

Econometrics & Machine Learning

Emmanuel Flachaire

Aix-Marseille University, AMSE

▶ <https://egallic.fr/ECB>

December 21, 2021

Econometrics & Machine Learning

① Introduction and General Principles

- The two Cultures
- Loss function and penalization
- In-sample, out-sample and cross validation

② Methods and Algorithms

- Ridge and Lasso Regression
- Classification and Regression Tree
- Bagging and Random Forests
- Boosting
- Support Vector Machine
- Neural Networks and Deep Learning

③ Using Machine Learning methods in Econometrics

- Misspecification detection
- Causal inference

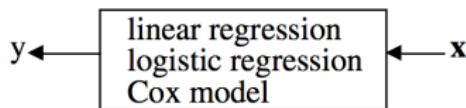
1. Introduction and General Principle

- The two Cultures
- Loss function and penalization
- In-sample, out-sample and cross validation

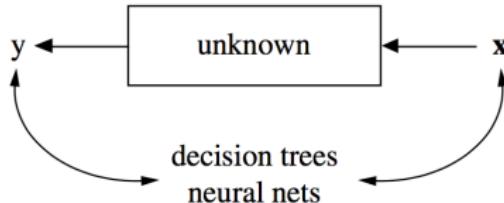
Statistical Modeling: The two Cultures¹

There are two cultures in the use of statistical modeling to reach conclusions from data:

- **Data Modeling Culture:** one assumes that the data are generated by a given stochastic data model (econometrics)



- **Algorithmic Modeling Culture:** one uses algorithmic models and treats the data mechanism as unknown (machine learning)



¹Léo Breiman, *Statistical Science*, 2001, Vol. 16, No. 3, 199-231

Statistical Modeling: The two Cultures

Léo Breiman (*Statistical Science*, 2001):

Upon my return, I started reading the *Annals of Statistics*, the flagship journal of theoretical statistics, and was bemused. Every article started with

Assume that the data are generated by the following model: ...

wavelet theory. Even in applications, data models are universal. For instance, in the *Journal of the American Statistical Association* (JASA), virtually every article contains a statement of the form:

Assume that the data are generated by the following model: ...

... an uncritical use of data models.

Misspecification bias

- Let's consider a quite general model:² $y = m(X) + \varepsilon$
- Assume that X is fixed. The expected (squared) prediction error, or EPE, is equal to

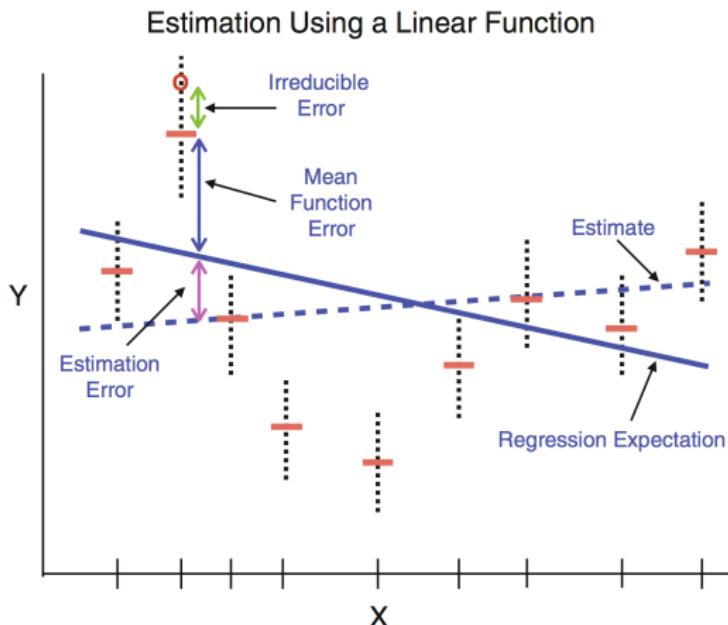
$$\begin{aligned} E(y - \hat{y})^2 &= E[m(X) + \varepsilon - \hat{m}(X)]^2 \\ &= \underbrace{E[m(X) - \hat{m}(X)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\varepsilon)}_{\text{Irreducible}} \end{aligned}$$

- The focus of Machine Learning is to estimate m with the aim of minimizing the reducible error
- Reducible error = MSE = $[\text{Bias}(\hat{m}(X))]^2 + \text{Var}(\hat{m}(X))$
- Assuming that the data are generated by a specific model, or that the model is correctly specified, remains to assume that the (misspecification) bias is zero: $\text{Bias}(\hat{m})=0$

² y is a vector and X a matrix of observations, m a function, ε an error term

Misspecification bias: linear model

Fig. 1.4 Estimation of a nonlinear response surface under the true linear model perspective (The *broken line* is an estimate from a given dataset, *solid line* is the expectation of such estimates, the *vertical dotted lines* represent conditional distributions of Y with the *red bars* as each distribution's mean.)

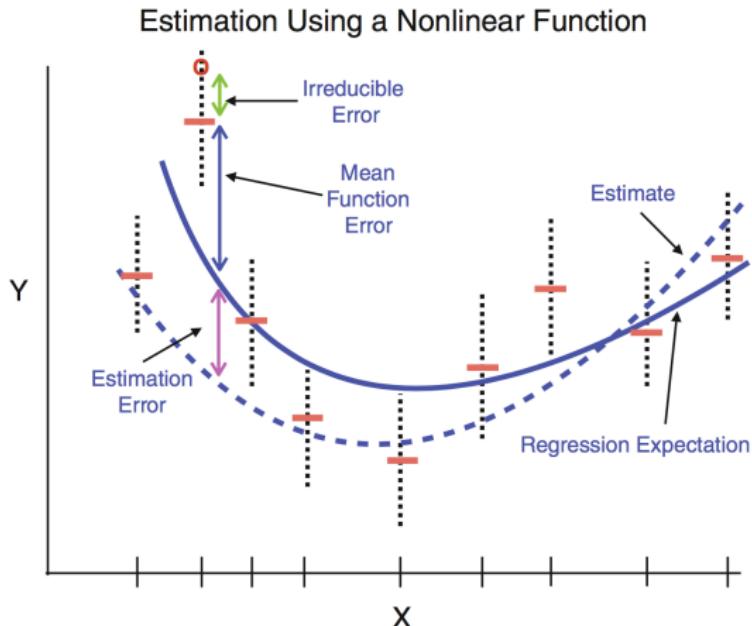


(Source: Berk, 2016)

$$\text{Reducible error} = \underbrace{\text{mean function error} + \text{estimation error}}_{\text{misspecification bias}}$$

Misspecification bias: quadratic model

Fig. 1.5 Estimation a nonlinear response surface under the true nonlinear model perspective (The *broken line* is an estimate from a given dataset, *solid line* is the expectation of such estimates, the *vertical dotted lines* represent conditional distributions of Y with the *red bars* as each distribution's mean.)



(Source: Berk, 2016)

$$\text{Reducible error} = \underbrace{\text{mean function error}}_{\text{misspecification bias}} + \text{estimation error}$$

Econometrics and Machine Learning

- **Parametric econometric:** we assume that the data come from a generating process that takes the following form

$$y = X\beta + \varepsilon$$

→ probability theory is a foundation of econometrics

- **Machine learning:** we do not make any assumption on how the data have been generated

$$y \approx m(X)$$

→ probability theory is not required

- Nonparametric econometrics makes the link between the two
- Machine Learning: an extension of **nonparametric econometric**

General Principle : optimization problem

Find the solution \hat{m} to the optimization problem:

$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \mathcal{L}(y_i, m(X_i)) \quad \text{subject to} \quad \|m\|_{\ell_q} \leq t \quad (1)$$

which can be rewritten in Lagrangian form, for some $\lambda \geq 0$:

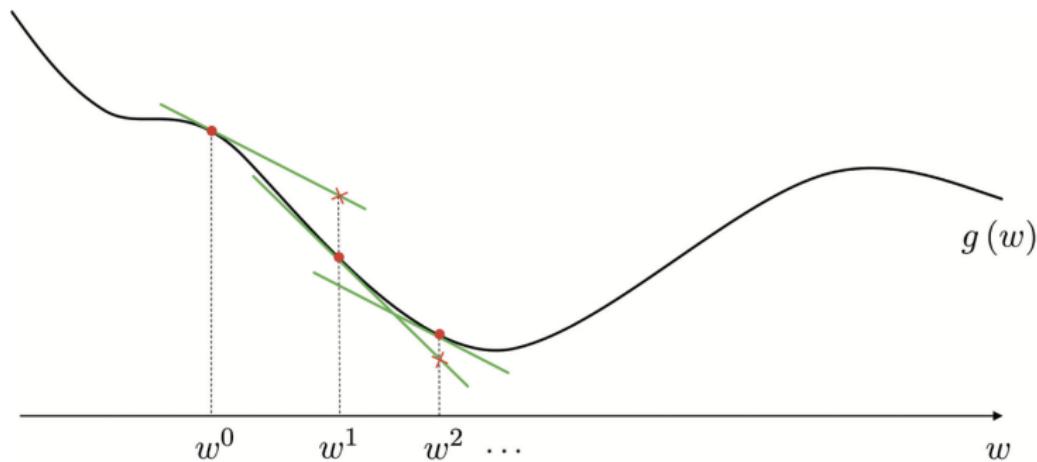
$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \underbrace{\mathcal{L}(y_i, m(X_i))}_{\text{loss function}} + \underbrace{\lambda \|m\|_{\ell_q}}_{\text{penalization}} \quad (2)$$

- The goal is to minimize a loss function under constraint
- It is usually done by numerical optimization

General Principle : resolution by numerical optimization

Gradient Descent

Use *linear* approximations at each steps, from Taylor expansion



(Source: Watt et al., 2016)

Algorithm: Gradient descent

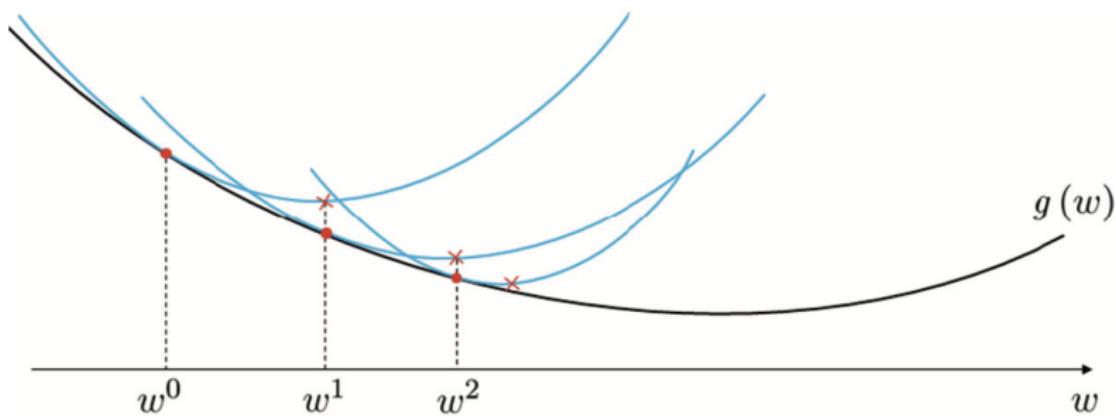
Input: differentiable function g , fixed step length α , initial point x^0

Repeat until stopping condition is met: $w^k = w^{k-1} - \alpha g'(w^{k-1})$

General Principle : resolution by numerical optimization

Newton's Method

Use quadratic approximations at each steps, from Taylor expansion³



(Source: Watt et al., 2016)

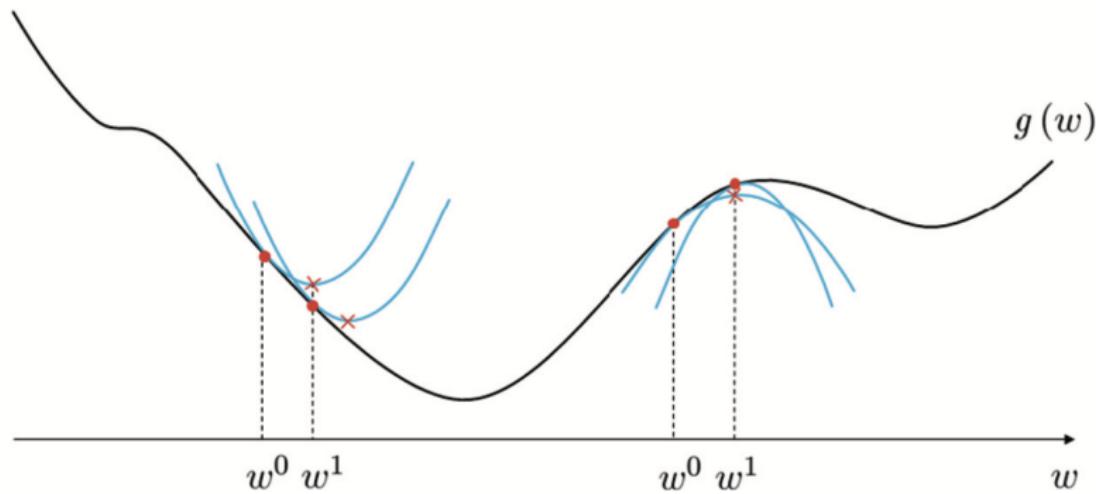
- Converges in fewer steps than gradient descent in convex fct
- Does not require step length to be determine

³The second order Taylor series approximation centered at w^k is equal to $h(w) = g(w^k) + g'(w^k)(w - w^k) + \frac{1}{2}g''(w^k)(w - w^k)^2$

General Principle : resolution by numerical optimization

Newton's Method

Use *quadratic* approximations at each steps, from Taylor expansion



(Source: Watt et al., 2016)

- It is used to find stationary points of a function: $g'(w) = 0$.
- It can lead to a maximum in concave function.

Regression: a simple Machine Learning method

Machine Learning (ML): solve the optimization problem

$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \underbrace{\mathcal{L}(y_i, m(X_i))}_{\text{loss function}} + \underbrace{\lambda \|m\|_{\ell_q}}_{\text{penalization}}$$

Let us consider:

- $\mathcal{L} = \ell_2$ (Euclidian distance): $\mathcal{L}(y_i, m(X_i)) = (y_i - m(X_i))^2$
- m is a linear function of parameters: $y_i \approx X_i\beta$ with $\beta \in R^p$
- no penalization: $\lambda = 0$

Thus, we have:

$$\hat{\beta} = \operatorname{argmin} \left\{ \sum_{i=1}^n (y_i - X_i\beta)^2 \right\},$$

It is the minimization of the Sum of Squared Residuals (SSR) in a linear regression model, that is, $\hat{\beta}$ is the OLS estimator.⁴

⁴A Gradient Descent method can be used to solve this optimization problem.



Linear regression from a Machine Learning perspective

Let us consider the simple linear regression model:

$$y = \beta_0 + \beta_1 x + \varepsilon \quad (3)$$

From a Machine Learning perspective:

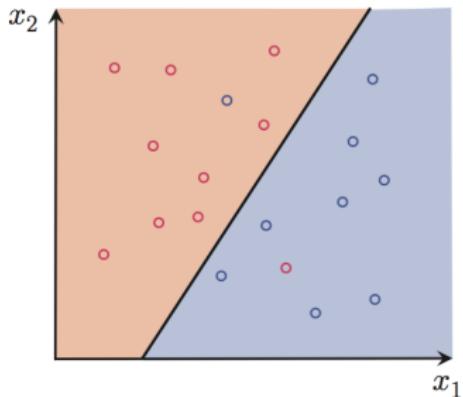
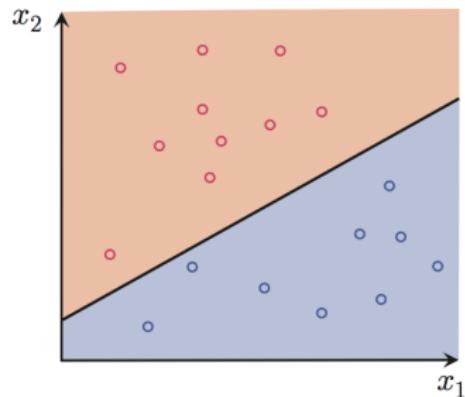
- The linear regression provides the best straight line approximation of the relationship between y and x ⁵
- OLS estimators are obtained by minimizing prediction errors.
No probability theory is required!

Econometrics put statistical assumptions on (3) in order to derive properties of the OLS estimators and to make inference.⁶

⁵In the sense that it minimizes prediction errors

⁶convergence, unbiased/biased estimators, BLUE, statistical tests, etc.

Classification: a simple Machine Learning method



(Source: Watt et al., 2016)

- we aim to learn a hyperplane $X\beta = 0$ (shown here in black) to separate feature representations of the two classes.⁷
- left panel: perfect linear separation
- right panel: two overlapping classes → minimize the number of missclassified points that end up in the wrong half-space.

⁷ $X = [\iota, x_1, x_2]$ is a $n \times 3$ matrix.

Classification: the perceptron

A hyperplane placing the points on its correct side is as follows:

$$X\beta > 0 \quad \text{if } y_i = +1$$

$$X\beta < 0 \quad \text{if } y_i = -1$$

In other words, with $y \in \{-1, +1\}$:

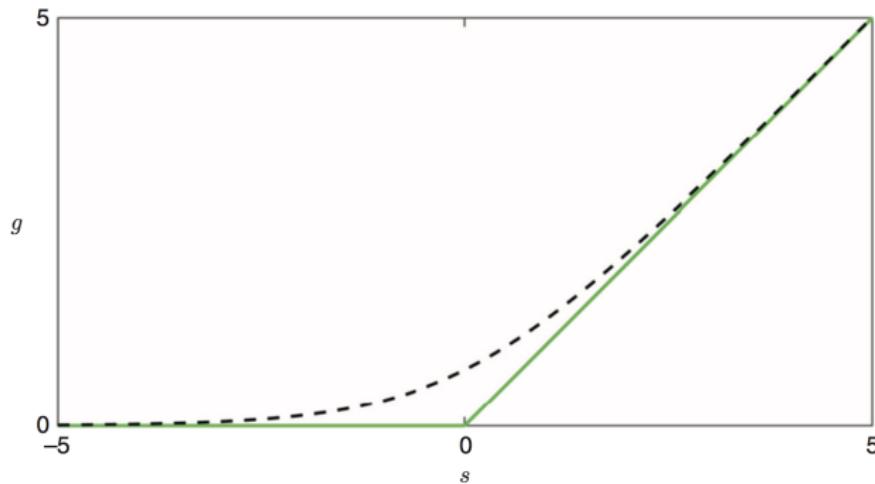
- if y_i is correctly classified: $y_i(X_i\beta) > 0$
- if y_i is missclassified: $y_i(X_i\beta) < 0$

To minimize the aggregated distance of missclassified points to the hyperplane, we can solve

$$\underset{\beta}{\text{Minimize}} \sum_{i=1}^n \max(0, -y_i(X_i\beta)),$$

where $\max(0, -y_i(X_i\beta))$ is the **perceptron** or **max** loss function.

Classification: smooth version of the perceptron



(Source: Watt et al., 2016)

- The **perceptron** loss function is **non-differentiable** (in green).⁸
- The **softmax** loss function is a **smooth approximation** (black).⁹

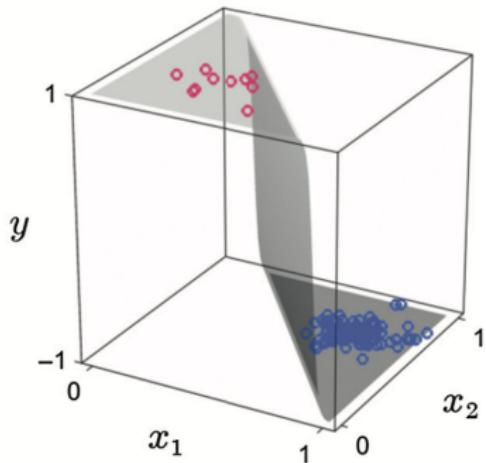
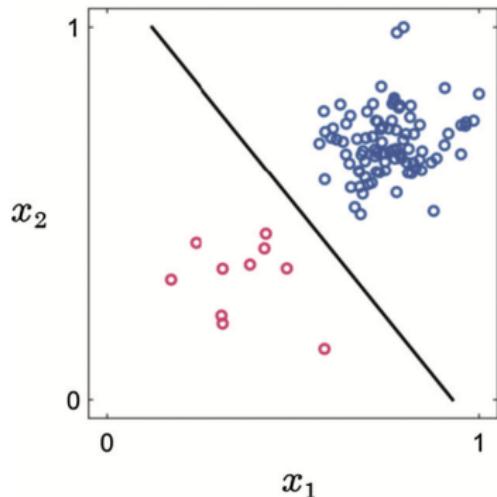
$$g(s) = \text{softmax}(0, s) = \log(1 + e^s)$$

⁸ $g(s) = \max(0, s)$

⁹ $g(s) = \log(1 + e^s)$. Gradient descent and Newton's methods can be used.



Classification: softmax and perceptron



(Source: Watt et al., 2016)

Minimizing the softmax loss function gives $\hat{\beta}$, that define:

- the linear separator $X\hat{\beta} = 0$ shown in the left panel,
- the surface $y(x) = 2\Lambda(X\hat{\beta}) - 1$ shown in the right panel.

The softmax model is a smooth approximation of the perceptron

Classification: logit regression and perceptron

Minimize the softmax loss function:¹⁰

$$\underset{\beta}{\text{Minimize}} \sum_{i=1}^n \log \left(1 + e^{-y_i(X_i\beta)} \right),$$

is similar to maximize the log-likelihood in a logit model:

$$\underset{\beta}{\text{Maximize}} \sum_{i=1}^n y'_i \log \Lambda(X_i\beta) + (1 - y'_i) \log (1 - \Lambda(X_i\beta))$$

with $y'_i \in \{0, 1\}$ and $\Lambda(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$ is the logistic function¹¹

→ Logit model = softmax model

→ The logit model is a smooth approximation of the perceptron

¹⁰ $\text{softmax}(0, -y_i(X_i\beta)) = \log(1 + e^{-y_i(X_i\beta)})$

¹¹ $\log \Lambda(X_i\beta) = -\log(1 + e^{-X_i\beta})$ and $\log(1 - \Lambda(X_i\beta)) = -\log(1 + e^{X_i\beta})$

Classification: a simple Machine Learning method

Machine Learning: solve the optimization problem

$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \underbrace{\mathcal{L}(y_i, m(X_i))}_{\text{loss function}} + \underbrace{\lambda \|m\|_{\ell_q}}_{\text{penalization}}$$

Let us consider:¹²

- the softmax loss function: $\mathcal{L} = \text{softmax}(0, -y_i(X_i\beta))$
- no penalization: $\lambda = 0$.

Thus, we have:¹³

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \log \left(1 + e^{-y_i(X_i\beta)} \right) \right\},$$

which is similar to maximize the log-likelihood in a **logit regression** model, that is, $\hat{\beta}$ is the **MLE estimator**.

¹² $y_i \approx m(X_i) = 1_{\pm}(X_i\beta \geq 0) = \{+1 \text{ if } X_i\beta \geq 0; -1 \text{ if } X_i\beta < 0\}$

¹³A Gradient Descent method can be used to solve this optimization problem.



Logit regression from a Machine Learning perspective

Let us consider the logit regression model:¹⁴

$$\mathbb{E}(y'|X) = \mathbb{P}(y' = 1) = \Lambda(X\beta) \quad (4)$$

From a Machine Learning perspective:

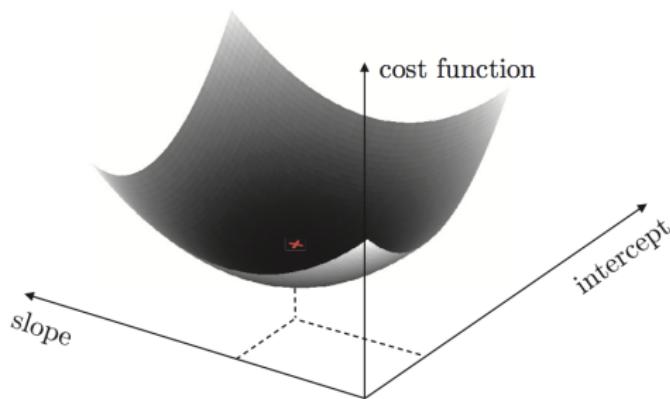
- The logit model is a smooth approximation of the perceptron
- MLE estimator is obtained by minimizing classification errors.
No probability theory is required!

Econometrics put statistical assumptions on (4) in order to derive properties of the MLE estimator and to make inference.

¹⁴Since $y' = \{0, 1\}$, then $\mathbb{E}(y'|X) = 0 \times \mathbb{P}(y' = 0) + 1 \times \mathbb{P}(y' = 1)$

Linear/Logit models from a Machine learning perspective

- Optimal parameters: Minimize prediction/classification errors
- The convenience of convexity:

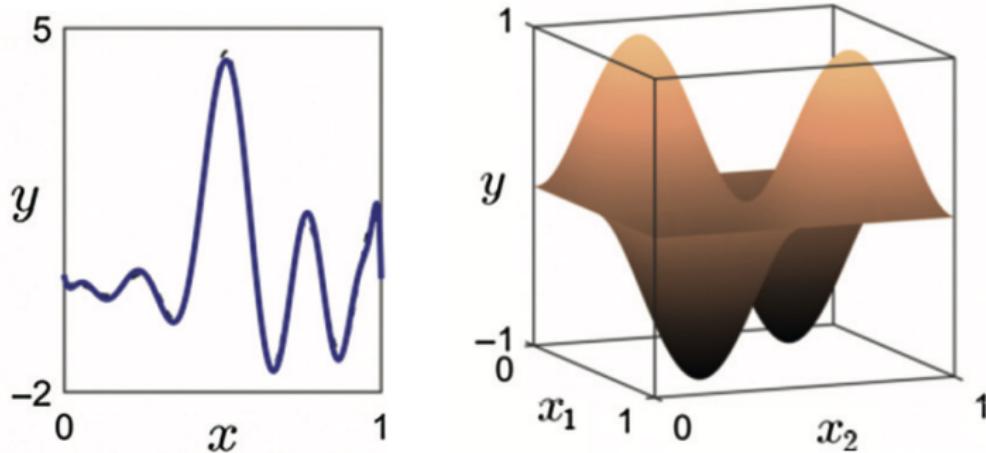


(Source: Watt et al., 2016)

A unique solution is easily obtained numerically/analytically.

