistic Regression: Credit Card Default Example (cont.)



- If we fit a linear regression model to the Default data, then:
  - For very low balances we predict a negative probability!
  - For high balances we predict a probability above 1!



- If we fit a logistic regression model, then the probability of default is between 0 and 1 for all balances.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \quad \log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

Logistic Regression: Credit Card Default Example (cont.)

- Interpreting what β_1 means is not very easy with logistic regression, simply because we are predicting Pr(*Y*) and not *Y*.

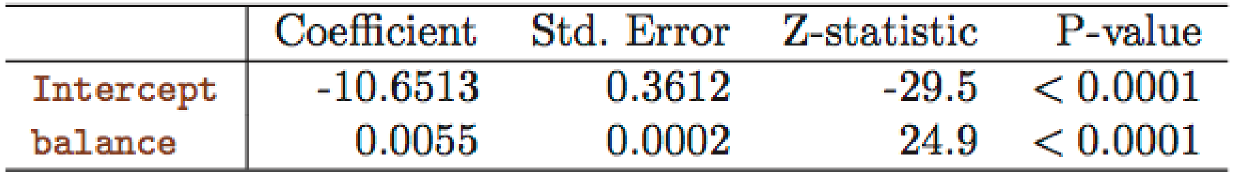
- In a logistic regression model, increasing *X* by one unit changes the *log odds* by β_1, or equivalently it multiplies the odds by e^{β_1}.

- If $\beta_1 = 0$, then there is <u>no</u> relationship between *Y* and *X*. If $\beta_1 > 0$, then when *X* gets larger so does the probability that $Y = 1$. If $\beta_1 < 0$, then when *X* gets larger the probability that $Y = 1$ gets smaller.

- How much increase or decrease in probability depends on the slope.

Logistic Regression: Credit Card Default Example (cont.)

- We still want to perform a hypothesis test to see whether we can be sure that β_0 and β_1 are significantly different from zero.
- We use a Z test and interpret the p-value as usual.
- In this example, the p-value for balance is very small and $\hat{\beta}_1$ is positive. This means that if the balance increases, then the probability of default will increase as well.



	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.6513	0.3612	-29.5	< 0.0001
balance	0.0055	0.0002	24.9	< 0.0001

Logistic Regression: Credit Card Default Example (cont.)

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.6513	0.3612	-29.5	< 0.0001
balance	0.0055	0.0002	24.9	< 0.0001

- What is the estimated probability of default for someone with a balance of $1000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.006$$

- What is the estimated probability of default for someone with a balance of $2000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 2000}}{1+e^{-10.6513+0.0055\times 2000}} = 0.586$$

Logistic Regression: Credit Card Default Example (cont.)

- We can also use student (0,1) as a predictor to estimate the probability that an individual defaults:

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	< 0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \boxed{}$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \boxed{}$$