- Using probability theory, properties of the optimal parameters are derived and inference can be drawn (Econometrics)

Moving beyond linearity: Regression



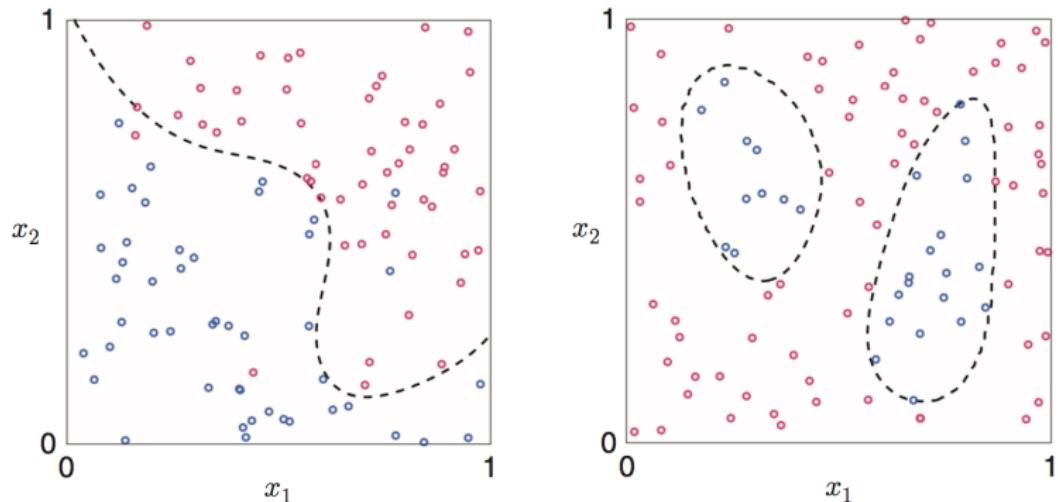
(Source: Watt et al., 2016)

- Non-linearity (left panel) and interaction effects (right panel).
- Knowledge-driven feature design are used in Econometrics.¹⁵
- Automatic feature design is used in Machine Learning

¹⁵fixed transformed covariates: quadratic, cubic, etc. □ and/or cross-products



Moving beyond linearity: Classification



(Source: Watt et al., 2016)

- Non-linearity (left panel) and interaction effects (right panel).
- Knowledge-driven feature design are used in Econometrics.¹⁶
- Automatic feature design is used in Machine Learning

¹⁶fixed transformed covariates: quadratic, cubic, etc. □ and/or cross-products



Nonparametric Econometrics

Machine Learning:

- High **non-linearity** and strong **interaction effects** are taken into account with **automatic** feature design.
- In general, a non-convex function is minimized.

Nonparametric Econometrics:

- A **nonparametric regression** take into account such effects.
- It may work well in **small-dimension**, not in high dimension.¹⁷

Machine Learning is an extension of Nonparametric Econometrics.

¹⁷Because of the curse of dimensionality. Note that GAM models may capture *automatically* non-linearities, but not interaction effects.

1. Introduction and General Principle

- The two Cultures
- Loss function and penalization
- In-sample, out-sample and cross validation

General Principle

Machine Learning: solve the optimization problem

$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \underbrace{\mathcal{L}(y_i, m(X_i))}_{\text{loss function}} + \underbrace{\lambda \|m\|_{\ell_q}}_{\text{penalization}}$$

- Choice of the **loss function**:
 - \mathcal{L} → conditional mean, quantiles, expectiles
 - m → linear, logit, splines, tree-based models, neural networks
- Choice of the **penalization**:
 - ℓ_q → lasso, ridge
 - λ → over-fitting, under-fitting, cross validation

Loss funct: Tradeoff between flexibility & interpretability

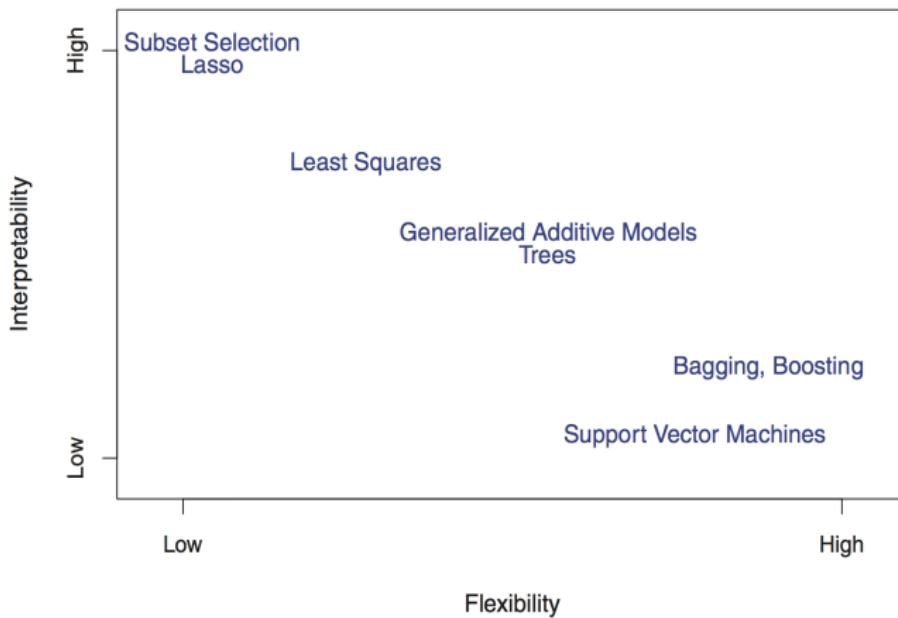
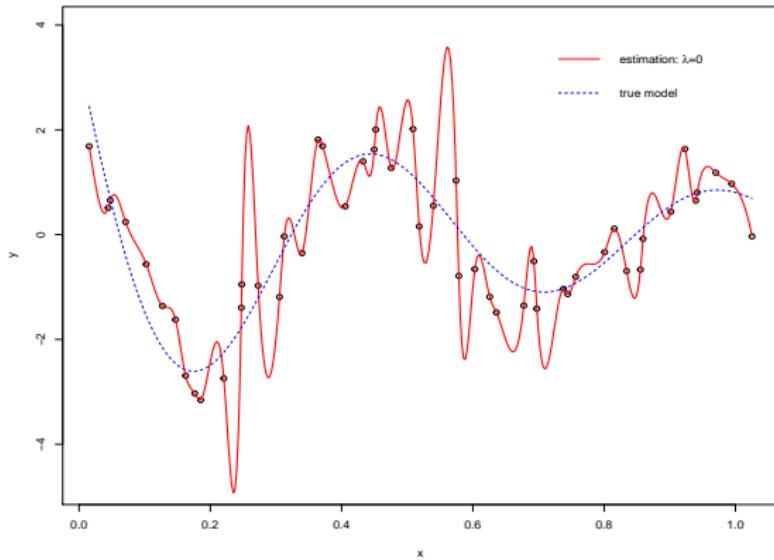


FIGURE 2.7. A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.

(Source: James et al., 2013)

Over-fitting

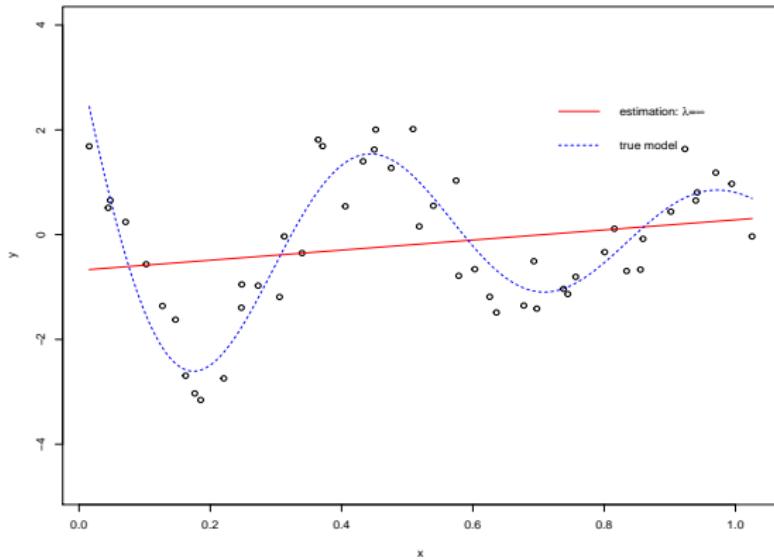
A model with high flexibility may fit perfectly observations used for estimation, but very poorly new observations



→ **penalization**: put a price to pay for having a more flexible model

Under-fitting

If we put a huge cost for a more complex model, $\lambda = \infty$, we obtain a linear regression model



→ if the cost is too large: low variance, but high bias

Penalization: Tradeoff between bias & variance

Penalization: put a price to pay for a having more flexible model

- $\lambda = 0$: it interpolates data low bias, high variance
 - $\lambda = \infty$: linear model high bias, low variance
- the penalty parameter $\lambda \equiv \text{bias}/\text{variance}$ tradeoff

Role of λ : avoid over-fitting and poor prediction with new data

Choice of λ : automatic selection procedures are based on model's performance evaluated out-sample, by cross-validation

1. Introduction and General Principle

- The two Cultures
- Loss function and penalization
- In-sample, out-sample and cross validation

Model assessment

- The best model has the lowest prediction error. With squared error loss, the mean squared prediction error is equal to

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_\lambda(x_i))^2 = \frac{\text{SSR}}{n}$$

- Due to overfitting, we cannot use SSR and R^2 based on the sample used for estimation (\equiv **in-sample**, training sample)
- We are interested in the accuracy of the predictions obtained from previously unseen data (\equiv **out-sample**, test sample)
- The in-sample MSE (**training error**) can be a poor estimate of the out-sample MSE (**test error**)

Model assessment

- In order to select the best model with respect to **test error**, we need to estimate this test error (out-sample MSE)
- There are two common approaches:
 - We can indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting
→ **penalization ex-post** ... R^2_{adj} , AIC, BIC
 - We can directly estimate the test error, using either a validation set approach or a cross-validation approach
→ **penalization ex-ante**
- CV provides a direct estimate of test error, makes fewer assumptions about the true model and can be used widely
- In the past, performing CV was computationally prohibitive. Nowadays, the computations are hardly ever an issue

Out-sample: Validation set



FIGURE 5.1. A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.

(Source: James et al., 2013)

- We split randomly the sample in two groups of observations: a training set ($q - 1$ obs.) and a validation/test set ($n - q$ obs)

$$1 \text{ estimation, } n - q \text{ values} \quad \rightarrow \quad \text{MSE} = \frac{1}{n - q} \sum_{i=q}^n (y_i - \hat{y}_i)^2$$

Cross-Validation: LOOCV or n -fold CV¹⁸

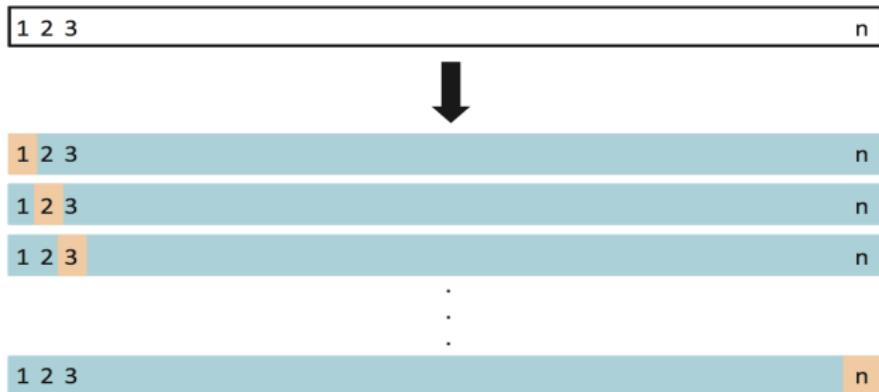


FIGURE 5.3. A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.
(Source: James et al., 2013)

$$n \text{ estimations, } n \text{ values} \rightarrow \text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

¹⁸LOOCV: leave-one-out cross-validation

Cross-Validation: K -fold CV

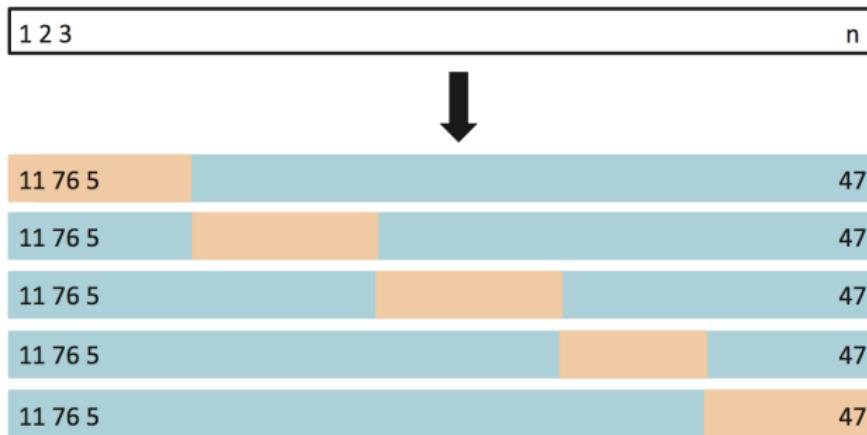
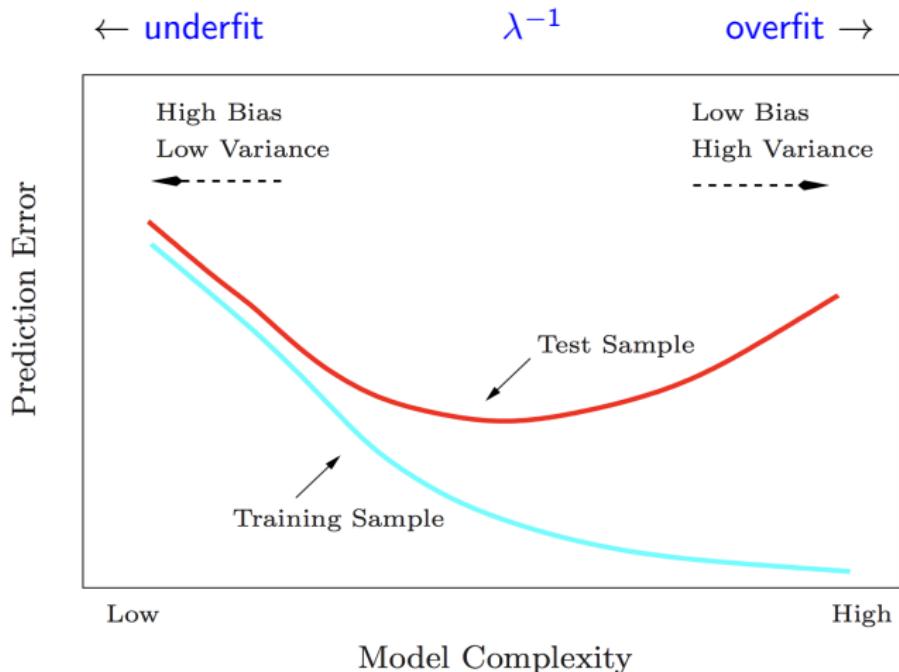


FIGURE 5.5. A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.
(Source: James et al., 2013)

$$K \text{ estimations, } n \text{ values} \rightarrow \text{MSE} = \frac{1}{n} \sum_{k=1}^K \sum_{i \in G_k} (y_i - \hat{y}_i)^2$$

Prediction error in-sample vs. out-sample



(Source: Hastie et al., 2009)

Underfitting: the model performs poorly on training and test samples

Overfitting: performs well on training sample, but generalizes poorly on test sample

Standardization and Normalization

- Several ML methods are sensitive to the units of the covariates as Ridge/Lasso regressions, SVM and Neural Networks
- The results may differ substantially when multiplying a given covariate by a constant (meters/kilometers, kilograms/grams)
- It is best to standardize the data before using these methods:

$$\frac{x}{\sqrt{\text{Var}(x)}} \quad \text{or} \quad \frac{x - \bar{x}}{\sqrt{\text{Var}(x)}}$$

so that they are all on the same scale

- Normalization is another scaling technique where the values end up ranging between 0 and 1:

$$\frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

2. Methods and Algorithms

- Ridge and Lasso Regression
- Classification and Regression Tree
- Bagging and Random Forests
- Boosting
- Support Vector Machine
- Neural Networks and Deep Learning

Introduction

Linear regression

$$y = X\beta + \varepsilon \quad n \text{ observations, } p \text{ covariates}$$

Least Squares

- Collinearity or many irrelevant covariates \rightarrow high variance
- More covariates than observations, $p > n \rightarrow$ undefined

Ridge and Lasso

- Collinearity, many irrelevant covariates \rightarrow smaller variance
- High-dimensional data analysis, $p \gg n \rightarrow$ feasible

Shrinkage Methods

$$\underset{\alpha, \beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - \alpha - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j|^q$$

It is equivalent to minimize SSR subject to $\sum_{j=1}^p |\beta_j|^q \leq c$

- No penalization corresponds to OLS *unbiased* estimation
- The penalization restricts the magnitude of the coefficients
- It shrinks the coefficients toward 0 as $\lambda \nearrow$ (or $c \searrow$)
- It introduces some *bias* in the coefficients

→ Add some bias if it leads to a substantial decrease in variance

Shrinkage Methods

$$\underset{\alpha, \beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - \alpha - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j|^q$$

It is equivalent to minimize SSR subject to $\sum_{j=1}^p |\beta_j|^q \leq c$

- Idea: biased coeff may result in model with smaller MSE
- The penalty term λ is a bias-variance tradeoff
- λ is selected by cross-validation (MSE out-sample)

Overall, use shrinkage methods when OLS exhibits large variance
(with many irrelevant or highly correlated covariates)

Standardization

$$\underset{\alpha, \beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - \alpha - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j|^q$$

It is equivalent to minimize SSR subject to $\sum_{j=1}^p |\beta_j|^q \leq c$

- The results are **sensitive to the scale** of the covariates
- If a covariate is divided by 10, its coefficient is multiplied by 10, which has an impact on the constraint
- It is best to **standardize** covariates before using shrinkage methods, so that they are all on the same scale:

$$\frac{x}{\sqrt{\text{Var}(x)}} \quad \text{or} \quad \frac{x - \bar{x}}{\sqrt{\text{Var}(x)}}$$

Ridge Regression

$$\underset{\alpha, \beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - \alpha - X_i \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Ridge = shrinkage method based on the ℓ_2 norm ($q = 2$)

- The restriction is convex and makes the problem easy to solve:

$$\hat{\beta} = (X^\top X + \lambda \mathbb{I}_n)^{-1} X^\top y$$

where \mathbb{I}_n is the $n \times n$ identity matrix

- $\lambda > 0$: $(X^\top X + \lambda \mathbb{I}_n)$ non-singular even if X is not of full rank
- Ridge method is defined in high-dimensional problems $p \gg n$

Lasso Regression

$$\underset{\alpha, \beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - \alpha - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

Lasso = shrinkage method based on the ℓ_1 norm ($q = 1$)

- The restriction is convex and makes the problem easy to solve numerically, but there is no close expression as in ridge
- The nature of the constraint will cause some coefficients to be exactly zero, with λ sufficiently large (or c sufficiently low)
- Lasso makes variable selection with many irrelevant variables
- Lasso is appropriate with sparse model, in which only a relative small number of covariates play an important role

Lasso vs. Ridge

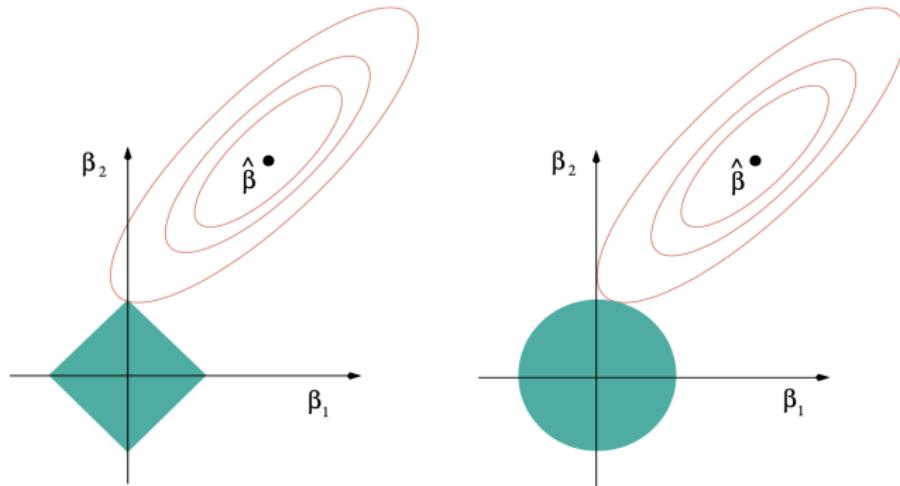


Figure 2.2 Estimation picture for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions $|\beta_1|+|\beta_2| \leq t$ and $\beta_1^2+\beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the residual-sum-of-squares function. The point $\hat{\beta}$ depicts the usual (unconstrained) least-squares estimate

(Source: Hastie et al., 2015)

Unlike the Ridge constraint, the Lasso constraint has corners
If the solution occurs at a corner, it has one parameter equal to 0

Lasso vs. Ridge

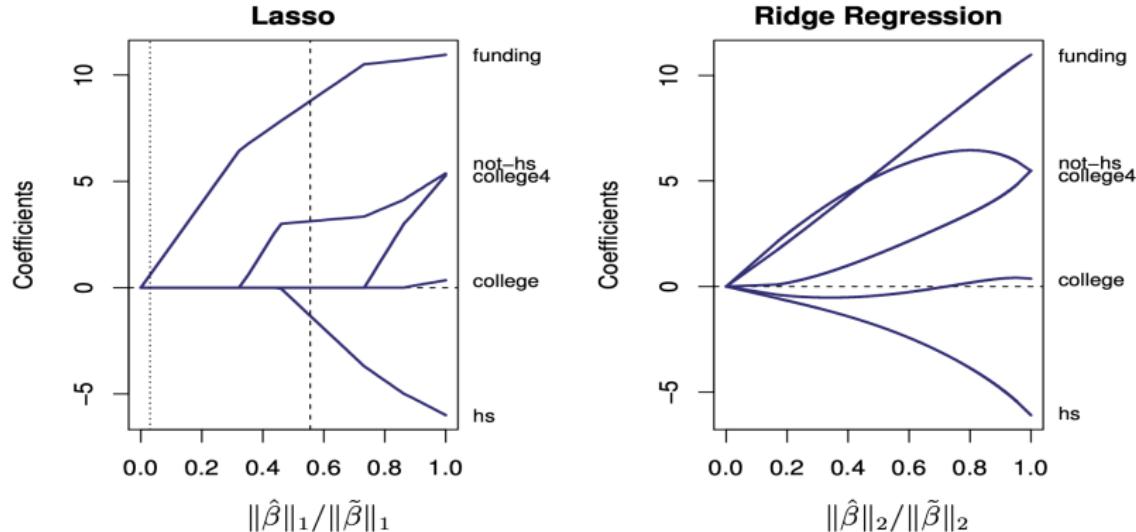


Figure 2.1 Left: Coefficient path for the lasso, plotted versus the ℓ_1 norm of the coefficient vector, relative to the norm of the unrestricted least-squares estimate $\tilde{\beta}$. Right: Same for ridge regression, plotted against the relative ℓ_2 norm.
(Source: Hastie et al., 2015)

The x-axis is the factor c , from $|\beta_1|^q + |\beta_2|^q \leq c$, normalised to 1

Lasso: many coef. are exactly zero with low $c \rightarrow$ variable selection

Lasso and Variable Selection

Lasso constraint:
$$\sum_{j=1}^p |\beta_j| \leq c$$

The optimal c for prediction and variable selection are different:

- For **variable selection**, the optimal parameter c shrinks non-zero coefficients toward zero \rightarrow bias
- For **prediction**, the optimal parameter c is often larger than for selection, to reduce the bias on non-zero coefficients
- Lasso selects λ or c by CV, based on MSE \rightarrow for prediction
- **Lasso often includes too many variables** (c is often too large)
- But the true model is very likely a subset of these variables

The one standard error rule

Breiman et al. (1984) proposed a rule-of-thumb:¹⁹

- Lasso selects λ by CV, based on MSE \rightarrow for prediction
- Consider values of λ within a 1-standard error interval
- Pick the largest value of λ within this interval (smallest c)

The main idea of the 1 SE rule is to choose the most parsimonious model whose accuracy is comparable with the best model

It is a rule-of-thumb, expected to provide a value of λ in between the optimal one for prediction and the optimal one for selection

¹⁹Breiman, Friedman, Stone, Olshen (1984) Classification and Regression Trees

Simulation results with uncorrelated covariates

- Linear regression model with many covariates, $n = 1000$
- Monte Carlo simulation with 1000 replications
- $\hat{\lambda}^{min}$ is selected by CV, $\hat{\lambda}^{1se}$ with the 1 SE rule
- Potency: proportion of relevant variables selected
- Gauge: proportion of irrelevant variables selected

Table 7: Comparison of potency, gauge and MSE under linear effects and uncorrelated covariates

Model	Rule of thumb	Potency	Gauge	MSE
LASSO	$\hat{\lambda}^{min}$	1	0.299	1.033
	$\hat{\lambda}^{1se}$	1	0.032	1.073

→ Lasso with $\hat{\lambda}^{min}$ selects 29.9% of irrelevant variables

→ Lasso with $\hat{\lambda}^{1se}$ selects 3.2% of irrelevant variables, but MSE ↗

Simulation results with correlated covariates

- Linear regression model with many covariates, $n = 1000$
- Monte Carlo simulation with 1000 replications
- $\hat{\lambda}^{min}$ is selected by CV, $\hat{\lambda}^{1se}$ with the 1 SE rule
- Potency: proportion of relevant variables selected
- Gauge: proportion of irrelevant variables selected

Table 6: Comparison of potency, gauge and MSE under linear effects and correlated covariates

Model	Rule of thumb	Potency	Gauge	MSE
LASSO	$\hat{\lambda}^{min}$	1	0.643	1.060
	$\hat{\lambda}^{1se}$	0.992	0.458	1.104

→ Lasso with $\hat{\lambda}^{min}$ selects 64.3% of irrelevant variables

→ Lasso with $\hat{\lambda}^{1se}$ selects 45.8% of irrelevant variables, MSE ↗

Adaptive Lasso

The Adaptive Lasso is based on the following constraint:²⁰

$$\sum_{j=1}^p w_j |\beta_j| \leq c \quad \text{where} \quad w_j = 1/|\hat{\beta}_j|^\nu$$

where $\hat{\beta}_j$ is the OLS estimate and $\nu > 0$.

- Put smaller weights to larger coefficients in the constraint
- Large non-zero coefficients shrink more slowly to zero as $c \searrow$
- This leads to the *oracle property*, simultaneously achieving
 - Consistent variable selection
 - Optimal estimation prediction
- ν is often set equal to 1, but it could be selected by CV

²⁰Zou (2006), JASA, 101 1418-1429

Elastic-net

The Elastic-net is based on the following constraint:²¹

$$\sum_{j=1}^p (r\beta_j^2 + (1-r)|\beta_j|) \leq c$$

where $r = 1$ corresponds to the Ridge and $r = 0$ to the Lasso.

- Lasso may perform poorly with highly correlated covariates, which is often encountered in high-dimensional data analysis
- By combining a ℓ_2 -penalty with the ℓ_1 -penalty, we obtain a method that deals better with such correlated groups, and tends to select the correlated covariates (or not) together.
- Like Lasso, Elastic-net often includes too many covariates

²¹Zou and Hastie (2005), JRSS Series B, 67 301-320. It corresponds to the penalization $\lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|$, where λ_1 and λ_2 are selected by CV.

Adaptive Elastic-net

The Adaptive Elastic-net is based on the following constraint:²²

$$\sum_{j=1}^p \{ r\beta_j^2 + (1-r)w_j|\beta_j| \} \leq c$$

where $w_j = 1/(|\hat{\beta}_j| + \frac{1}{n})^\nu$ and $\nu > 0$.²³

- Adaptive Lasso has oracle property (consistent variable selection), but inherits the instability of Lasso for high-dimensional data
- Elastic-net deals better in high-dimensional data analysis, but it lacks the oracle property
- Adaptive Elastic-net combines the two approaches

²²Zou and Zhang (2009) Annals of Statistics, 37, 1733-1751. It remains to the penalization $\lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p w_j|\beta_j|$, where λ_1, λ_2 are selected by CV

²³1/n in w_j is used to avoid division by 0

Application: Predict baseball player's Salary

- What predictors are associated with baseball player's Salary?

Salary – 1987 annual salary on opening day in thousands of dollars;

Years – Number of years in the major leagues;

Hits – Number of hits in 1986;

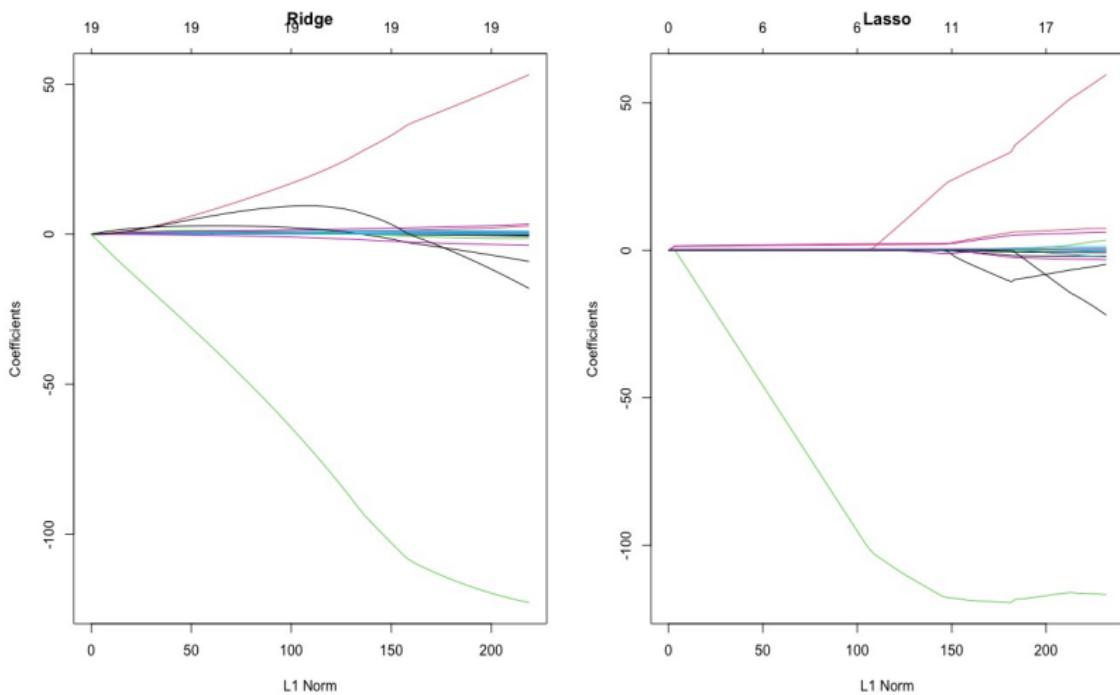
Atbat – Number of times at bat in 1986;

...

```
1 library(ISLR)
2 Hitters=na.omit(Hitters)
3 x=model.matrix(Salary~.,Hitters)[,-1]
4 y=Hitters$Salary
5 # Ridge and Lasso
6 library(glmnet)
7 ridge.model=glmnet(x,y,alpha=0)
8 lasso.model=glmnet(x,y,alpha=1)
9 par(mfrow=c(1,2))
10 plot(ridge.model, main="Ridge")
11 plot(lasso.model, main="Lasso")
```

By default, the covariates are standardized, otherwise use the argument
standardize=FALSE in the function `glmnet`

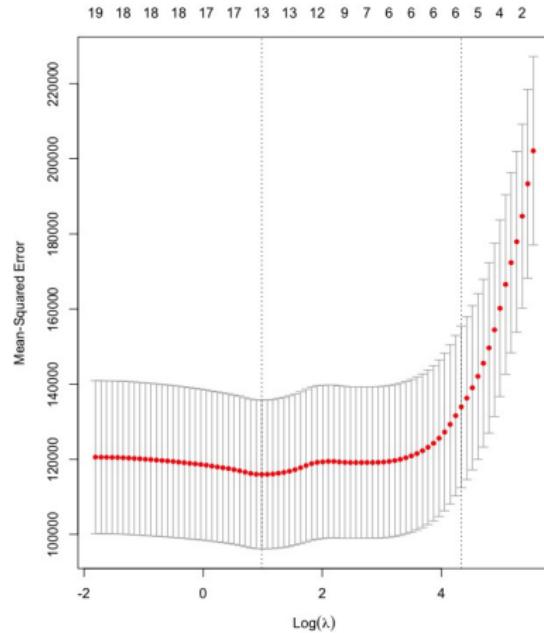
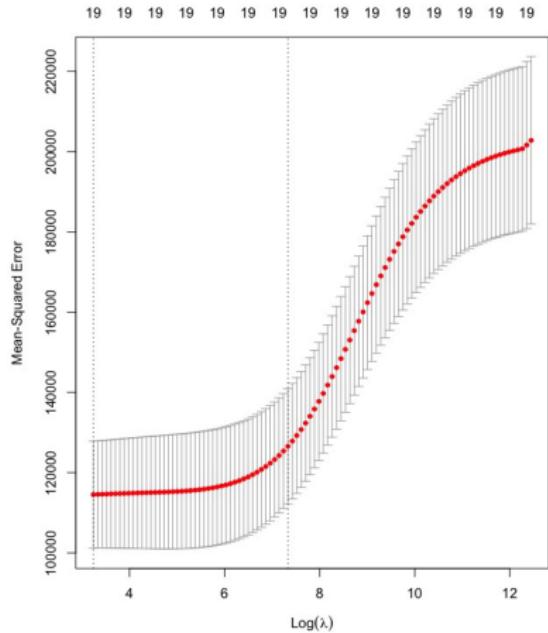
Application: Coefficient paths



Coefficient paths for Ridge and Lasso as c increases

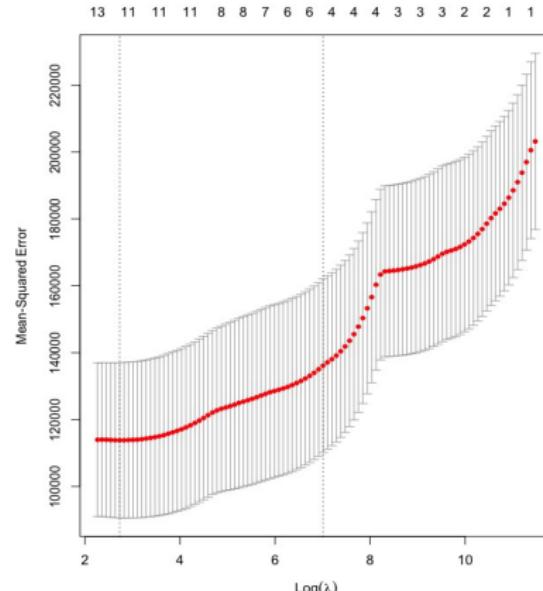
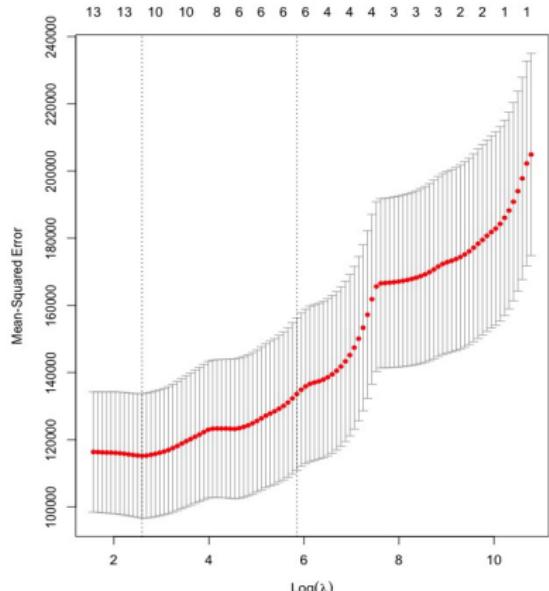
Application: Cross Validation

```
1 cv.ridge=cv.glmnet(x,y, alpha=0)
2 cv.lasso=cv.glmnet(x,y, alpha=1)
3 plot(cv.ridge, main="Ridge")
4 plot(cv.lasso, main="Lasso")
```



Application: Adaptive Lasso and Adaptive Elastic-net

```
1 ols=lm(Salary ~ ., Hitters)
2 w=1/abs(coef(ols))
3 cv.adalasso <- cv.glmnet(x,y,alpha=1,penalty.factor=w)
4 cv.adaelast <- cv.glmnet(x,y,alpha=.5,penalty.factor=w)
5 plot(cv.adalasso, main="Adaptive Lasso")
6 plot(cv.adaelast, main="Adaptive Elastic-net")
```



Application: Compare the coefficients

```
1 coef1=coef(ols)
2 coef2=coef(cv.ridge, s="lambda.min")
3 coef3=coef(cv.lasso, s="lambda.min")
4 coef4=coef(cv.lasso, s="lambda.1se")
5 coef5=coef(cv.adalasso, s="lambda.min")
6 coef6=coef(cv.adaelast, s="lambda.min")
7 options(scipen = 999) # disable scientific notation
8 coeff=cbind(coef1,coef2,coef3,coef4, coef5, coef6)
9 colnames(coeff) <- c("ols","ridge","lasso","lasso.1se",
10           "adaLasso", "adaElastic")
11 coeff
```

Application: Compare the coefficients

	ols	ridge	lasso	lasso.1se	adaLasso	adaElastic
(Intercept)	163.1035878	81.126931475	123.7520754	144.37970485	126.57066439	132.25254600
AtBat	-1.9798729	-0.681595884	-1.5473426	.	-1.48615273	-1.56187585
Hits	7.5007675	2.772311573	5.6608972	1.36380384	5.88349361	5.96271593
HmRun	4.3308829	-1.365680118	.	.	.	-0.03269556
Runs	-2.3762100	1.014826485	.	.	0.89821264	0.91521857
RBI	-1.0449620	0.713022451
Walks	6.2312863	3.378557588	4.7296908	1.49731098	2.79140518	3.24066883
Years	-3.4890543	-9.066800376	-9.5958374	.	-12.36078333	-13.89945521
CAtBat	-0.1713405	-0.001199478	.	.	0.03339294	0.04086003
CHits	0.1339910	0.136102881
CHmRun	-0.1728611	0.697995815	0.5108207	.	.	.
CRuns	1.4543049	0.295889601	0.6594856	0.15275165	.	.
CRBI	0.8077088	0.257071062	0.3927505	0.32833941	0.57889147	0.59178814
CWalks	-0.8115709	-0.278966594	-0.5291586	.	.	-0.06028053
LeagueN	62.5994230	53.212720264	32.0650811	.	.	13.41281263
DivisionW	-116.8492456	-122.834451470	-119.2990171	.	-127.54437276	-127.92668761
PutOuts	0.2818925	0.263887567	0.2724045	0.06625755	0.26875502	0.26560908
Assists	0.3710692	0.169879574	0.1732025	.	.	.
Errors	-3.3607605	-3.685645334	-2.0585083	.	.	.
NewLeagueN	-24.7623251	-18.105095858	.	.	17.44998998	6.53776837

- Shrinkage methods: the coefficients are shrunk towards zero
- Variable selection is sensitive to the method

Summary

- Ridge and Lasso can be used in high-dimension ($p \gg n$)
- They are based on a bias-variance tradeoff
- Tradeoff selected minimizing MSE out-sample by CV
- Sparse models: Lasso is a variable selection method
- Ridge puts similar coefficients to strongly correlated variables, while Lasso selects one randomly
- Extension to adaptive Lasso and Elastic-net

2. Methods and Algorithms

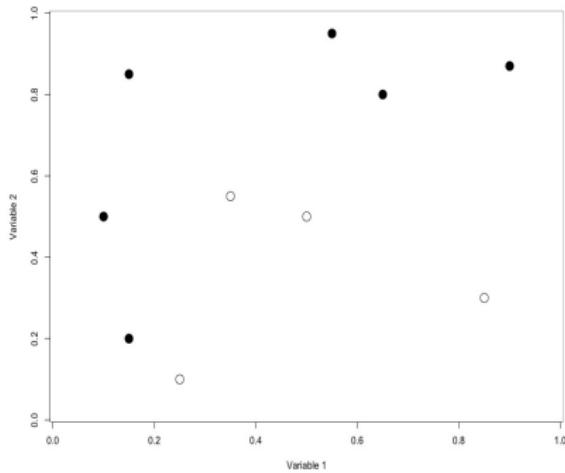
- Ridge and Lasso Regression
- Classification and Regression Tree
- Bagging and Random Forests
- Boosting
- Support Vector Machine
- Neural Networks and Deep Learning

Classification Tree

$y \in \{0, 1\}$ is a qualitative variable

Classification Tree: Principle from a small sample

Find the best rule on a single variable to classify black/white balls

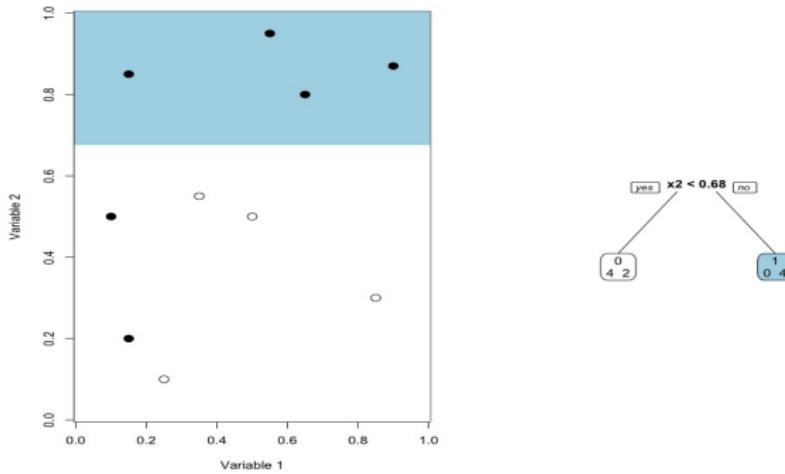


→ find a cutoff on x_1 or x_2 such that the maximum number of observations is correctly classified²⁴

²⁴See <https://freakonometrics.hypotheses.org/52776>

Classification Tree: graphical representation

Minimizing misclassification, we find $x_2 < k$, where $k \in (0.56, 0.8)$

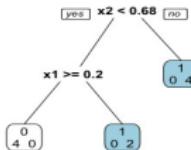
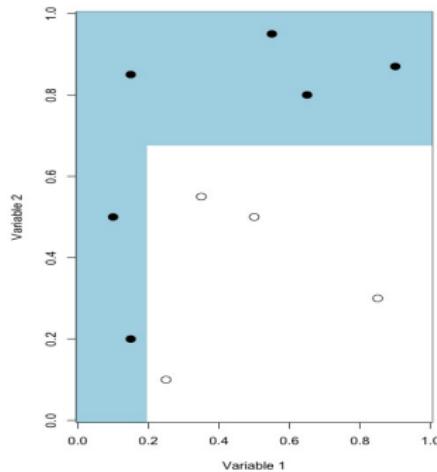


This Figure represents the best split in a competition between all possible splits of x_1 and x_2 .

From this simple rule, two bullets are misclassified . . . we can try to find a new rule in the white area sub-group . . .

Classification Tree: A sum of simple rules

The additional rule $x_1 \geq c$, where $c \in (.16, .2)$, produces the best subsequent split:

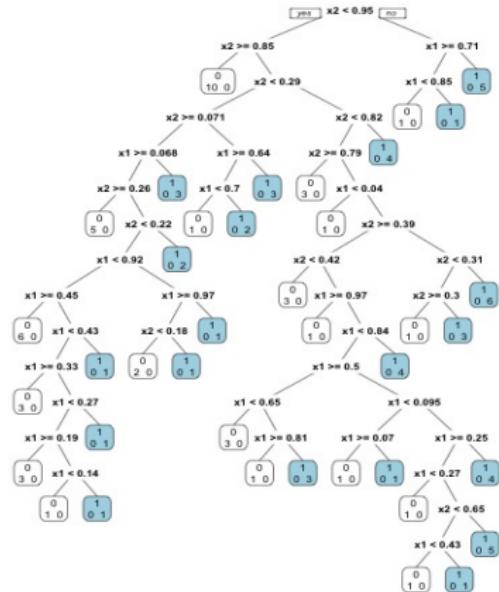
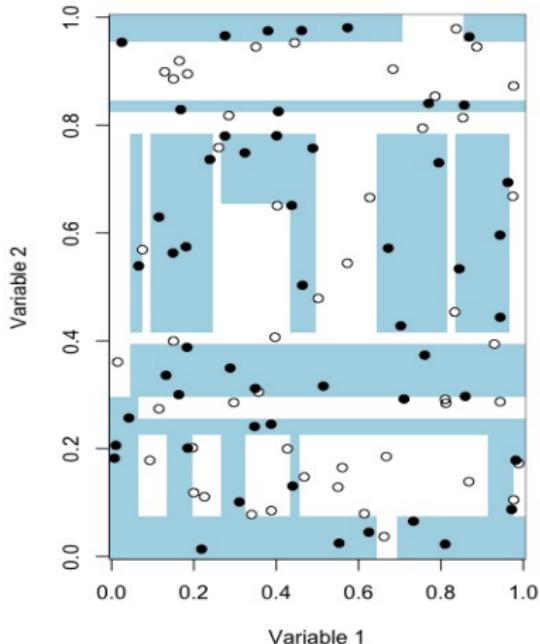


Using these two rules, all bullets are correctly classified

Classification Tree: Extension to large sample

- Interpretation is quite easy and intuitive
- We use **recursive binary splitting** to grow a tree
- A tree can grow until every observations is correctly classified
- With a large sample, a tree may have many **nodes**, that is, many points where the predictor is splitted into two **leaves**
- Note that this principle is easy to apply, even with several regressors and several classes

Classification Tree: Example with 100 observations



The resulting tree is quite complex and not so easy to interpret

Classification: Tree pruning

An **unpruned tree**:

- classifies correctly every observation from a training sample
- may classify poorly observations from a test sample
 - (it is the standard problem of overfitting)
- maybe difficult to interpret

A **pruned tree**:

- is a smaller tree with fewer splits
 - might perform better on a test sample
 - might have an easier interpretation
- define a criterion for making binary splits

Classification: Tree pruning

- A fully grown tree fits perfectly training data, poorly test data
- Tree pruning is used to control the problem of overfitting
- A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of a little bias
- Poor strategy: Add new split only if it is worthwhile to do so
- However, a poor split could be followed by a very good split
- Good strategy: Grow a very large tree and prune it back in order to obtain a subtree, keep a split only if it is worthwhile
→ We need to define what "only if it is worthwhile" means

Classification tree: Gini impurity index

- Classification: not only concerned by class prediction, also by class proportion → Min impurity rather than misclassification
- Gini impurity index at some node \mathcal{N} :²⁵

$$G(\mathcal{N}) = \sum_{k=1}^K p_k(1 - p_k) = 1 - \sum_{k=1}^K p_k^2$$

with p_k the fraction of items labeled with class k in the node

- Node: 100-0 or 0-100 → minimal impurity/diversity: $G = 0$,
Node: 50-50 → maximal impurity/diversity: $G = 1/4$.²⁶
- A small value means that a node contains predominantly observations from a single class (homogeneity)

²⁵Another index is the Entropy “impurity” index $E(\mathcal{N}) = -\sum_{k=1}^K p_k \log p_k$

²⁶With 100-0: $p_1 = 1, p_2 = 0$; 0-100, $p_1 = 0, p_2 = 1$; 50-50: $p_1 = p_2 = 1/2$

Classification: Tree pruning

- If we split the node into two leaves, \mathcal{N}_L and \mathcal{N}_R , the Gini impurity index becomes

$$G(\mathcal{N}_L, \mathcal{N}_R) = p_L G(\mathcal{N}_L) + p_R G(\mathcal{N}_R)$$

where p_L, p_R are the proportion of observations in $\mathcal{N}_L, \mathcal{N}_R$

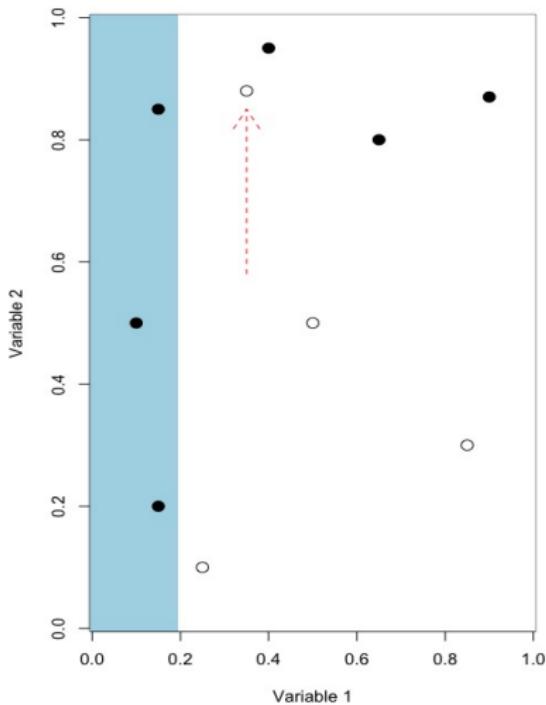
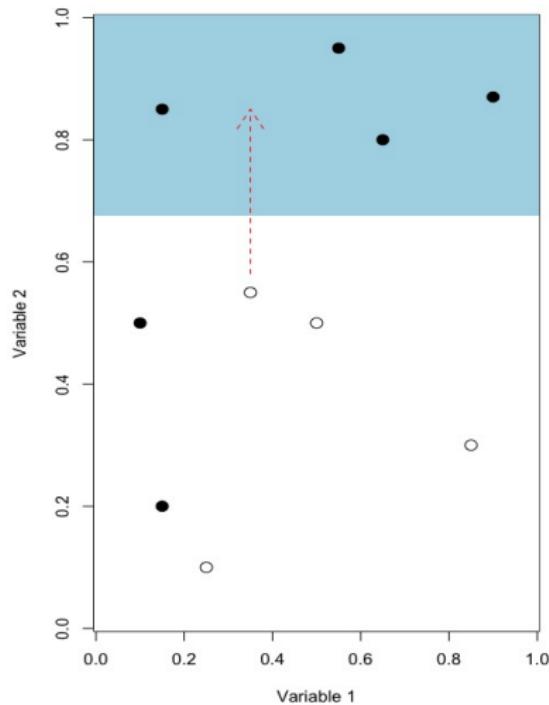
- When do we split? ... when impurity is reduced substantially:

$$\Delta = G(\mathcal{N}) - G(\mathcal{N}_L, \mathcal{N}_R) > \epsilon$$

we can also require a minimum of observations per node

- How do we split? ... find the cutoff on a single variable that minimise impurity rather than misclassification ($\rightarrow \max \Delta$)

Classification tree: Limitation



Small change in the original sample \Rightarrow Tree may differ significantly

Application: Predict survival on the Titanic

- Consider passenger data from the sinking of the Titanic
- What predictors are associated with those who perished compared to those who survived?

`survived` – 1 if true, 0 otherwise;

`sex` – the gender of the passenger;

`age` – age of the passenger in years;

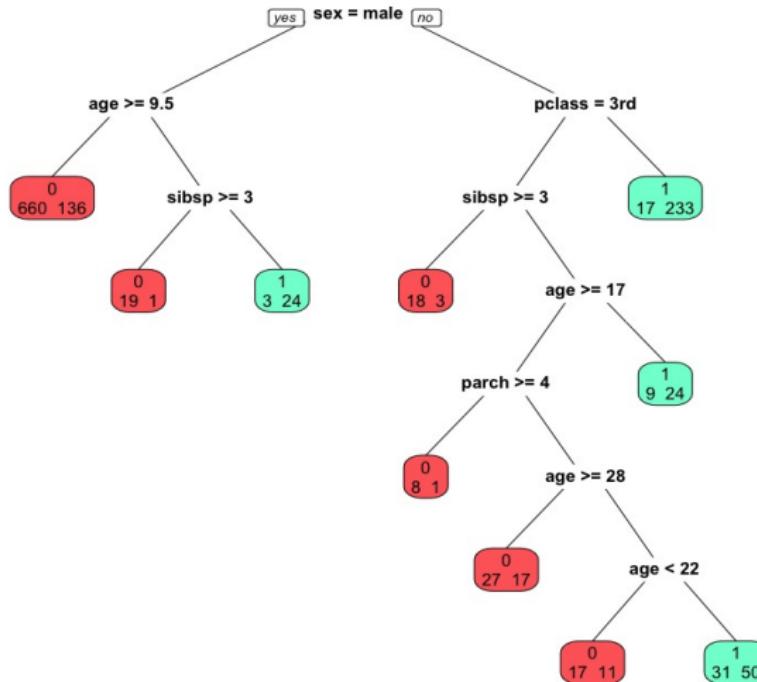
`pclass` – the passenger's class of passage;

`sibsp` – the number of siblings/spouses aboard;

`parch` – the number of parents/children aboard.

```
1 library(PASWR)      # get the data
2 data(titanic3)
3 library(rpart)       # CART package
4 library(rpart.plot) # fancy plots
5 X=cbind(sex,age,pclass,sibsp,parch)
6 tree <- rpart(survived~X, data=titanic3, method="class")
7 prp(tree, extra=1, faclen=5, box.col=c("indianred1",
  "aquamarine"))[tree$frame$yval])
```

Application: Titanic classification tree



Application: Variable importance

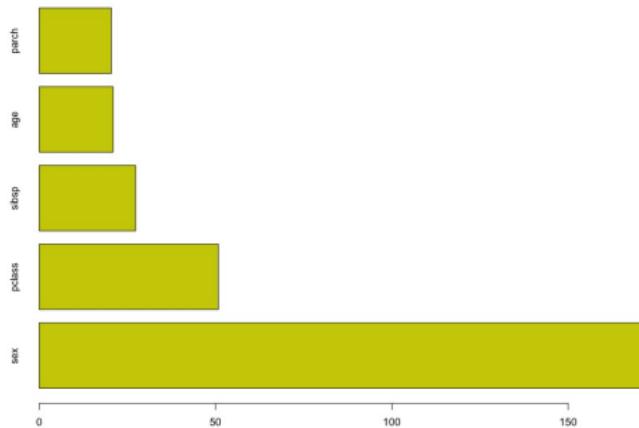
The importance of each variable, related to the gain in Gini, is

```
1 tree$variable.importance
```

```
1 sex      pclass     sibsp      age      parch  
2 172.74924  50.78568  27.33127  20.95528  20.46938
```

that we can plot using

```
1 barplot(tree$variable.importance, horiz=TRUE, col="yellow3")
```

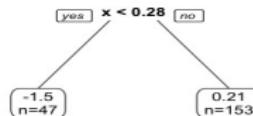
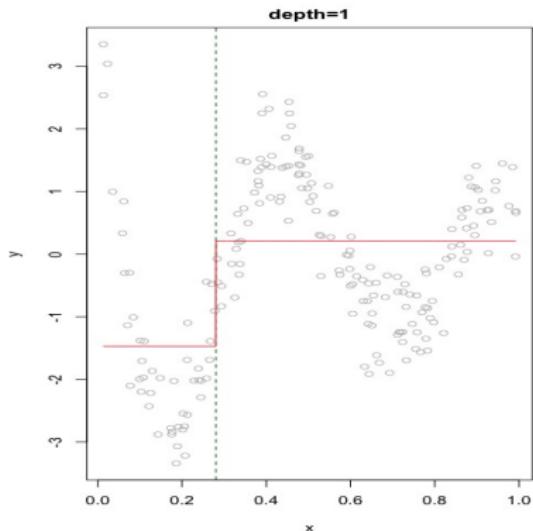


Regression Tree

$y \in \mathbb{R}$ is a quantitative variable

Regression Tree: Principle with one covariate

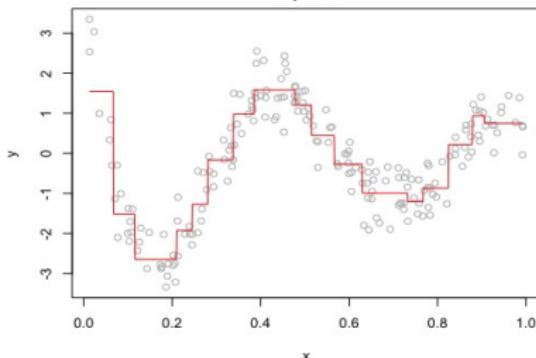
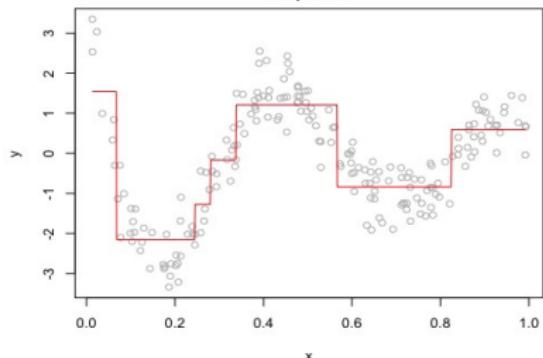
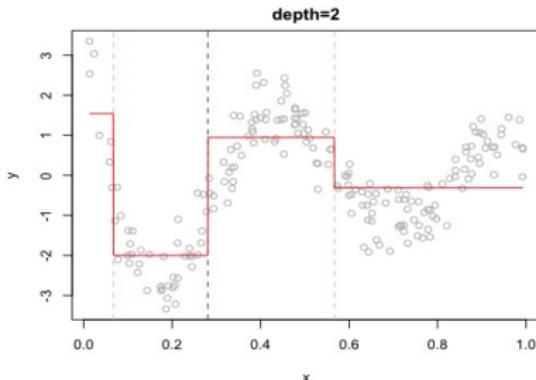
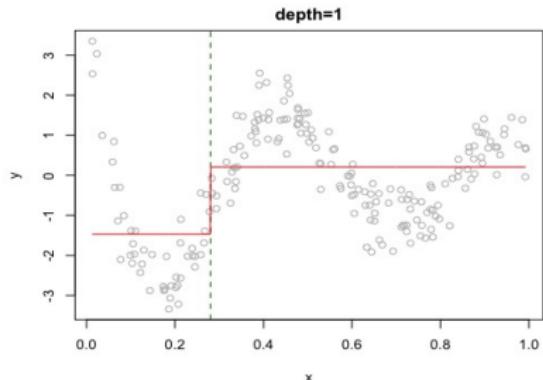
Find the best split, which minimizes deviations to the mean (variances) in each leaf:



→ find cutoff on x such that: $\text{Min} \sum_{x_i \in \mathcal{N}_L} (y_i - \bar{y}_L)^2 + \sum_{x_i \in \mathcal{N}_R} (y_i - \bar{y}_R)^2$

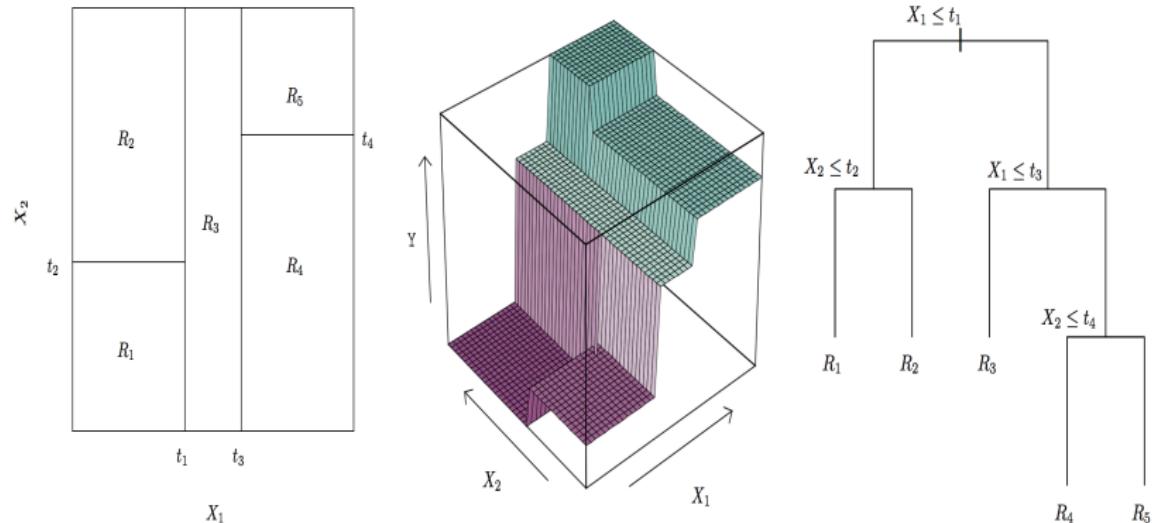
Regression Tree: Principle with one covariate

Then use recursive binary splitting:



Regression Tree: Principle with two covariates

With two covariates, $y \approx m(x_1, x_2)$, we have:



(Source: James et al., 2013)

Find boxes R_1, \dots, R_J that minimize SSR:¹ $\text{Min } \sum_{j=1}^J \sum_{x_i \in R_j} (y_i - \bar{y}_{R_j})^2$

¹We cannot consider every possible partition → use recursive binary splitting

Application: Predict baseball player's Salary

- Let's consider 3 covariates: $y \approx m(x_1, x_2, x_3)$
- What predictors are associated with baseball player's Salary?

Salary – 1987 annual salary on opening day in thousands of dollars;

Years – Number of years in the major leagues;

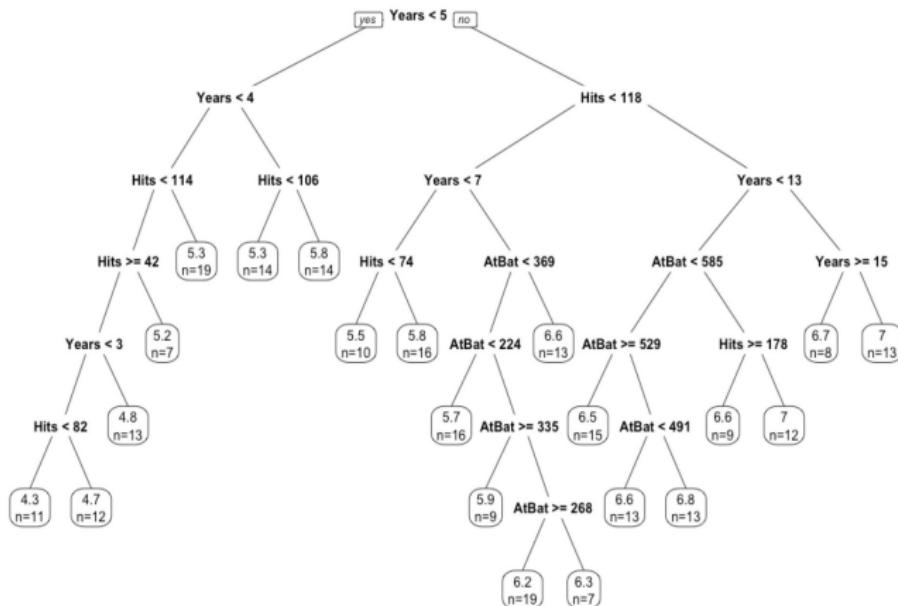
Hits – Number of hits in 1986;

Atbat – Number of times at bat in 1986;

```
1 library(ISLR)
2 # remove observations that are missing Salary values
3 df=Hitters[complete.cases(Hitters$Salary),]
4 # load CART library
5 library(rpart)
6 library(rpart.plot)
7 # estimate the tree
8 tree <- rpart(log(Salary)~Years+Hits+AtBat, data=df, cp=0)
9 # plot the tree
10 prp(tree, extra=1, faclen=5)
```

Regression Tree: Principle with several covariates

With more covariates, we can only use the decision tree figure:



based on the same principle: find terminal nodes that min SSR

Regression: Tree pruning

- A fully grown tree fits perfectly training data, poorly test data
- Tree pruning is used to control the problem of overfitting
- A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of a little bias
- Poor strategy: Build the tree only so long as the decrease in the SSR due to each split exceed some threshold
- However, a poor split could be followed by a very good split
- Good strategy: Grow a very large tree and prune it back in order to obtain a subtree, keep a split only if it is worthwhile

Regression: Tree Pruning

- **Penalization:** we put a prize to pay for having a tree with many terminal nodes J , or regions,

$$\text{Min} \sum_{j=1}^J \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2 + \lambda J$$

For given λ , we can find the subtree minimizing this criterion²⁷

- **Cross-validation:** we select λ using cross validation
- A smaller tree with **fewer splits** might lead to **lower variance** and better interpretation at the cost of a **little bias**

²⁷ λ is called the *complexity parameter*

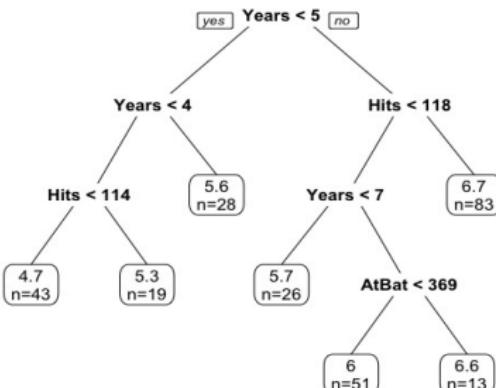
Tree pruning: Application

```
1 tree <- rpart(log(Salary) ~ Years+Hits+AtBat, data=df) # CV  
2 prp(tree, extra=1, faclen=5)
```

Unpruned tree



Pruned tree

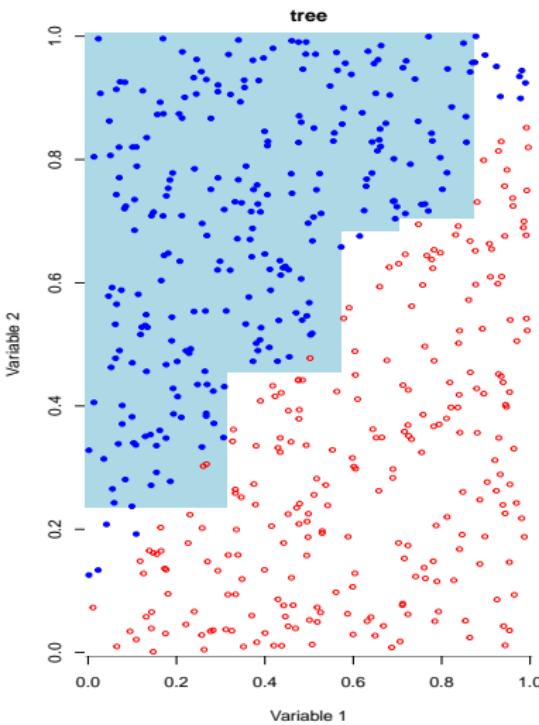
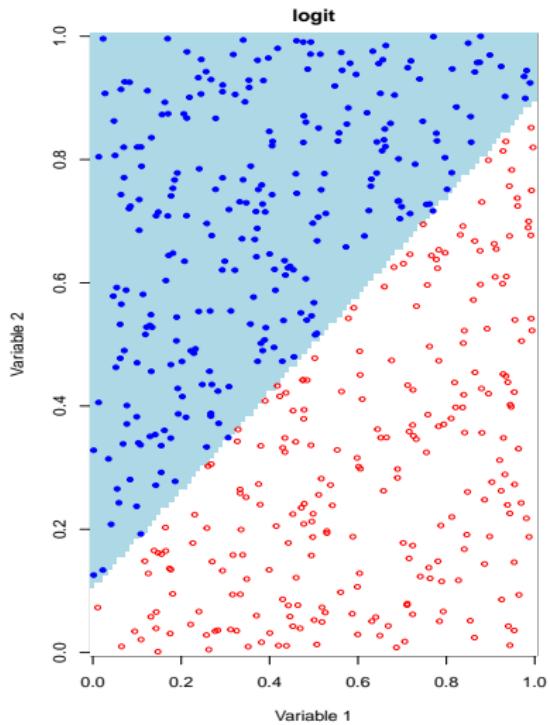


Tree versus linear model

Tree vs. linear model: Which model is better?

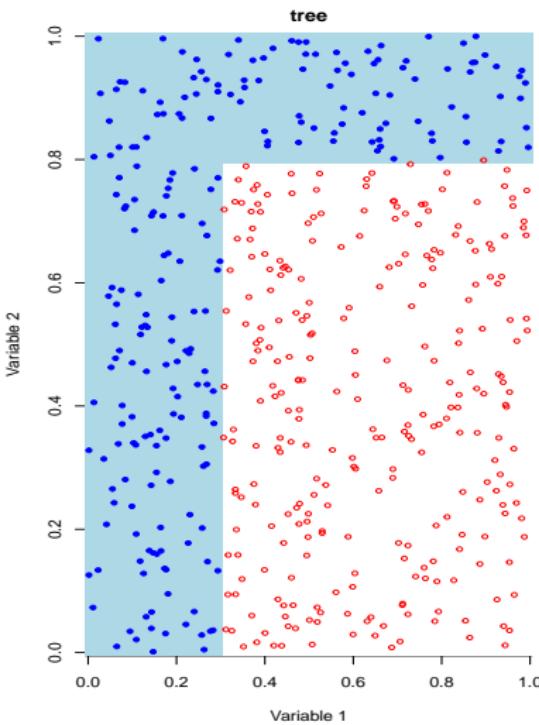
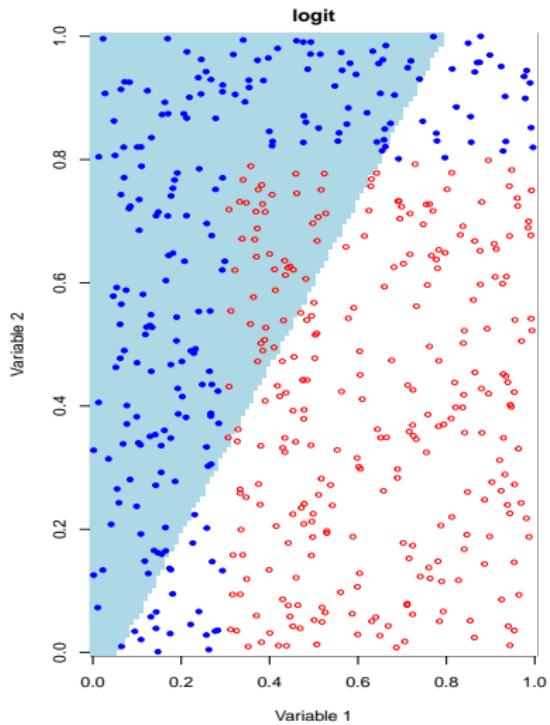
- It depends on the problem at hand:
 - Linear regression: $m(X) = \beta_0 + \sum_{j=1}^K X_j \beta_j$
 - Regression tree: $m(X) = \sum_{j=1}^J c_j \mathbb{1}(x \in R_j)$
- If the relationship between y and x_1, \dots, x_K is linear: a linear model should perform better
- If the relationship between y and x_1, \dots, x_K is highly non-linear and complex: a tree model should perform better

True decision boundary: linear



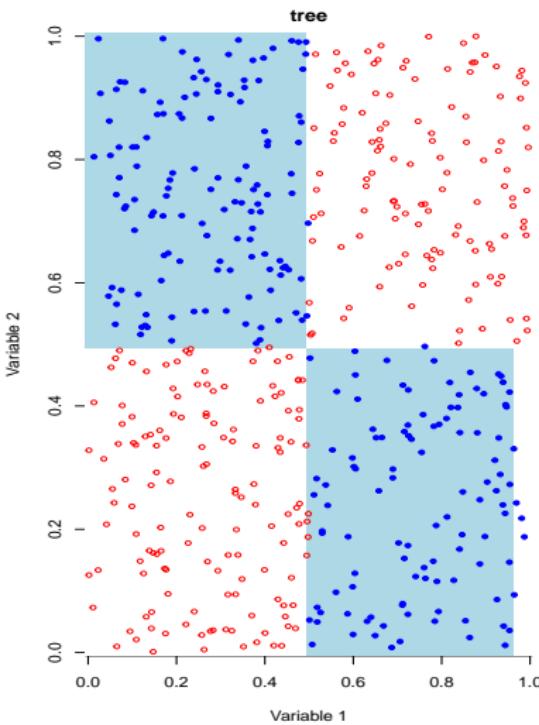
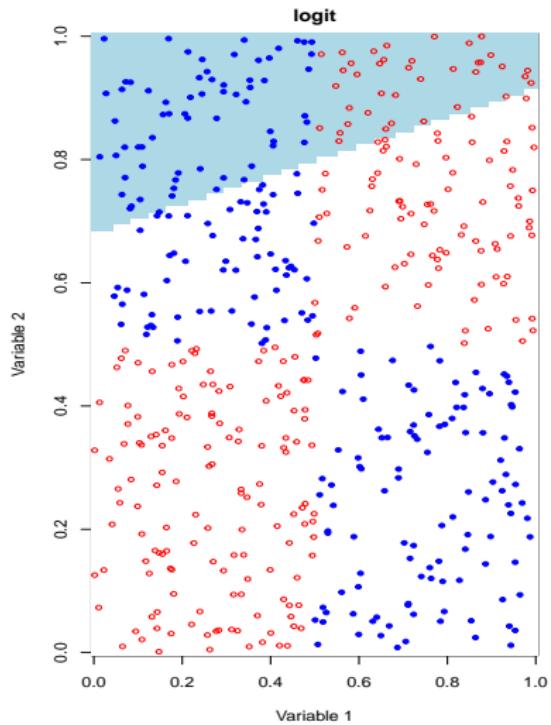
Blue area \equiv fitted values in blue from linear (left) and tree (right) models

True decision boundary: nonlinear



Blue area \equiv fitted values in blue from linear (left) and tree (right) models

True decision boundary: interactions



Blue area \equiv fitted values in blue from linear (left) and tree (right) models

Classification And Regression Tree (CART)

Advantages:

- Trees tend to work well for problems where there are important nonlinearities and interactions
- The results are really intuitive and can be understood even by people with no experience in the field

Disadvantage:

- Trees are quite sensitive to the original sample (non-robust)
- They may have poor predictive accuracy out-sample

2. Methods and Algorithms

- Ridge and Lasso regression
- Classification and Regression Tree
- **Bagging and Random Forests**
- Boosting
- Support Vector Machine
- Neural Networks and Deep Learning

Bagging and Random Forest

How bagging and random forest work intuitively:

- Based on your symptoms, suppose a doctor diagnoses an illness that requires surgery
 - Instead asking one doctor, you may choose to ask several
 - If one diagnosis occurs more than any others, you may choose this one as the final diagnosis
- the final diagnosis is made based on a majority vote of doctors

Bagging and Random Forest: replace doctors by bootstrap samples

Bagging: algorithm

Algorithm 1: Bagging

Select number of trees B , and tree depth D ;

for $b = 1$ to B **do**

generate a bootstrap sample from the original data;

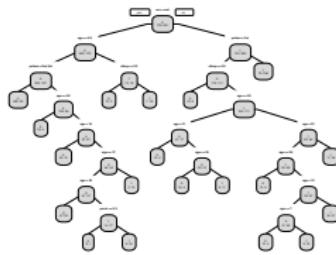
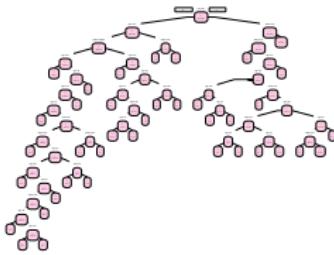
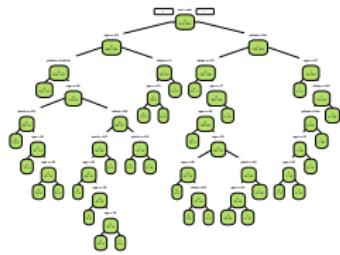
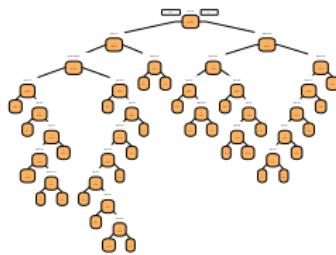
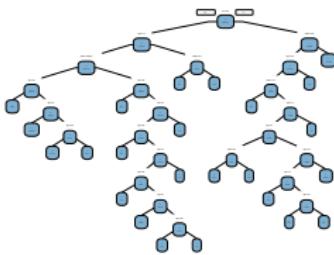
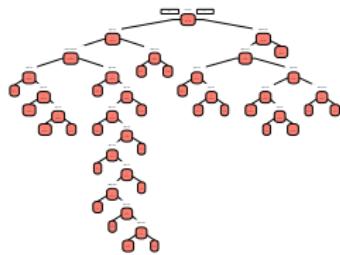
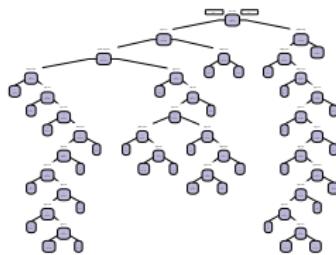
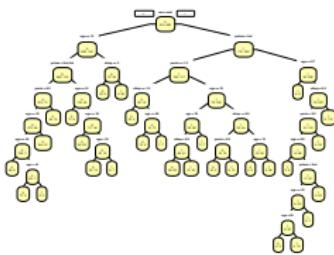
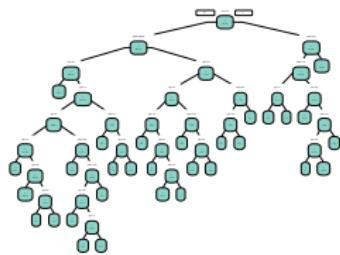
estimate a tree model of depth D on this sample;

end

For instance, with the titanic dataset:

```
1 library(rpart) ; library(rpart.plot)
2 library(PASWR) ; data(titanic3)
3 n = NROW(titanic3$survived)
4 par(mfrow=c(3,3))
5 for(i in 1:9){
6   idx = sample(1:n, size=n, replace=TRUE)
7   cart = rpart(as.factor(survived)~sex+age+pclass+sibsp+
8     parch, data=titanic3[idx,], cp=0) # unpruned
9   prp(cart, type=1, extra=1)}
```

Bagging: Generate several trees by bootstrapping



Bagging: Why bootstrapping CART model?

Bagging = Bootstrap aggregating

Prediction:

- Regression: average the resulting predictions
- Classification: take a majority vote

Impact of bootstrapping:

- Averaging a set of observations reduces variance²⁸
- It reduces variance and hence increase the prediction accuracy
- Compared to CART, the results are much less sensitive to the original sample, they show impressive improvement in accuracy
- Loss of interpretability

²⁸The variance of the mean of the observations \bar{X} is given by σ^2/n

Random Forest: algorithm

Algorithm 2: Random Forest

Select number of trees B , subsampling parameter m , tree depth D ;

for $b = 1$ to B **do**

generate a bootstrap sample from the original data;
estimate a tree model on this sample;

for each split do

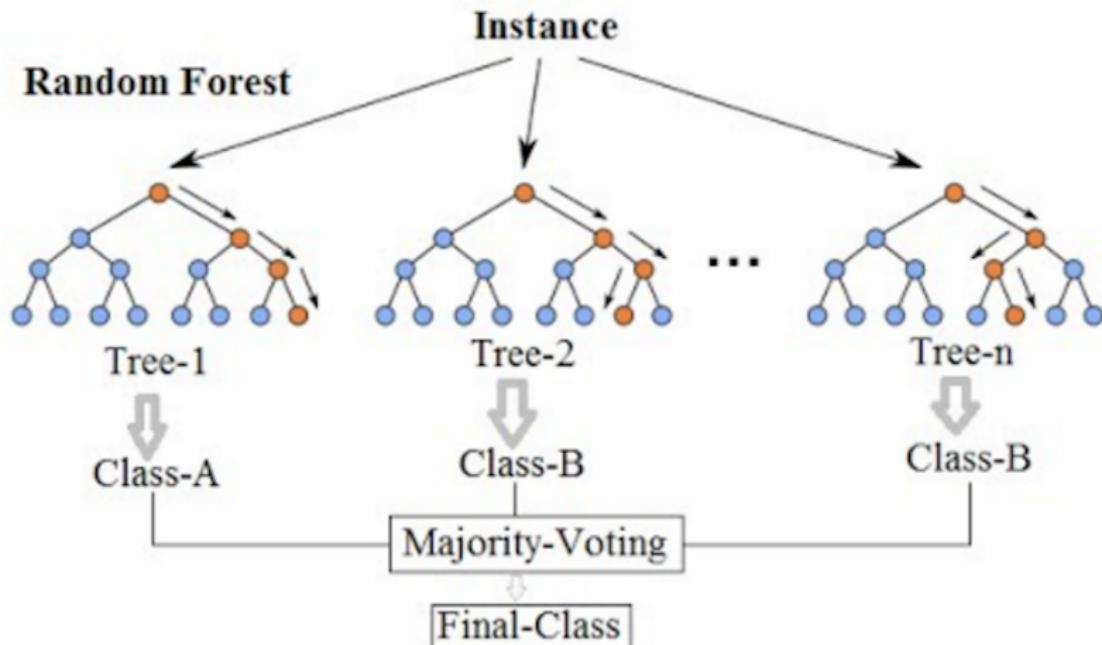
Randomly select m of the original covariates ($m < P$);
Split the data with the best covariate (among the m);

end

end

- Random Forest = Bagging + subsample covariates at each node
- Bagging is a special case of Random Forest, with $m = P$

Bagging and Random Forest



Random Forest = Bagging + subsampling covariates at each node

Random forest: Why subsampling covariates?

Subsampling covariates may sound crazy, it has clever rationale:

- Suppose there is one very strong covariate in the sample
 - Most or all trees will use this covariate in the top split
 - All of the trees will look quite similar to each other
 - Hence the predictions will be highly correlated
- Averaging many highly correlated quantities does not lead to a large reduction in variance
- Random forests overcome this problem by forcing each split to consider only a subset of the covariates

→ Random forests **decorrelate** the trees

In practice, default values: $m = p/3$ in regression and $m = \sqrt{p}$ in classification

Random forest: Overfitting

There is not much overfitting in random forests . . .

- as B increases: average effect over trees \rightarrow no overfitting
- as D increases: overfitting is argued to be minor

"Segal (2004) demonstrates small gains in performance by controlling the depths of the individual trees grown in random forests. Our experience is that using full-grown trees seldom costs much, and results in one less tuning parameter. Figure 15.8 shows the modest effect of depth control in a simple regression example." (Hastie et al., 2009, p.596)

The goal is to grow trees with as little bias as possible. The high variance that would result from deep trees is tolerated because of the averaging over a large number of trees

. . . However, a simple example shows that it can be problematic

Random forest: Overfitting . . . a simple example

Let us consider a realistic (simulated) sample

```
1 set.seed(1)
2 n=200
3 x=runif(n)
4 y=sin(12*(x+.2))/(x+.2) + rnorm(n)/2
```

We can fit CART and random forest models:²⁹

```
5 fit.tr <- rpart(y~x) # CART
6 fit.ba1 <- randomForest(y~x) # no depth control
7 fit.ba2 <- randomForest(y~x,maxnodes=20) # depth control
```

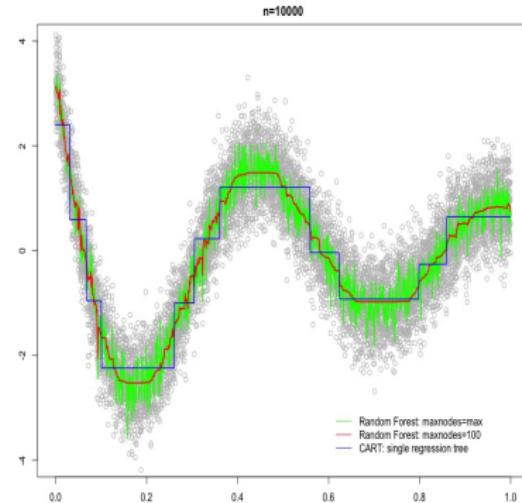
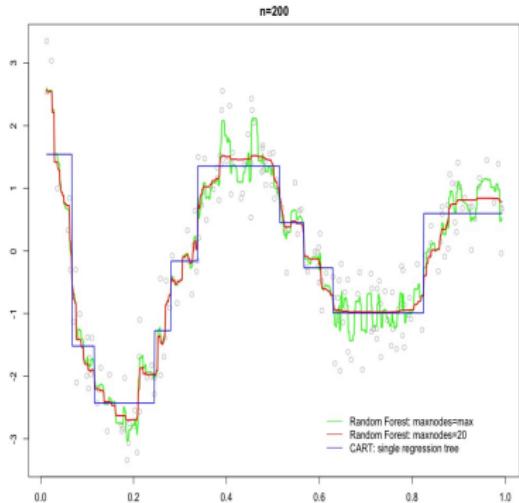
We can plot observations and predicted values:

```
8 u=seq(min(x),max(x),length.out=1000)
9 plot(x,y,col="gray",main="n=200")
10 lines(u,predict(fit.ba1,data.frame(x=u)), col="green")
11 lines(u,predict(fit.ba2,data.frame(x=u)), col="red")
12 lines(u,predict(fit.tr,data.frame(x=u)), col="blue")
```

We run this code for $n = 200$ and $n = 10000$

²⁹Note that since it is a simple regression, with 1 covariate, then RF=bagging

Random forest: Overfitting . . . a simple example



- improvement of random forest over a single regression tree
- overfitting can be very large without controlling tree depth

Random forest: Out-of-bag (OOB) observations

No need to perform cross-validation:

- By bootstrapping, each tree uses around $2/3$ of the obs. The remaining $1/3$ obs are referred to as the out-of-bag (OOB) obs
- Use OOB observations for out-sample predictions
- We obtain around $B/3$ out-sample predictions for the i^{th} obs.
- average these values (or majority vote) = OOB prediction for i
- An OOB-MSE can be computed over all OOB predictions

The OOB approach for estimating the test error is particularly convenient with large sample, for which CV would be onerous

Random forest: Tuning parameters

We can use OOB-MSE to tune Random Forest parameters

- Depth tree, D : from our previous example, with $n = 10000$

```
1 >randomForest(y~x)$mse[500] # OOB-MSE, no depth control
2 [1] 0.3252183

3 >maxnode=c(10,50,100,500,1000,2000)
4 >for (i in 1:NROW(maxnode)){ # OOB-MSE, depth control
5 > aa=randomForest(y~x,maxnodes=maxnode[i])$mse[500];
6 > print(c(maxnode[i],aa))}
7 [1] 10.0000000 0.3747725
8 [1] 50.0000000 0.2553131
9 [1] 100.0000000 0.2508479
10 [1] 500.0000000 0.2570217
11 [1] 1000.000000 0.268357
12 [1] 2000.000000 0.2921307
```

We can see that OOB-MSE is smaller with `maxnode=100`

- Subsampling parameter, m : can be selected similarly

Random forest: Variable importance

- Random forest improves prediction accuracy at the expense of interpretability . . . the resulting model is difficult to interpret
- One can obtain an overall summary of the importance of each covariates using SSR (regression) or Gini index (classification)
- Index: record the total amount that the SSR/Gini is decreased due to splits over a given covariate, averaged over all B trees

```
1 >rf <- randomForest(as.factor(survived)~sex+age+pclass+
  + sibsp+parch, data=titanic3, na.action=na.omit)
2 >importance(rf)
      MeanDecreaseGini
3 
4 sex           133.75916
5 age           63.13448
6 pclass        52.45753
7 sibsp         18.74009
8 parch        17.49320
```

Random Forests compared to Single Trees (CART)

TABLE 2
Test set misclassification error (%)

Data set	Forest	Single tree
Breast cancer	2.9	5.9
Ionosphere	5.5	11.2
Diabetes	24.2	25.3
Glass	22.0	30.4
Soybean	5.7	8.6
Letters	3.4	12.4
Satellite	8.6	14.8
Shuttle $\times 10^3$	7.0	62.0
DNA	3.9	6.2
Digit	6.2	17.1

Source: Breiman (2001)

Bagging and Random Forest

Advantages:

- They tend to work well for problems where there are important nonlinearities and interactions.
- They are robust to the original sample and more efficient than single trees

Disadvantage:

- The results are not intuitive and difficult to interpret.

Bagging and Random forest: Exercise

Consider the dataset used to predict baseball player's salary:

```
1 library(ISLR)
2 df=Hitters[complete.cases(Hitters$Salary),]
```

- Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations
- Perform bagging on the training set for a range of values of the tree depth D , with $B = 1000$ trees. Produce a plot with D on the x -axis and the corresponding test set MSE on the y -axis
- Perform random forest on the training set with $B = 1000$ trees for several values of the subsampling parameter m , and compute the corresponding test set MSEs
- Compare the test MSE of bagging and random forest to the test MSE that results from a CART model
- Which variables appear to be the most important predictors in the random forest model?

2. Resampling-based Methods and Algorithms

- Classification and Regression Tree (CART)
- Bagging and Random Forests
- **Boosting**
- Support Vector Machine
- Neural Networks and Deep Learning

Boosting: Principle

- Like bagging, boosting involves combining a large number of decision trees, but the trees are grown **sequentially**
- Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set:
 - Each tree is fit to the residuals from the previous tree model
 - Each iteration is then focused on **improving** previous **errors**³⁰
 - Each tree is shallow (low depth): "**weak**" **classifier/predictor**³¹
- Boosting combines the outputs of many "**weak**" learners (classifiers, predictors) to produce a powerful "**committee**"

³⁰Each subsequent model pays more attention to the errors from previous models . . . it is a process that learns from past errors

³¹Weak classifier: its error rate is only slightly better than random guessing



Boosting for Regression

$y \in \mathbb{R}$ is a quantitative variable

Boosting: algorithm

Algorithm 3: Boosting for regression trees

Select number of trees B , tree depth D , shrinkage parameter λ ;

Set initial predicted values, $\hat{m}(x) = 0$;

for $b = 1$ to B **do**

Compute the residuals, $r = y - \hat{m}(x)$;

Fit a regression tree $\hat{m}^b(x)$ of depth D to the data (r, x) ;

Update the predicted values: $\hat{m}(x) \leftarrow \hat{m}(x) + \lambda \hat{m}^b(x)$;

end

→ By fitting trees to the residuals, we seek to improve \hat{m} in areas where it does not perform well

Boosting: Overfitting

Number of trees, B :

- The role of each new (sequential) tree is to improve the fit
- Unlike random forests, **boosting** can **overfit if B is too large**³²

Depth of trees, D :

- In CART, fully-grown or deep trees are known to overfit
- Boosting can then overfit **if D is too large**
- Depth tree is usually very small, by default it is often $D = 1$

→ B and D can be selected by cross-validation

³²By averaging over a large number of trees, bagging and random forests reduces variability. Boosting does not average over the trees

Boosting: Shrinkage

Idea behind shrinkage:

- Slow down the boosting process to avoid overfitting . . . scale the contribution of each tree by a factor $0 < \lambda < 1$
 - A smaller λ typically requires more trees B . It allows more and different shaped trees to attack the residuals³³
- Small values of D and λ : by fitting small trees to the residuals, we slowly improve \hat{m} in areas where it does not perform well³⁴
- The boosting approach learns slowly (λ = learning rate)
- Statistical methods that learn slowly tend to perform well

³³Typical values are $\lambda = 0.01$, or $\lambda = 0.001$

³⁴By default, $D = 1$ and $\lambda = 0.1$ in the `gbm` function in R

Boosting: a simple regression example

Let us consider a realistic (simulated) sample

```
1 set.seed(1)
2 n=200
3 x=runif(n)
4 y=sin(12*(x+.2))/(x+.2) + rnorm(n)/2
```

We can fit CART and boosting models:

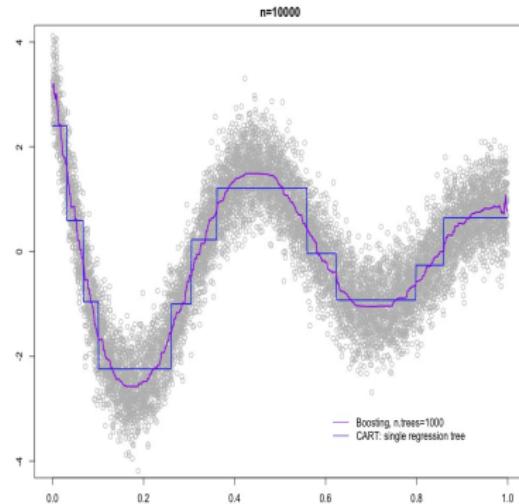
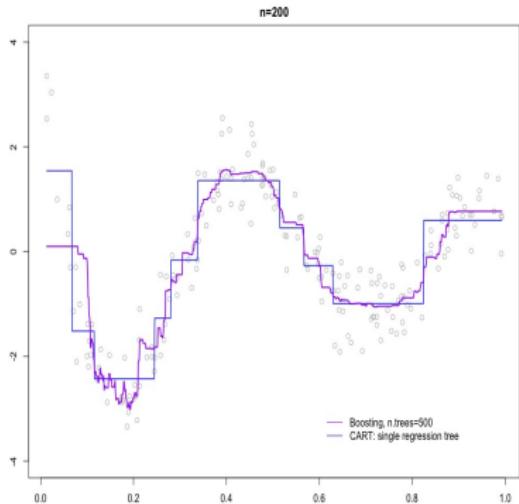
```
5 library(gbm)
6 nb=500
7 # By default: interaction.depth=1 and shrinkage=0.1
8 fit.bo <- gbm(y~x, distribution="gaussian",n.tree=nb)
9 fit.tr <- rpart(y~x)
```

We can plot observations and predicted values:

```
8 u=seq(min(x),max(x),length.out=1000)
9 plot(x,y,col="gray",main="n=200",xlab=NA,ylab=NA)
10 lines(u,predict(fit.tr,data.frame(x=u)),col="blue")
11 lines(u,predict(fit.bo,data.frame(x=u),n.trees=nb), col="purple")
```

We run this code for $n = 200$ and $n = 10000$

Boosting: a simple regression example



→ Boosting provides nice improvement over single regression tree

Boosting: Exercise

Consider the previous simple regression example:

- Re-run the code with $D = 4$, $B = 1000$ and $\lambda = 1$. Do you observe overfitting?
- Perform boosting with different values of D , B and λ and look how sensitive the results are to these choices

Consider the random forest exercise, on baseball player's salary:

- Perform boosting on the training set for a range of values of the shrinkage parameter λ , with $B = 1000$ trees and $D = 1$.
- Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.
- Compare the test MSE of boosting to the test MSE that results from bagging, random forest and CART model

Boosting for Classification

$y \in \{-1, 1\}$ is a qualitative variable

AdaBoost algorithm

Algorithm 4: AdaBoost

Select the number of trees B , and the tree depth D ;

Set initial weights, $w_i = 1/n$;

for $b = 1$ to B **do**

Fit a classification tree $\hat{m}_b(x)$ to the data using weights w_i ;

Update the weights: $\nearrow w_i$ if misclassified, $\searrow w_i$
otherwise[†];

end

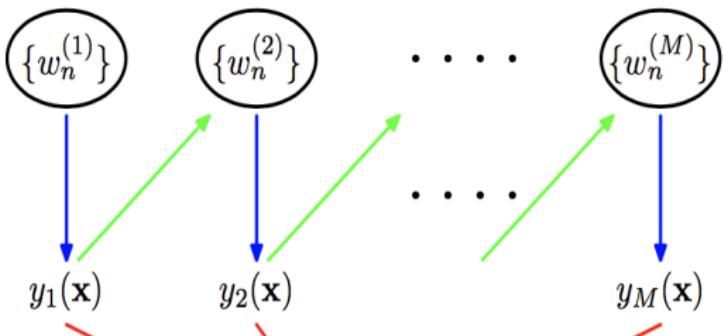
Output: $\hat{y}_i = \text{sign}(\sum_{b=1}^B \alpha_b \hat{m}_b(x))$

[†] If i is misclassified: $w_i \leftarrow w_i e^{\alpha_b}$, where $\alpha_b = \log(\frac{1 - err_b}{err_b})$ and err_b is the model's misclassification error, $err_b = \frac{\sum_{i=1}^n w_i I(y_i \neq \hat{m}_b(x_i))}{\sum_{i=1}^n w_i}$. If i is correctly classified $w_i \leftarrow w_i$.

- Observ. misclassified have more influence in the next classifier
- In the output, the contributions from classifiers that fit the data better are given more weight (a larger α_b means a better fit)

Schematic illustration of the boosting framework

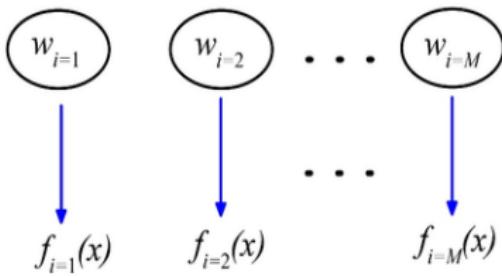
Schematic illustration of the boosting framework. Each base classifier $y_m(\mathbf{x})$ is trained on a weighted form of the training set (blue arrows) in which the weights $w_n^{(m)}$ depend on the performance of the previous base classifier $y_{m-1}(\mathbf{x})$ (green arrows). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(\mathbf{x})$ (red arrows).



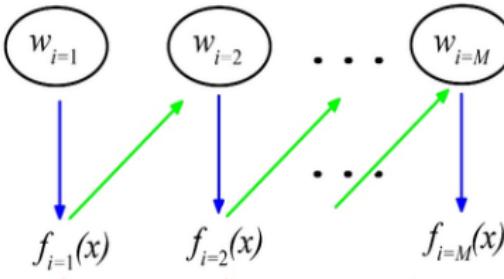
$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

(Source: Bishop 2006, Pattern recognition and machine Learning, Figure 14.1)

Boosting vs. bagging



b. Bagging



a. Boosting

(Source: Internet, @@)

→ Bootstrap samples \equiv Original sample reweighted independently

Illustration of boosting for classification tree

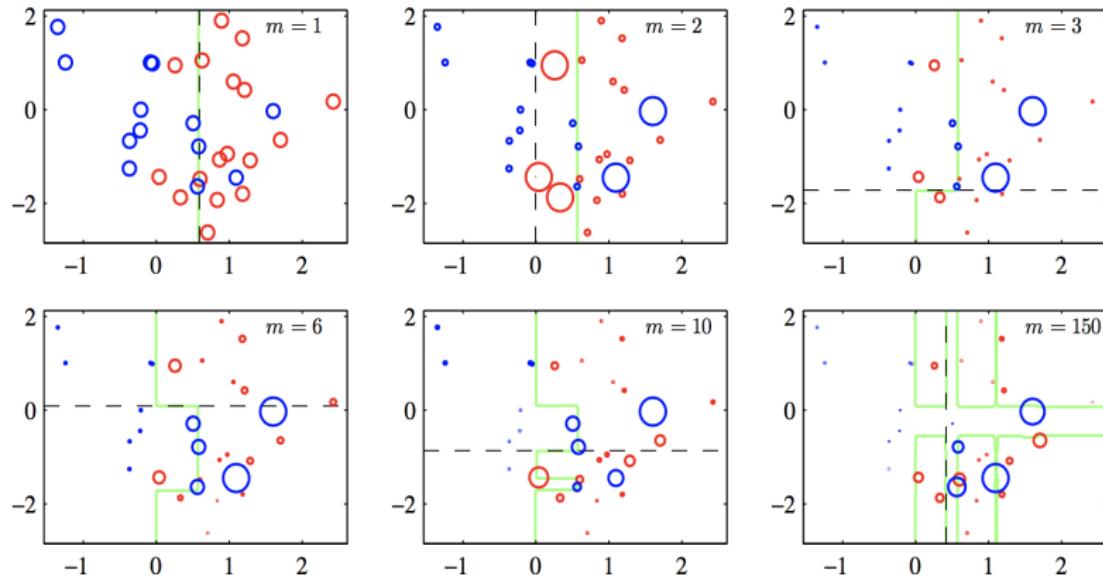


Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

(Source: Bishop (2006), Pattern recognition and machine Learning, Figure 14.2)

Generalizations into a unifying framework

- Breiman referred to AdaBoost with trees as the "best off-the-shelf classifier in the world" (NIPS Workshop, 1996)
- Friedman et al. (2000) show that Adaboost fits an additive model in a base learner, optimizing a novel exponential loss function, which is very similar to the binomial log-likelihood
- They proposed generalizations into a unifying framework, which includes several loss functions that can be used
- They describe loss functions for regression and classification that are more robust than squared error or exponential loss

→ Gradient boosting

Stochastic gradient boosting

Algorithm 5: Stochastic gradient boosting

Select number of trees B , tree depth D , shrinkage parameter λ ;

for $b = 1$ to B **do**

 Compute the gradient vector, $r_i = -\partial L(y_i, m(x_i))/\partial m(x_i)$;

 Draw a subset of the original sample (r^*, x^*) ;

 Fit a regression tree $m^b(x)$ of depth D to the data (r^*, x^*) ;

 Update the predicted values: $m(x) \leftarrow m(x) + \lambda m^b(x)$;

end

→ **Gradient boosting**: Depending the choice of the loss function, we consider a specific regression or classification model

→ **Stochastic**:

- Shrinkage: slow down the boosting process to avoid overfitting
- Subsampling: it reduces the computing time and, in many cases, it produces a more accurate model (see random forest)

Loss functions in regression, $y \in \mathbb{R}$

- **Squared error** loss function:

$$L = \frac{1}{2} (y_i - m(x_i))^2$$

for which the gradient vector is the residuals $r_i = y_i - m(x_i)$

- **Absolute error** loss function, or Laplacian:³⁵

$$L = |y_i - m(x_i)|$$

→ median of the conditional distribution . . . robust regression

- **Huber** loss function: a robust alternative to absolute error loss,

$$L = \begin{cases} \frac{1}{2}(y_i - m(x_i))^2 & |y_i - m(x_i)| \leq \delta \\ \delta(|y_i - m(x_i)| - \delta/2) & |y_i - m(x_i)| > \delta \end{cases}$$

³⁵We can also derive a **quantile** loss function: $L = (1 - \alpha)|y_i - m(x_i)|$ if $|y_i - m(x_i)| \leq 0$, and $L = (1 - \alpha)|y_i - m(x_i)|$ otherwise (α : desired quantile) 

Loss functions in regression: A comparison

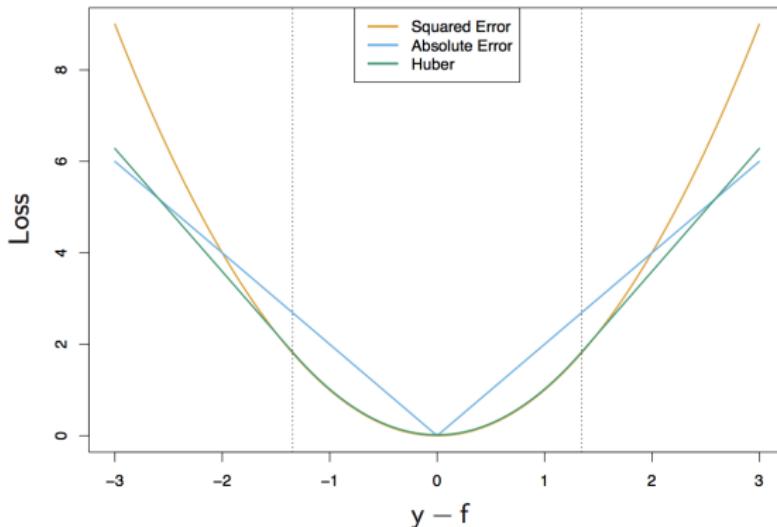


FIGURE 10.5. A comparison of three loss functions for regression, plotted as a function of the margin $y - f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large.

(Source: Hastie et al., 2009)

- When robustness is a concern, squared error is not the best criteria

Loss functions in classification, $y \in \{-1, 1\}$

- Misclassification loss function:³⁶

$$L = \mathbb{1}(\text{sign}[m(x)] \neq y)$$

- Adaboost loss function:

$$L = e^{-ym(x)}$$

- Bernouilli loss function, or Binomial deviance:

$$L = \log(1 + e^{-2ym(x)})$$

- Minimizing Adaboost or Bernouilli loss functions leads to the same solution at the population level . . . not in finite sample
- Bernouilli loss function is more robust to outliers in finite sample

³⁶The sign of $m(x_i)$ implies that observations with $y_i m(x_i) > 0 (< 0)$ are classified correctly (misclassified)

Loss functions in classification: A comparison

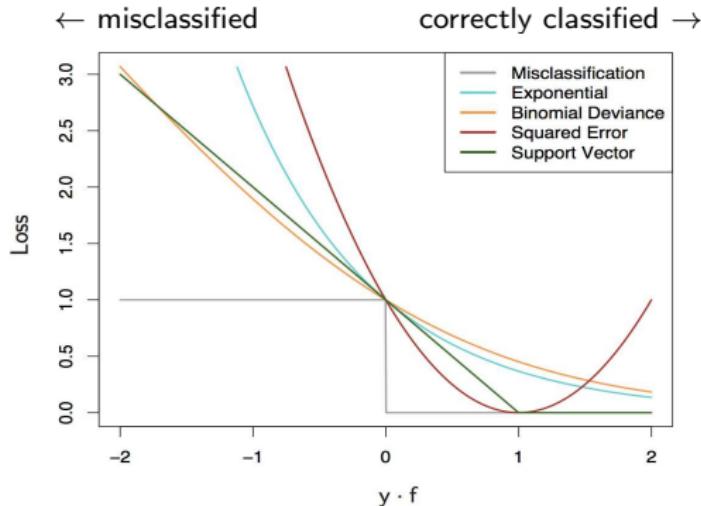


FIGURE 10.4. Loss functions for two-class classification. The response is $y = \pm 1$; the prediction is f , with class prediction $\text{sign}(f)$. The losses are misclassification: $I(\text{sign}(f) \neq y)$; exponential: $\exp(-yf)$; binomial deviance: $\log(1 + \exp(-2yf))$; squared error: $(y - f)^2$; and support vector: $(1 - yf)_+$ (see Section 12.3). Each function has been scaled so that it passes through the point $(0, 1)$. (Source: Hastie et al., 2009)

- More weight for obs. more clearly misclassified (large negative $y \cdot m(x)$)
- When robustness is a concern, exponential loss is not the best criteria

Tuning parameters

- The **number of trees** B . Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all.
- The **number of splits** D in each tree, which controls the complexity of the boosted ensemble. Often $D = 1$ works well, in which case each tree is a stump, consisting of a single split.
- The **shrinkage parameter** λ . This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem.³⁷

→ We use **cross-validation** to select B , D and λ

³⁷Very small λ can require very large B in order to achieve good performance

Boosting: Interpretation

- Single tree are highly interpretable. Linear combinations of trees must therefore be interpreted in a different way.
- **Variable importance:** using the relative importance of a variable for a single tree,³⁸ we then average over the trees³⁹
- After the most relevant variables have been identified, the next step is to attempt to understand the nature of the dependence of the approximation $m(X)$ on their joint values
- **Partial dependence plot** illustrate the marginal effect of the selected variables on the response after integrating out the other variables.

³⁸The squared relative importance of X_i is the sum of squared improvements over all internal nodes for which it was chosen as the splitting variable

³⁹Due to the stabilizing effect of averaging, this measure turns out to be more reliable than its counterpart (10.42) for a single tree

Application: spam email

The data for this example consists of information from 4601 email messages, in a study to try to predict whether the email was spam.

The response variable is binary, with values **email** or **spam**, and there are 57 predictors as described below:

- 48 quantitative predictors: the percentage of words in the email that match a given word⁴⁰
- 6 quantitative predictors: the percentage of characters in the email that match a given character (; ! # ([\$)
- Uninterrupted sequences of capital letters: average length (**CAPAVE**), length of the longest (**CAPMAX**), sum of the length (**CAPTOT**)

→ use gradient boosting to design an automatic spam detector that could filter out spam before clogging the users' mailboxes

⁴⁰Examples include **business**, **address**, **internet**, **free**, and **george**. The idea was that these could be customized for individual users (Hastie et al., 2009)

Application: spam email

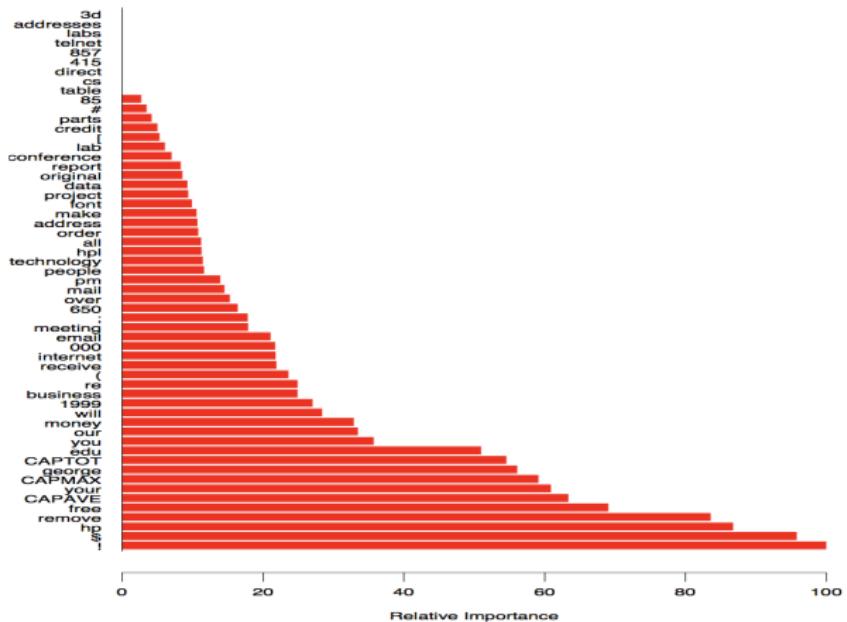


FIGURE 10.6. Predictor variable importance spectrum for the spam data. The variable names are written on the vertical axis.

(Source: Hastie et al., 2009)

Application: partial dependence

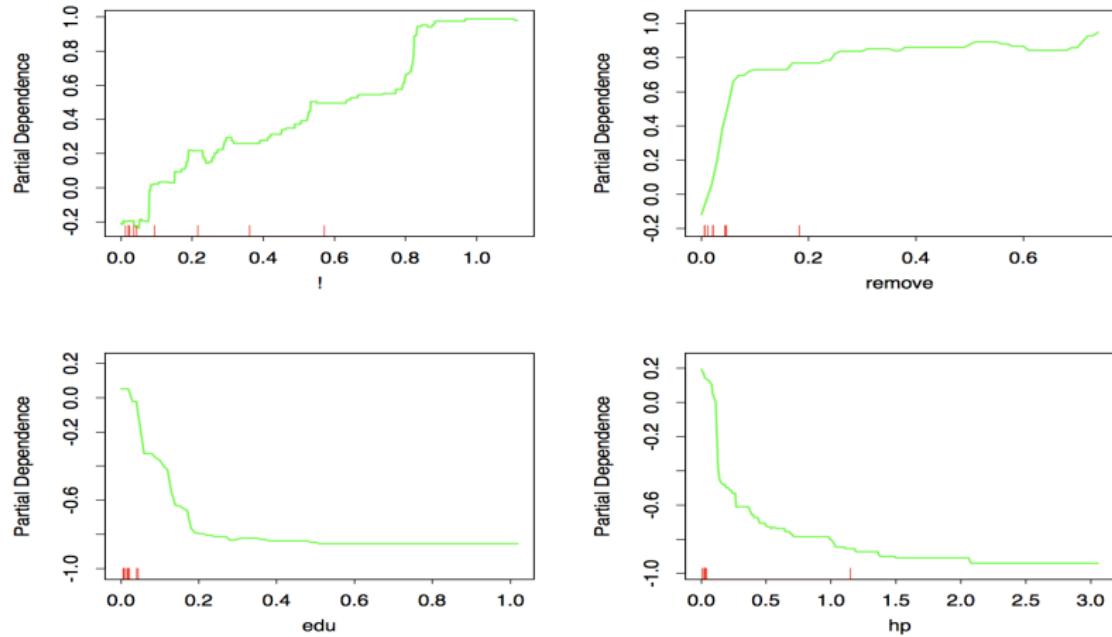


FIGURE 10.7. Partial dependence of log-odds of spam on four important predictors. The red ticks at the base of the plots are deciles of the input variable.
(Source: Hastie et al., 2009)

→ effect of X_j on $m(X)$ after accounting for the average effects of the other variables

Application: joint frequencies

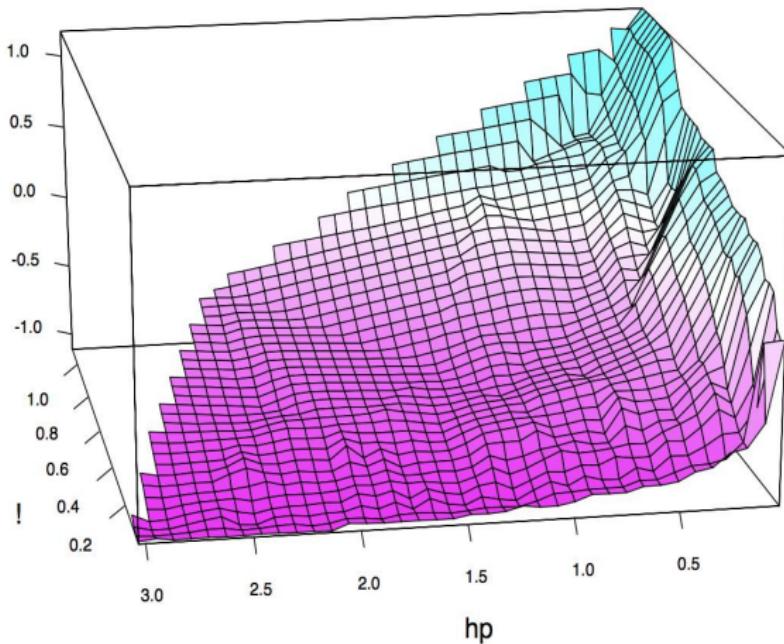


FIGURE 10.8. Partial dependence of the log-odds of spam vs. email as a function of joint frequencies of hp and the character !.

→ This plot displays strong interaction effects

2. Methods and Algorithms

- Ridge and Lasso Regression
- Classification and Regression Tree
- Bagging and Random Forests
- Boosting
- **Support Vector Machine**
- Neural Networks and Deep Learning

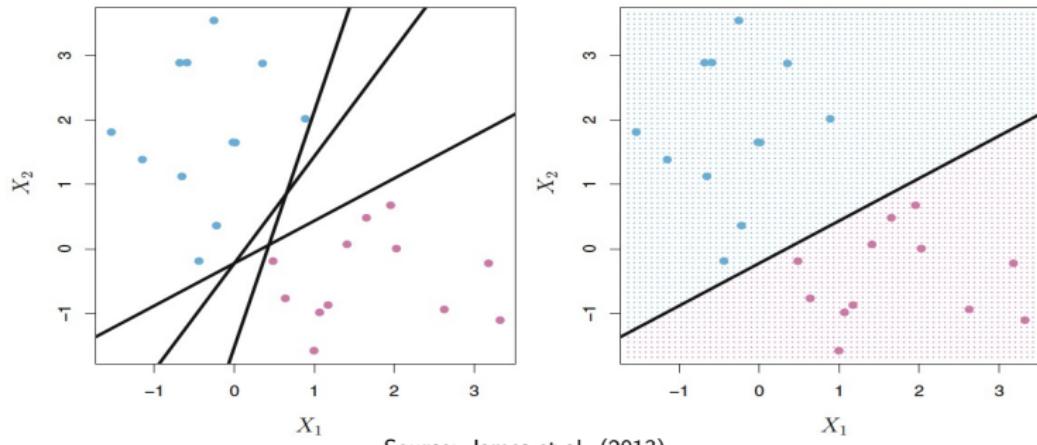
Introduction

- A method developed in the computer science community in the 1990s
- It uses a basis expansion to capture non-linear class boundaries
- Well suited for classification of complex but small- or medium-sized datasets
- Often considered one of the best "out of the box" classifiers

Support Vector Classifier

The separable case

Classification and hyperplane



Source: James et al. (2013)

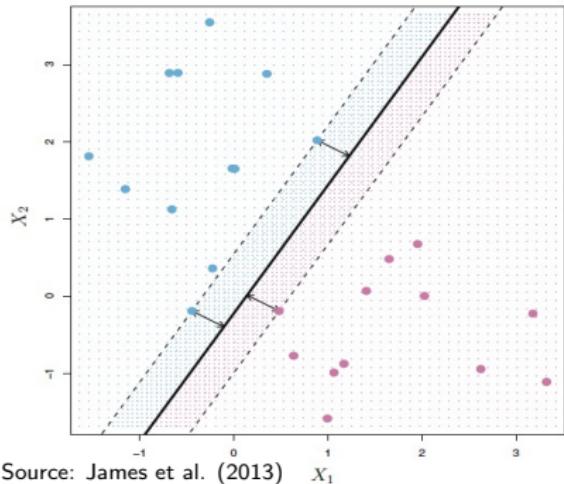
- A **hyperplane** separates the space in two halves:

$$\beta_0 + X_1\beta_1 + X_2\beta_2 > 0 \text{ (blue)} \quad \text{or} < 0 \text{ (red)}$$

- An ∞ number of hyperplanes, with same classification score
- What would make a difference is their **capacity to generalize**

The Maximal Margin Classifier

Margins = two parallel separating hyperplanes, located at the smallest distance from the observations of each classes



- *Margins*: the dashed lines
- *Support vectors*: the two blue points and the purple point that lie on the margins
- *Optimal hyperplane*: solid line

Principle: Maximize the distance between the two margins

The maximal margin (or optimal) hyperplane is the separating hyperplane that is **farthest from the training observations**

How to find the maximal margin?

It is an optimization problem:

$$\underset{\beta, \|\beta\|=1}{\text{maximize}} M \quad \text{subject to} \quad y_i(X_i\beta) \geq M, \forall i = 1, \dots, n$$

Here, $\|\beta\| = 1$ ensures that the perpendicular distance from the i^{th} observation to the hyperplane is given by $y_i(X_i\beta)$. Thus, the restriction ensures that each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane.

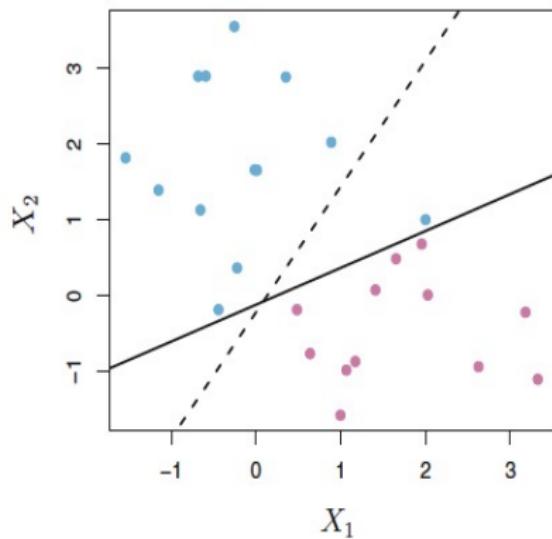
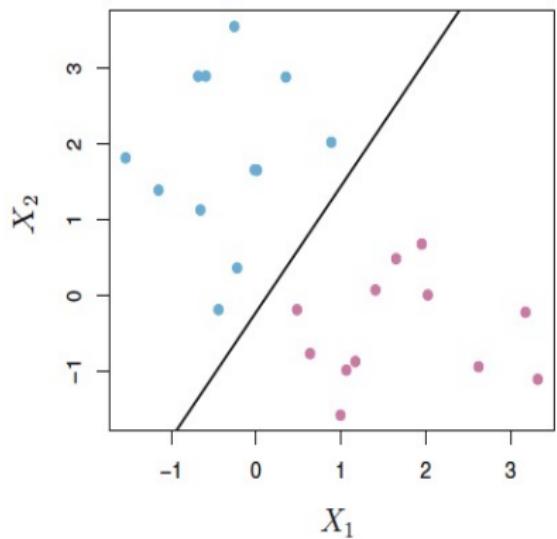
It is equivalent to:⁴¹

$$\underset{\beta}{\text{minimize}} \|\beta\|^2 \quad \text{subject to} \quad y_i(X_i\beta) \geq 1, \forall i = 1, \dots, n$$

⁴¹We get rid of $\|\beta\| = 1$ by replacing the restriction with $y_i(X_i\beta) \geq M\|\beta\|$ and, by setting $\|\beta\| = 1/M$, see Hastie et al (2009, section 4.5.2)

Sensitivity to individual observations

Adding one blue observation leads to a quite different hyperplane, with a significant decrease of the distance between the two margins



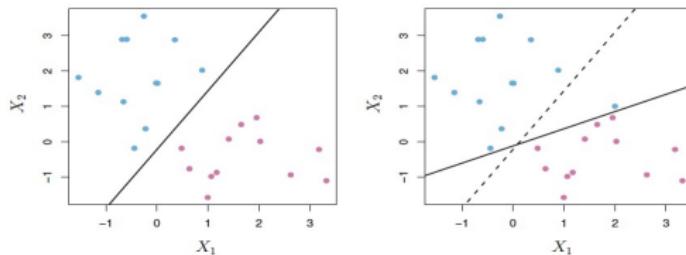
→ It could be worthwhile to **misclassify** a few training observations in order to obtain a better **generalization** (out-sample classification)

Support Vector Classifier (SVC)

Why should we consider a classifier that is not a perfect separator?

In the interest of:

- greater robustness to individual observations
- better classification of the out-sample observations



Underlying principles:⁴²

- SVC: maximal margin classifier, tolerating margin violations
- Logit: minimize misclassification error

⁴²Figures: logit \approx SVC (left), logit=solid line & SVC=dashed line (right)

How to tolerate margin violations?

It is a slightly modified [optimization problem](#):

$$\begin{array}{ll} \text{maximize}_{M} & M \\ \text{subject to} & y_i(X_i\beta) \geq M(1 - \epsilon_i), \\ \beta, \|\beta\|=1 & \\ & \text{and} \quad \epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C \end{array}$$

$\forall i = 1, \dots, n$, where C is a nonnegative tuning parameter.

- $\epsilon_1, \dots, \epsilon_n$ are *slack variables* that allow observations to be on the wrong side of the margin ($\epsilon_i > 0$) or hyperplane ($\epsilon_i > 1$)
- C is a *budget* for the amount that the margins can be violated
 - $C = 0$: no margin violation is tolerated
 - as C increases, we become more tolerant of margin violations
- C is the maximal number of observations allowed to be on the wrong side of the hyperplane
- In practice, C is a tuning parameter chosen by cross-validation

Support Vector Classifier vs. Logit model

- **SVC**: the previous optimization problem can be rewritten as:⁴³

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \underbrace{\max [0, 1 - y_i(X_i \beta)]}_{\text{hinge loss function}} + \lambda \sum_{k=1}^K \beta_k^2$$

It's a minimization of the hinge loss function with penalization

- **Logit**: minimizing missclassification, we have:

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \underbrace{\log \left(1 + e^{-y_i(X_i \beta)} \right)}_{\text{softmax function}},$$

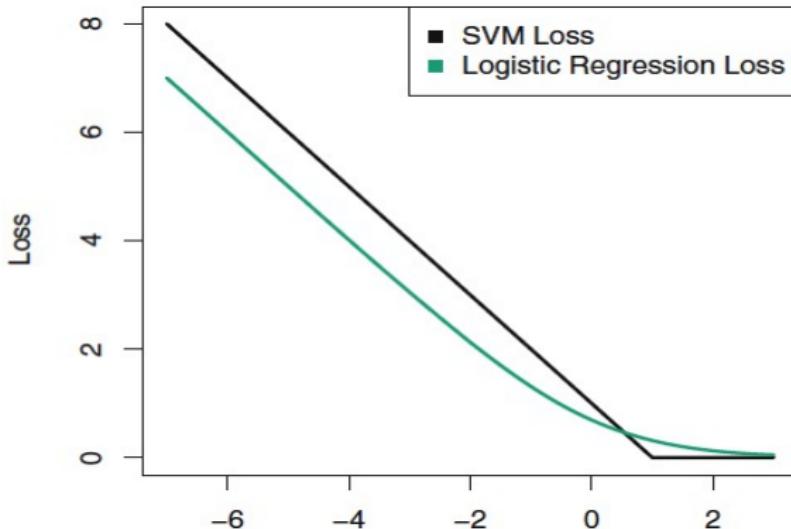
It's a minimization of the softmax function, no penalization

→ SVC ≈ **penalized** Logit model, using a **hinge loss function**

→ Role of penalization = tradeoff min missclassif. & max margin

⁴³With $\lambda = 1/(2C)$, see Hastie et al. (2009)

SVC and Logit: loss function



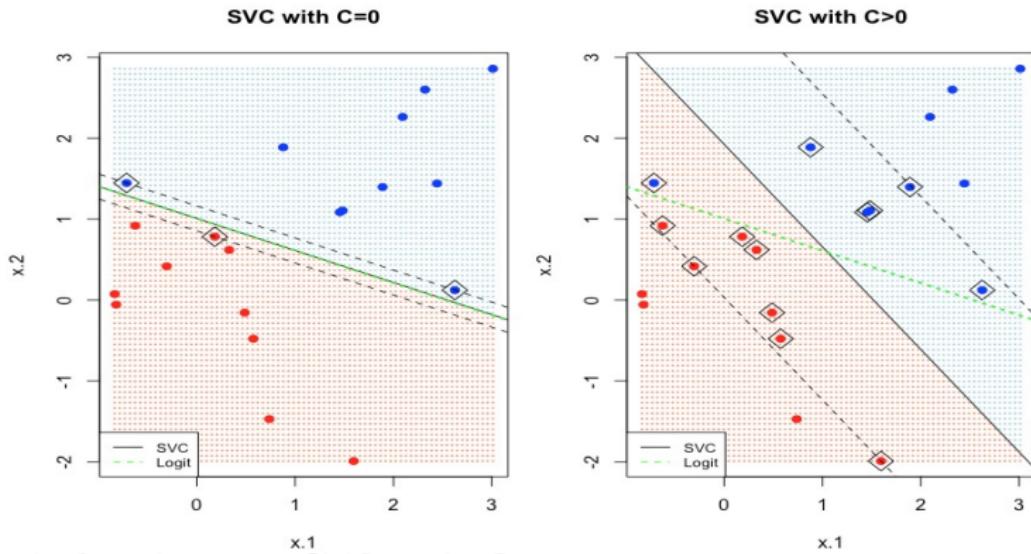
Source: James et al. (2013)

$$y_i(X_i\beta)$$

Overall, the two loss functions have quite similar behavior

Hinge loss = 0 for obs on the correct side of the margin: $y_i(X_i\beta) > 1$

SVC and Logit: The separable case



- Left: $\text{Logit} \approx \text{SVC with } C = 0$
 - Right: $\text{Logit} \not\approx \text{SVC with } C > 0$ (chosen by cross-validation)
- SVC = tradeoff between min missclassification & max margin

44

⁴⁴ max margin: pushing away the obs. as far as possible from the hyperplane
min missclassif: smallest aggregated distance from the hyperplane of wrong obs

Support Vector Classifier

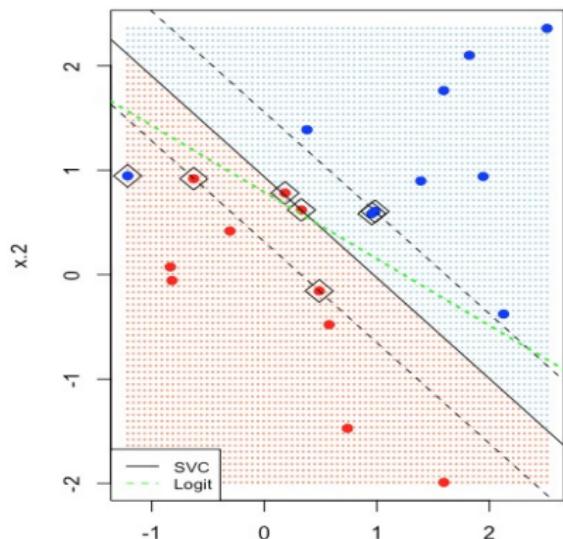
The non-separable case

The non-separable case

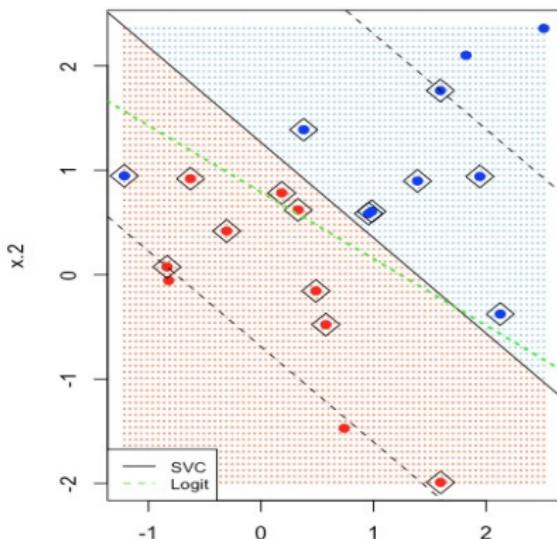
- In the non-separable case, some observations are on the wrong side of the hyperplane
- The Maximal Margin Classifier has no solution
- Logit minimizes the **aggregated distance** from the hyperplane of the missclassified observations, **not** the **number** of missclass.
- SVC is a **tradeoff** between:
 - **minimizing** the aggregated distance from the hyperplane of the missclassified observations
 - **pushing away** as far as possible from the hyperplane the correctly classified observations

SVC and Logit: The non-separable case

SVC with $C=0$



SVC with $C>0$



- Left: Logit $\not\approx$ SVC with C small
- Right: Logit $\not\approx$ SVC with C chosen by cross-validation
- SVC: 1 mistake - Logit: 3 mistakes

Support Vector Machine

Nonlinear separability

Support Vector Machine (SVM)

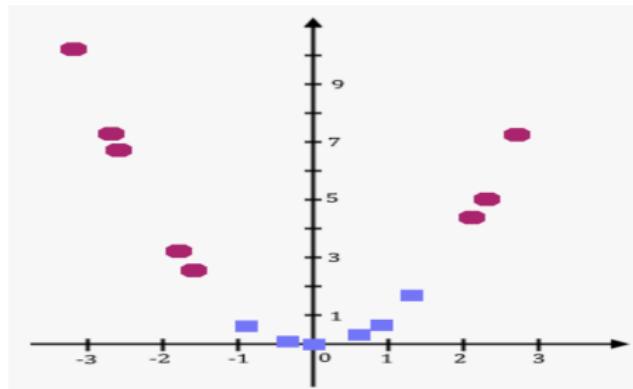
- Many datasets are not linearly separable
- Adding polynomial features and interactions can be used
- But a low polynomial degree cannot deal with very complex datasets
- The **support vector machine (SVM)** is an extension of the **support vector classifier** that results from enlarging the feature space in a specific way, using **kernels**.
- SVM works well for complex but small- or medium-sized datasets

Moving into higher dimension

Find a SVM classifier to identify teenagers from the height:⁴⁵



Using the projection $\varphi : x \mapsto \left(\frac{x-150}{10}, \left(\frac{x-150}{10} \right)^2 \right)$, we obtain:

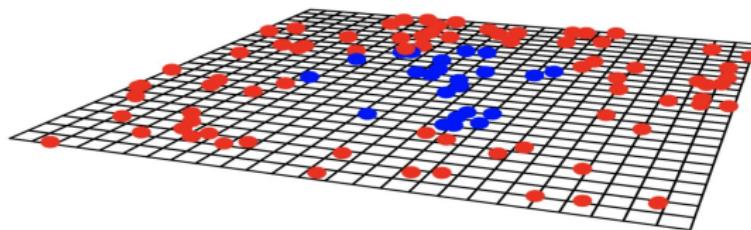


The data are linearly separable in the 2-dimensional space

⁴⁵Source: Internet

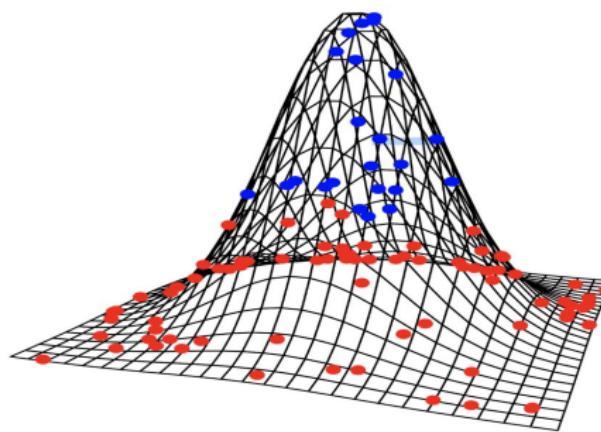
The kernel trick

The data are not linearly separable in the 2-dimensional space, S



The kernel trick:

Source: <https://freakonometrics.hypotheses.org/52775>



The data are linearly separable in the 3-dimensional space, S'

SVM: The optimization problem

It is the SVC optimization problem, with **transformed covariates**:

$$\underset{\beta, \|\beta\|=1}{\text{maximize}} M \quad \text{subject to} \quad y_i(\varphi(X_i)\beta) \geq M, \forall i$$

or

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \max [0, 1 - y_i(\varphi(X_i)\beta)] + \lambda \sum_{k=1}^K \beta_k^2$$

In the resolution, φ only appears in the form $\varphi(X_i)^\top \varphi(X_j)$. Thus,

- we don't need to express explicitly φ
- we don't need to express the higher dimension space S'

We use a **kernel function** defined as $K(x, x') = \varphi(x)^\top \varphi(x')$

Polynomial kernel

The kernel should be a symmetric positive (semi-) definite function.

The d th-degree polynomial kernel is: $K(x, x') = (1 + \langle x, x' \rangle)^d$

- 1st-degree polynomial kernel with two covariates X_1 and X_2 :⁴⁶

$$K(X, X') = (1 + \langle X, X' \rangle) = (1 + X_1 X'_1 + X_2 X'_2)$$

- With $\varphi(X) = \{1, X_1, X_2\}$, we have $K(X, X') = \varphi(X)^\top \varphi(X')$. It corresponds to the linear case, or SVC.

→ SVM with 1st-degree polynomial kernel is similar to SVC

⁴⁶With two n -vectors, the inner product is: $\langle x_1, x_2 \rangle = x_1^\top x_2 = \sum_{i=1}^n x_i y_i$

Polynomial kernel

The d th-degree polynomial kernel is: $K(x, x') = (1 + \langle x, x' \rangle)^d$

- 2nd-degree polynomial kernel with two covariates X_1 and X_2 :

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 = (1 + X_1 X'_1 + X_2 X'_2)^2 \\ &= 1 + 2X_1 X'_1 + 2X_2 X'_2 + (X_1 X'_1)^2 + (X_2 X'_2)^2 + 2X_1 X'_1 X_2 X'_2 \end{aligned}$$

- Here, $\varphi(X) = \{1, \sqrt{2}X_1, \sqrt{2}X_2, X_1^2, X_2^2, \sqrt{2}X_1 X_2\}$ defines a 6-dimensional space, with squared and interaction terms
 - We move from 3-dimensional space to 6-dimensional space
- SVM with d th-degree polynomial kernel ($d \geq 2$) is similar to SVC with additional powers and interaction terms of the covariates

Radial kernel

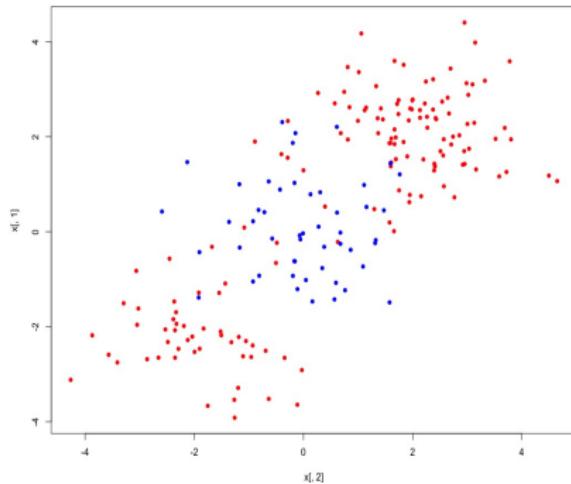
Radial basis function (RBF) kernel: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$

- $\gamma > 0$ accounts for the smoothness of the decision boundary⁴⁷
 - It returns values between 0 and 1:
 - It returns large value for x close to x'
 - It returns small value for x far from x'
 - It is a **similarity measure** between two covariates
- The radial kernel has a **local** behavior

⁴⁷ Bias-variance tradeoff: large value of γ leads to high variance (overfitting), small value leads to low variance and smoother boundaries (underfitting)

Illustration: Simulated data

```
1 set.seed(1)
2 x=matrix(rnorm(200*2), ncol=2)
3 x[1:100,]=x[1:100,]+2
4 x[101:150,]=x[101:150,]-2
5 y=c(rep(1,150), rep(2,50))
6 plot(x[,2], x[,1], pch=16, col=y*2)
```



Non-linear decision boundaries → SVC will perform poorly

Illustration: Fit SVM with polynomial and radial kernels

We can fit a SVM with 2nd-degree polynomial kernel and fixed cost of constraints violation:

```
7 library(e1071)
8 dat=data.frame(x=x,y=as.factor(y))
9 svmfit=svm(y~., data=dat, kernel="polynomial", cost=1, degree=2)
10 plot(svmfit,dat, grid=200)
```

Or select the cost parameter by 10-fold CV among several values:

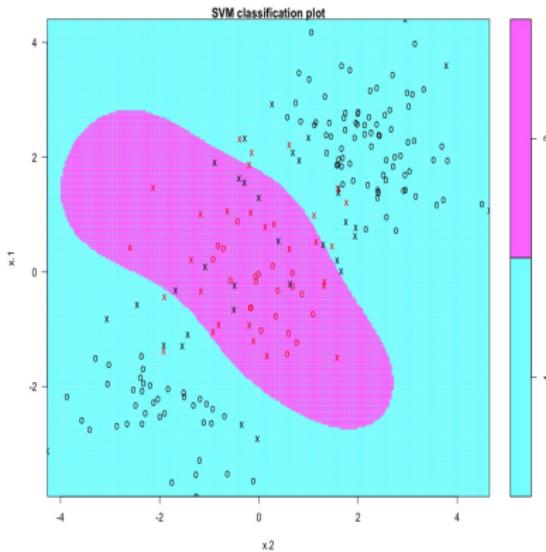
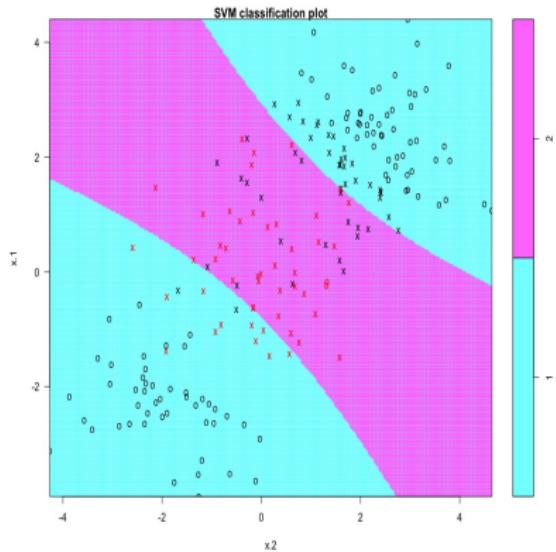
```
11 tune.out=tune(svm, y~., data=dat, kernel="polynomial", degree=2,
+                 ranges=list(cost=c(.1,1,10,100)))
12 plot(tune.out$best.model,dat, grid=200)
13 summary(tune.out)
```

Similarly, we can fit a SVM with radial kernel:⁴⁸

```
14 tune.out=tune(svm, y~., data=dat, kernel="radial", ranges=list(
+                 cost=c(.1,1,10,100), gamma=c(.5,1,2,3,4)))
15 plot(tune.out$best.model,dat, grid=200)
```

⁴⁸We then have 2 tuning parameters, the cost of constraints violation and

Illustration: Polynomial vs. radial kernels



- Either kernel is capable of capturing the decision boundary
- However, the results are different

ROC curve

- With more than 2 covariates, we can't plot decision boundary
- We can produce a ROC curve to analyze the results
- SVM doesn't give probabilities to belong to classes, as in logit
- We compute scores of the form $\hat{f}(X) = \varphi(X_i)\hat{\beta}$ for each obs.
Scores = predicted values.
- For any given **cutoff t** , we can classify observations into a category, depending on whether

$$\hat{f}(X) < t \quad \text{or} \quad \hat{f}(X) \geq t$$

- The **ROC curve** is obtained by computing the false positive and true positive rates for a **range of values of t**

Illustration: ROC curves

We write a short function to plot a ROC curve:

```
16 library(ROCR)
17 rocplot=function(pred,truth,...){
18   predob=prediction(pred,truth)
19   perf=performance(predob, "tpr", "fpr")
20   plot(perf,...) }
```

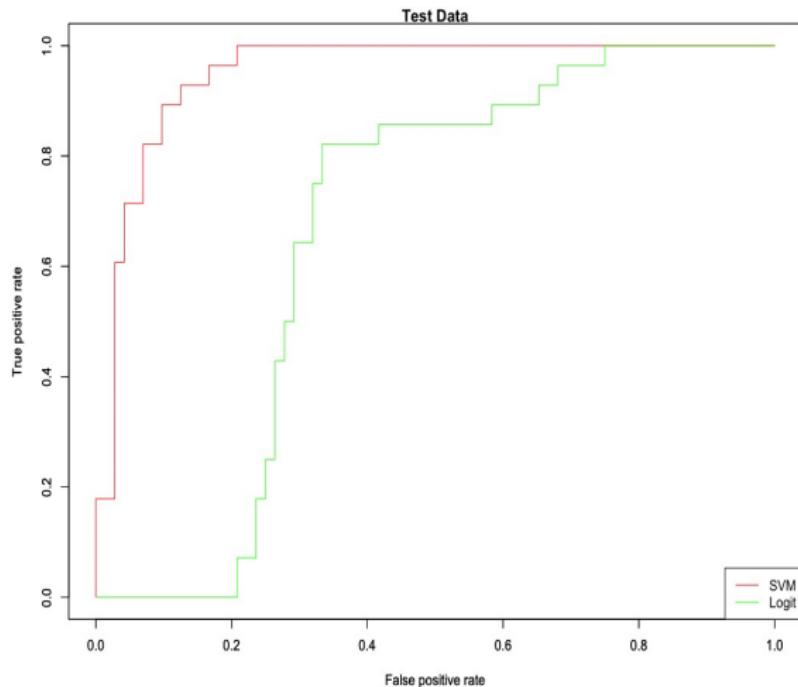
We can fit a SVM with radial kernel and plot a ROC curve:

```
21 set.seed(1)
22 train=sample(200,100)
23 train=sort(train,decreasing=TRUE) # to avoid reverse ROC
24 svmfit=svm(y~.,data=dat[train,],kernel="radial",cost=1,
25             gamma=0.5)
26 fit=attributes(predict(svmfit,dat[-train,],decision.values=
27                   TRUE))$decision.values
28 rocplot(fit,dat[-train,"y"], main="Test Data", col="red")
```

We can also fit a Logit model and plot a ROC curve:

```
27 lgt=glm(y ~ .,data=dat[train,],family=binomial(link='logit'))
28 fit=predict(lgt,dat[-train,],type="response")
29 par(new=TRUE)
30 rocplot(fit,dat[-train,"y"], col="green")
```

Illustration: ROC curves



As expected in this example, SVM outperforms Logit model

2. Methods and Algorithms

- Ridge and Lasso Regression
- Classification and Regression Tree
- Bagging and Random Forests
- Boosting
- Support Vector Machine
- Neural Networks and Deep Learning

Neural networks with one covariate

Looking for a more flexible model ...

A linear model maybe quite restrictive:

$$y \approx \alpha + \beta x$$

We can obtain a more flexible model by adding:

- successive powers *polynomial regression*⁴⁹

$$y \approx \alpha + \sum_{m=1}^M \beta_m x^m$$

- nonlinear functions of linear combinations .. *neural networks*⁵⁰

$$y \approx \alpha + \sum_{m=1}^M \beta_m f(\alpha_m + \delta_m x)$$

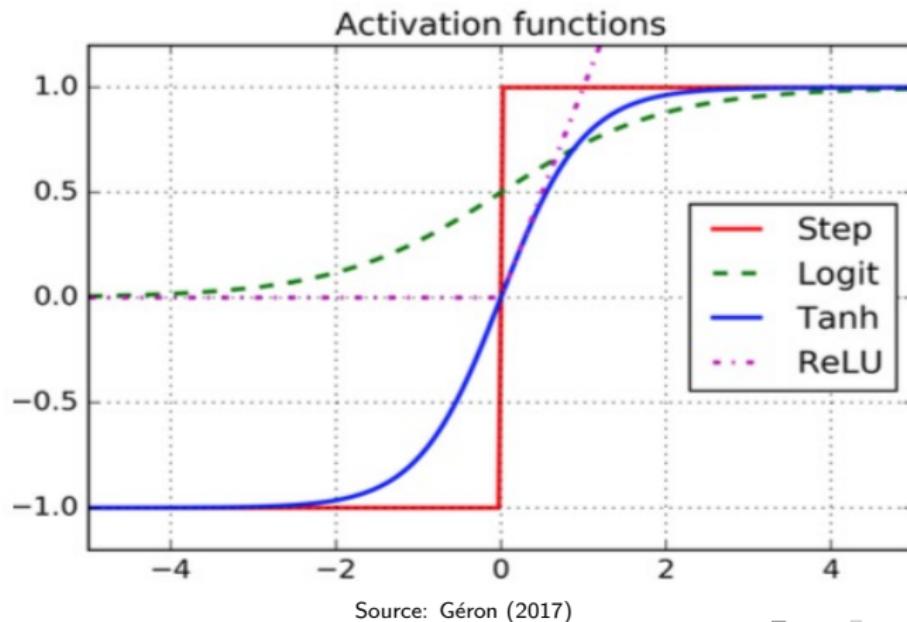
where f is an *activation* function – a fixed **nonlinear** function

⁴⁹ $y \approx \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots$

⁵⁰ $y \approx \alpha + \beta_1 f(\alpha_1 + \delta_1 x) + \beta_2 f(\alpha_2 + \delta_2 x) + \beta_3 f(\alpha_3 + \delta_3 x) + \dots$

Common examples of activation functions

- The **logistic** (or sigmoid) function: $f(x) = \frac{1}{1+e^{-x}}$
- The **hyperbolic tangent** function: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- The **Rectified Linear Unit (ReLU)**: $f(x) = \max(0, x) = (x)_+$



Neural network vs. polynomial: A simple example

Let us consider a realistic (simulated) sample

```
1 set.seed(1) ; n=200
2 x=sort(runif(n))
3 y=sin(12*(x+.2))/(x+.2) + rnorm(n)/2
4 df=data.frame(y,x)
```

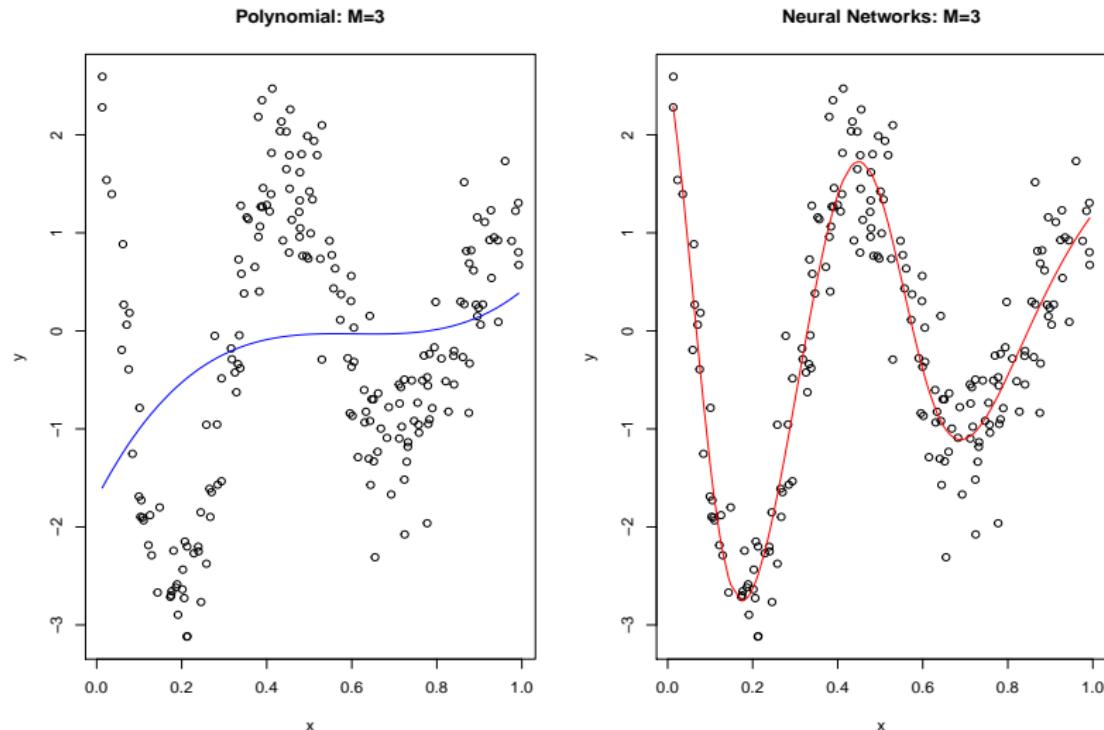
We can fit a polynomial regression with $M = 3$:

```
5 ols=lm(y~x+I(x^2)+I(x^3))
6 plot(x,y, main=" Polynomial: M=3")
7 lines(x, predict(ols), col="blue")
```

We can fit a neural network model with $M = 3$:

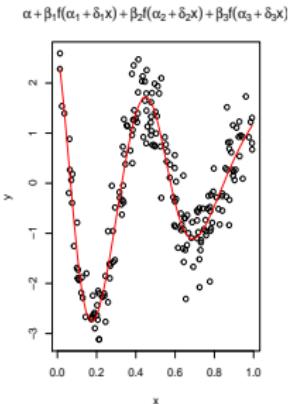
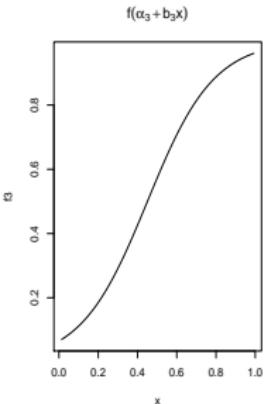
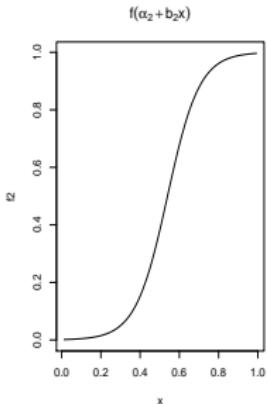
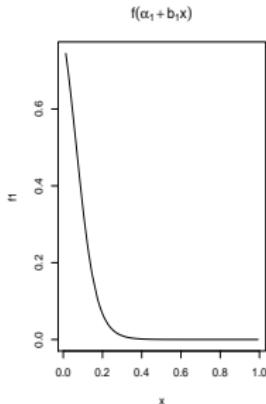
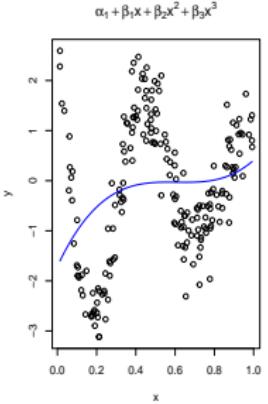
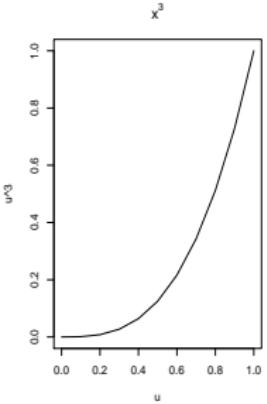
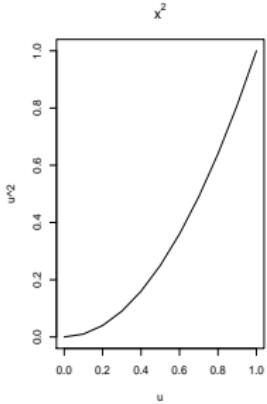
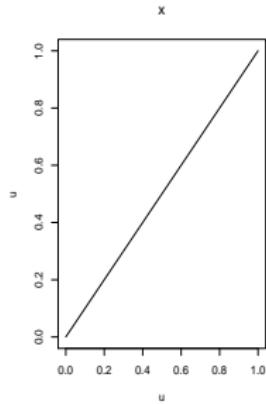
```
8 library(neuralnet)
9 nn=neuralnet(y~x, data=df, hidden=3, threshold=.05)
10 yfit=compute(nn, data.frame(x))$net.result
11 plot(x,y, main=" Neural Networks: M=3")
12 lines(x,yfit, col="red")
```

Neural network vs. polynomial: A simple example



→ Neural networks can capture nonlinearity

A weighted sum of fixed/adjustable components



Fixed vs. adjustable components

Why neural networks perform better than polynomial regression in the previous example?

- Polynomial regression is based on **fixed** components, or bases:⁵¹

$$x, x^2, x^3, \dots, x^M$$

- Neural network is based on **adjustable** components, or bases:⁵²

$$f(\alpha_1 + \delta_1 x), f(\alpha_2 + \delta_2 x), \dots, f(\alpha_M + \delta_M x)$$

- Adjustable components have tunable internal parameters
- They can express several shapes, not just one (fixed) shape
- Each component is more flexible than a fixed component

→ Adjustable components enable to capture complex models with fewer components (smaller M)

⁵¹ $y \approx \alpha + \sum_{m=1}^M \beta_m x^m$

⁵² $y \approx \alpha + \sum_{m=1}^M \beta_m f(\alpha_m + \delta_m x)$

Neural networks with several covariates

Neural network with several covariates

With a set of covariates $X = (1, x_1, x_2, \dots, x_k)$, we have

$$y \approx \alpha + \sum_{m=1}^M \beta_m f(\alpha_m + X\delta_m)$$

- The nonlinearity of the activation function f is essential, otherwise it is a simple linear model in X
- Combining several nonlinear functions f is essential to capture interaction effects, $M > 1$, otherwise it is just a logit model⁵³

By adding nonlinear functions of linear combinations of X , we obtain a more flexible model, which is able to capture nonlinearity and interaction effects

⁵³With $M = 1$ and the logistic activation function, it is a logit model

Interaction effects

Adding two nonlinear functions can generate an [interaction effect](#):

$$y \approx \alpha + \sum_{m=1}^2 \beta_m f(\alpha_m + x_1 \delta_m + x_2 \gamma_m)$$

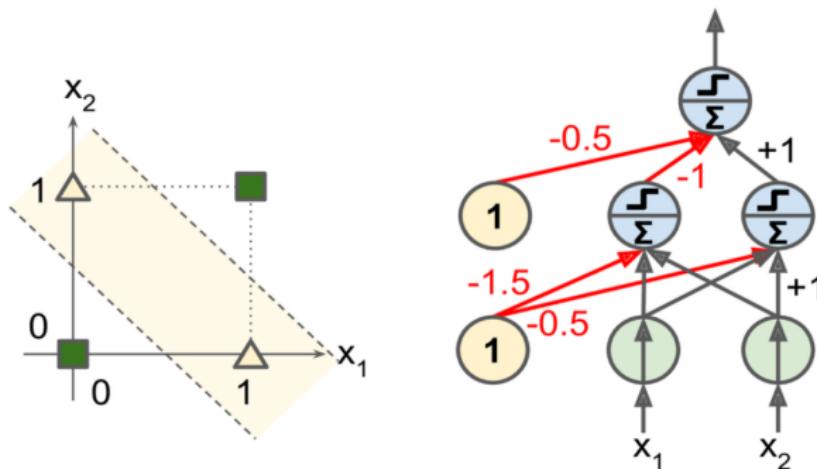
Let us consider $\alpha = \alpha_1 = \alpha_2 = 0$, $\beta_1 = -\beta_2 = \frac{1}{4}$, $\alpha_1 = \alpha_2 = 0$, $\delta_1 = \delta_2 = \gamma_1 = -\gamma_2 = 1$ and $f(z) = z^2$, we have:

$$\begin{aligned} y &\approx 0 + \frac{1}{4}(0 + x_1 + x_2)^2 - \frac{1}{4}(0 + x_1 - x_2)^2 \\ &\approx \frac{1}{4}[(x_1 + x_2)^2 - (x_1 - x_2)^2] \\ &\approx x_1 x_2 \end{aligned}$$

So the sum of two nonlinear transformations of linear functions can give us an interaction! Here, we would always get a 2nd-degree polynomial in X . Other activations do not have such a limitation.

XOR: Exclusive or (true if its arguments differ)

Diagram of $y \approx \alpha + \sum_{m=1}^2 \beta_m f(\alpha_m + x_1 \delta_m + x_2 \gamma_m)$



Source: Géron (2017, p.260)

With the step activation function (=1 if positive, 0 otherwise)

$$y \approx -0.5 - \mathbb{I}(x_1 + x_2 > 1.5) + \mathbb{I}(x_1 + x_2 > 0.5)$$

With (0,0) or (1,1) we have -0.5, with (1,0) or (0,1) we have +0.5

Neural network with a single hidden layer

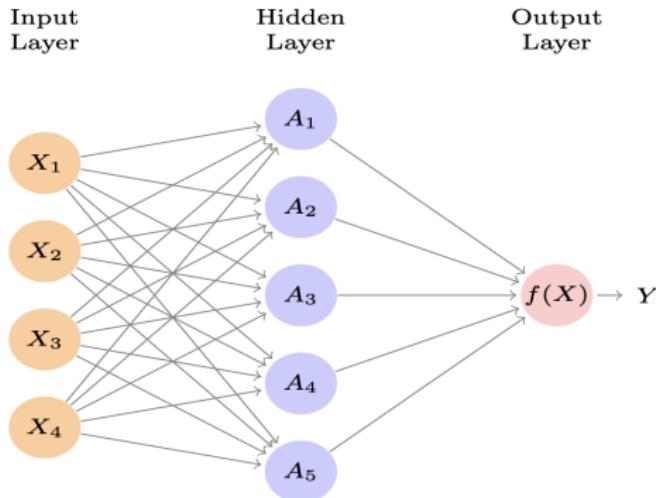


FIGURE 10.1. Neural network with a single hidden layer. The hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs X_1, X_2, \dots, X_p . Hence these A_k are not directly observed. The functions $h_k(\cdot)$ are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_k as inputs, resulting in a function $f(X)$.

Source: James et al. (2021)

Diagram of $y \approx \alpha + \sum_{m=1}^M \beta_m f(\alpha_m + X\delta_m)$ with $M = 5$ neurons

Ridge regularization & standardization

- NN tends to overfit due to large number of coefficients
- A solution is to regularize similar to ridge regression:⁵⁴

Minimize the SSR subject to $\sum_{j=1}^p \theta_j^2 \leq c$

- The results are **sensitive to the scale** of the covariates
- It is best to **standardize** covariates before using Neural Networks, so that they are all on the same scale:

$$\frac{x - \bar{x}}{\sqrt{\text{Var}(x)}}$$

⁵⁴ θ is the set of coefficients α, β, δ

Backpropagation algorithm

In 1986, Rumelhart et al. found a way to train neural networks, with the **backpropagation** algorithm.⁵⁵ Today, we would call it a **Gradient Descent** using reverse-mode autodiff.

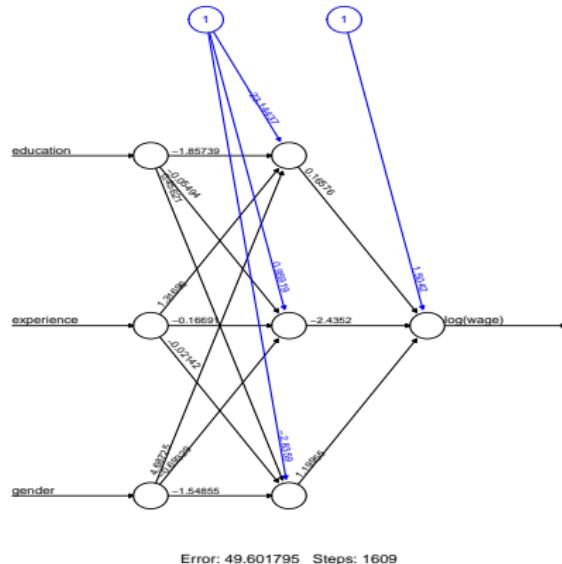
For each training instance:

- ① the algorithm first makes a prediction (forward path)
- ② measures the error
- ③ goes through each layer in reverse to measure the error contribution from each connection (reverse pass)
- ④ slightly tweaks the connection weights to reduce the error (Gradient Descent step)

⁵⁵Rumelhart et al.: Learning Internal Representations by Error Propagation

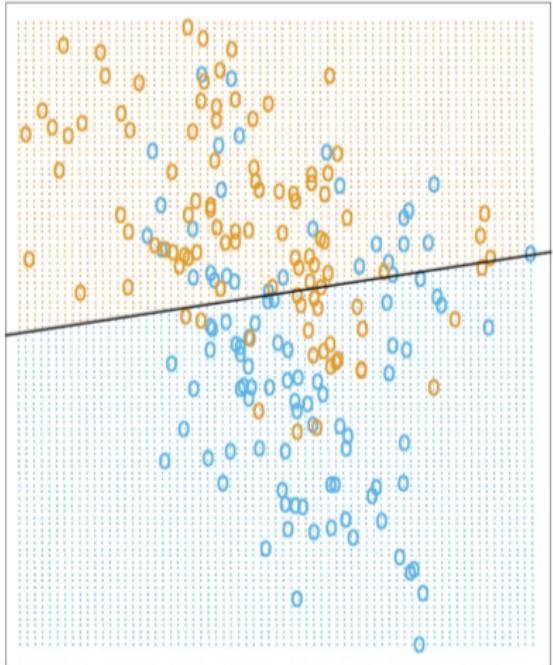
Application 1: Mincer equation

```
1 library(AER) ; data("CPS1985")
2 CPS1985$gender=as.numeric(CPS1985$gender)
3 library(neuralnet)
4 nn=neuralnet(log(wage) ~ education+experience+gender , data=
  CPS1985 , hidden=3, threshold=.05)
5 plot(nn)
```

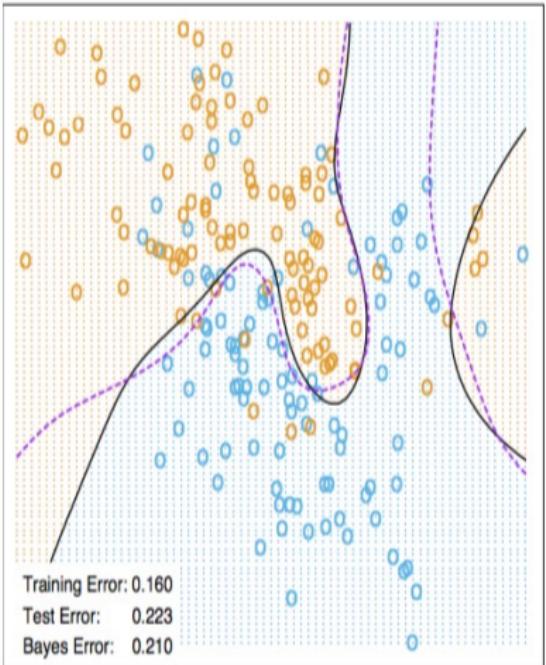


Application in classification

Logit model



Neural network with 10 units



Source: Hastie, Tibshirani and Friedman (2009), based on simulated data

In classification, the softmax function is applied to the outputs

Multilayer neural networks

Multilayer neural networks

Even greater flexibility is achieved via **composition** of activation functions:

$$y \approx \alpha + \sum_{m=1}^M \beta_m f\left(\underbrace{\alpha_m^{(1)} + \sum_{p=1}^P f\left(\alpha_p^{(2)} + X \delta_p^{(2)} \right) \delta_m^{(1)}}_{\text{it replaces } X} \right)$$

- The composition of activation functions puts one additional hidden layer between inputs and outputs → multi-layers NN
- A NN with three hidden layers can be obtained by simply repeating the procedure used to create the two layer basis.

Multilayer neural networks: when a NN has 2 or more hidden layers

Multilayer neural networks

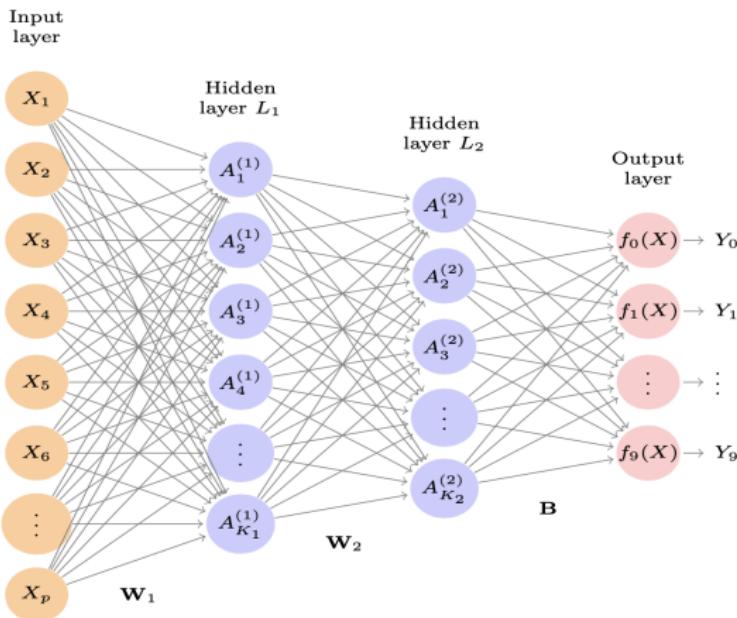


FIGURE 10.4. Neural network diagram with two hidden layers and multiple outputs, suitable for the MNIST handwritten-digit problem. The input layer has $p = 784$ units, the two hidden layers $K_1 = 256$ and $K_2 = 128$ units respectively, and the output layer 10 units. Along with intercepts (referred to as biases in the deep-learning community) this network has 235,146 parameters (referred to as weights).

Multilayer neural networks

From single layer with many neurons to multilayer with less neurons

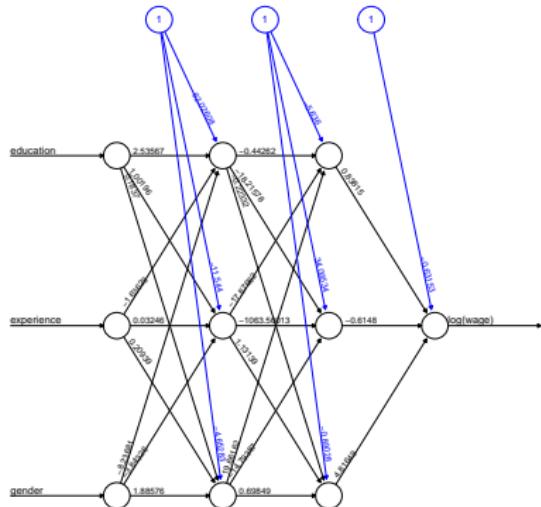
James et al. (2021):

- In theory a single hidden layer with a large number of units/neurons has the ability to approximate most functions
- However, the learning task of discovering a good solution is made much easier with multiple layers each of modest size
- Modern neural networks typically have more than one hidden layer, and often many units/neurons per layer

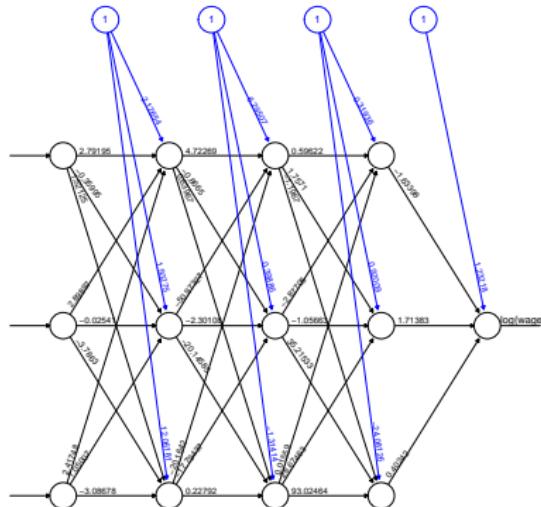
Deep Neural Networks = Multilayer Neural Networks

Application 1: Mincer equation

```
1 nn=neuralnet(log(wage)~education+experience+gender, data=  
    CPS1985, hidden=c(3,3), threshold=.05)  
2 plot(nn)  
3 nn=neuralnet(log(wage)~education+experience+gender, data=  
    CPS1985, hidden=c(3,3,3), threshold=.05)  
4 plot(nn)
```



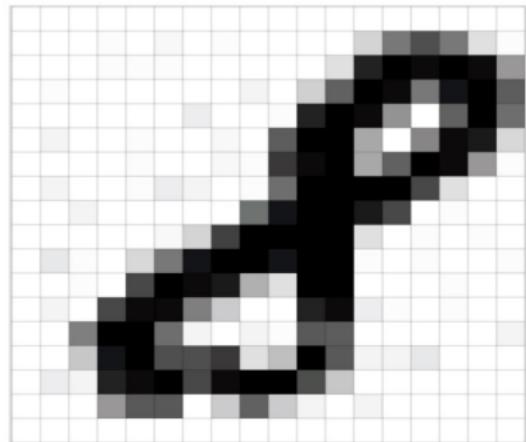
Error: 47.665751 Steps: 23831



Error: 50.041345 Steps: 5234

Pattern recognition

Everything is just numbers:



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0	
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	
0	0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	
85	237	253	246	255	210	21	1	0	1	0	0	0	6	2	4	0	0	
6	23	212	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Source: internet, link

A 18x18 pixel image can be seen as an array of 324 numbers that represent how dark each pixel is (grayscale value in (0, 255))

A vector of these numbers can be used to feed a neural networks

MNIST handwritten digit dataset

0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9



FIGURE 10.3. Examples of handwritten digits from the MNIST corpus. Each grayscale image has 28×28 pixels, each of which is an eight-bit number (0–255) which represents how dark that pixel is. The first 3, 5, and 8 are enlarged to show their 784 individual pixel values.

Source: James et al. (2021)

- Input vector X : $p = 28 \times 28 = 784$ pixels
- Output Y : class label $Y = (Y_0, Y_1, \dots, Y_{10})$
- 60,000 training images and 10,000 test images

Application 2: Handwritten digit recognition

```
1 # Source: section 10.9.2 in James et al. (2021)
2 library(keras)
3 # load the MNIST digit data
4 mnist <- dataset_mnist()
5 x_train <- mnist$train$x
6 g_train <- mnist$train$y
7 x_test <- mnist$test$x
8 g_test <- mnist$test$y
9 # reshape images into matrices
10 x_train <- array_reshape(x_train, c(nrow(x_train), 784))
11 x_test <- array_reshape(x_test, c(nrow(x_test), 784))
12 y_train <- to_categorical(g_train, 10)
13 y_test <- to_categorical(g_test, 10)
14 # rescale to the unit interval
15 x_train <- x_train / 255
16 x_test <- x_test / 255
17 # define the multilayer NN
18 modelnn <- keras_model_sequential()
19 modelnn %>%
20   layer_dense(units = 256, activation = "relu",
21               input_shape = c(784)) %>%
22   layer_dropout(rate = 0.4) %>%
23   layer_dense(units = 128, activation = "relu") %>%
24   layer_dropout(rate = 0.3) %>%
25   layer_dense(units = 10, activation = "softmax")
26 summary(modelnn)
27 # add details to the model
28 modelnn %>% compile(loss = "categorical_crossentropy",
29                         optimizer = optimizer_rmsprop(), metrics = c("accuracy"))
30 )
```

Application 2: Handwritten digit recognition

```
31 # fit the NN with training data
32 system.time(
33   history <- modelnn %>%
34     fit(x_train, y_train, epochs = 30, batch_size = 128,
35       validation_split = 0.2)
36 )
37 plot(history, smooth = FALSE)
38 # obtain the test error
39 accuracy <- function(pred, truth)
40   mean(pred == drop(truth))
41 modelnn %>% predict(x_test) %>% max.col %>% accuracy(g_test+1)
42
43 # fit a multinomial logit as a NN without hidden layer
44 modellr <- keras_model_sequential() %>%
45   layer_dense(input_shape = 784, units = 10,
46               activation = "softmax")
47 summary(modellr)
48 modellr %>% compile(loss = "categorical_crossentropy", optimizer =
49   optimizer_rmsprop(), metrics = c("accuracy"))
50 modellr %>% fit(x_train, y_train, epochs = 30, batch_size = 128, validation_
51   split = 0.2)
50 modellr %>% predict(x_test) %>% max.col %>% accuracy(g_test+1)
```

You may need to install Keras first:

```
1 install.packages("tensorflow")
2 install.packages("keras")
3 library(keras)
4 tensorflow::install_tensorflow()
5 tensorflow::tf_config()
6 install_keras()
```



Multilayer NN for handwritten digit recognition

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

TABLE 10.1. Test error rate on the **MNIST** data, for neural networks with two forms of regularization, as well as multinomial logistic regression and linear discriminant analysis. In this example, the extra complexity of the neural network leads to a marked improvement in test error.

Source: James et al. (2021)

- NN with 2 hidden layers L_1 (256 units) and L_2 (128 units)
- 235,146 coef in the NN and 7,065 in the multinomial logit⁵⁶
- To avoid overfitting, two forms of regularization are used

⁵⁶ $L_1: 785 \times 256 = 200,960$ and $L_2: 257 \times 128 = 32,896$ and 10 outputs 129×10



Dropout regularization

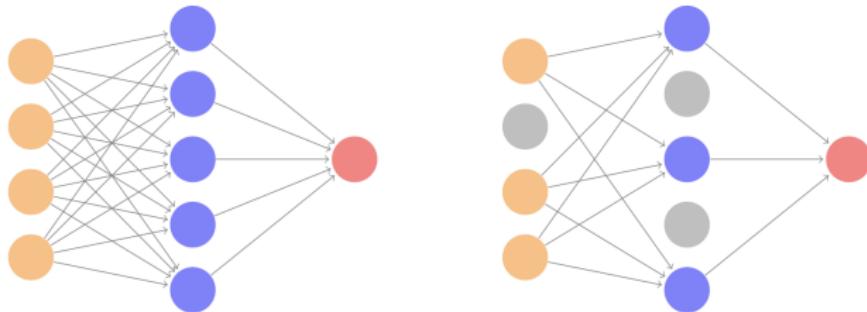


FIGURE 10.19. *Dropout Learning. Left: a fully connected network. Right: network with dropout in the input and hidden layer. The nodes in grey are selected at random, and ignored in an instance of training.*

Source: James et al. (2021)

- New efficient form of regularization, inspired by random forest
- Randomly remove a fraction of the units in a layer
- In practice, randomly set the dropped out units to zero

Limitations

Multilayer NN can model complex non-linear relationships

With very complex problems, such as detecting hundreds of types of objects in high-resolution images, we need to train deeper NN:

- perhaps 10 layers, each with hundreds of neurons, connected by hundreds of thousands of connections
- training a fully-connected DNN is very slow
- severe risk of overfitting with millions of parameters
- gradients problems make lower layers very hard to train

Solutions:

- Convolutional Neural Networks (CNN or ConvNets)
- Recurrent Neural Networks (RNN)

Convolutional Neural Networks

Pattern recognition

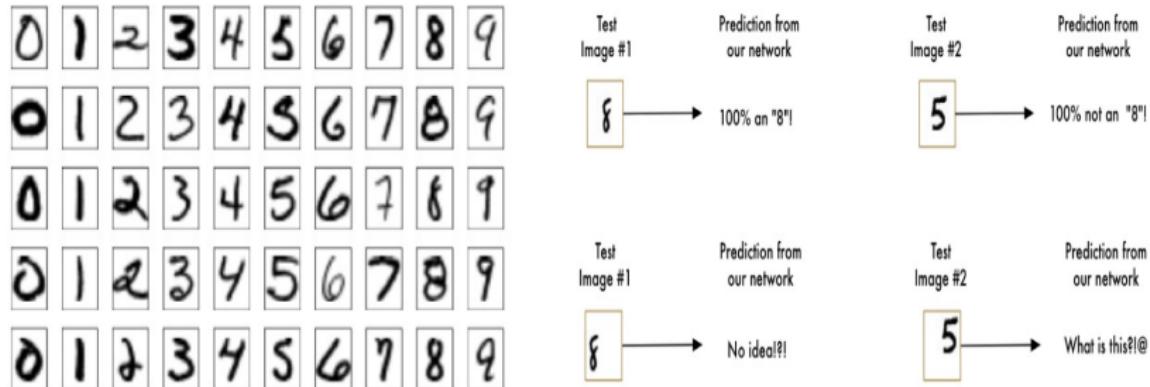


FIGURE 11.9. Examples of training cases from ZIP code data. Each image is a 16×16 8-bit grayscale representation of a handwritten digit.

The network fails to recognize '8' when the letter is not centered

→ translation, scale and (small) rotation invariances are needed

The solution is convolution

Convolutional Neural Network (CNN or ConvNet)⁵⁸

Step 1: Break the image into overlapping image tiles and, feed each image tile into a **small** neural network **with the same weights**⁵⁷



- It remains to use a **sliding window** over the entire picture
- using the same small NN reduces the number of weights
- same neural networks weights \equiv filter or convolution kernel

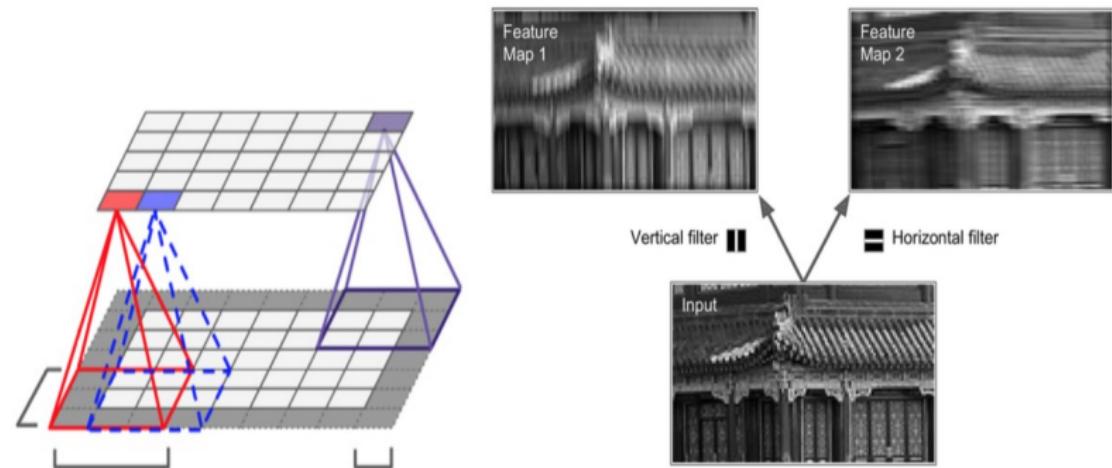
⁵⁷and the same activation function, $\text{ReLU} = \max(0, \text{input})$, \tanh or sigmoid

⁵⁸Source: Adam Geitgey [link](#), Ujjwal Karn [link](#), Andrej Karpathy [link](#)



CNN: The convolution step 1

CNN exploit spatially local correlation: each neuron is locally-connected (to only a small region of the input volume)



Source: Géron (2017)

- Different values of weights will produce different feature maps
- The convolution step plays like a filter
- Different filters can detect different features from an image⁵⁹

⁵⁹as for instances edges, curves, ...

CNN: The convolution step 1

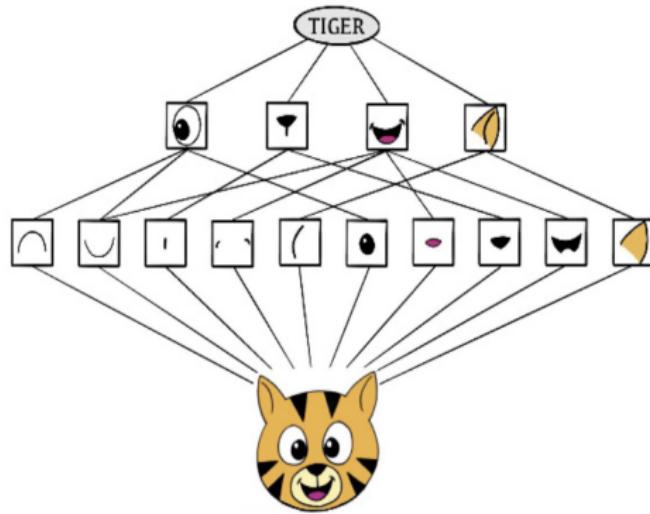
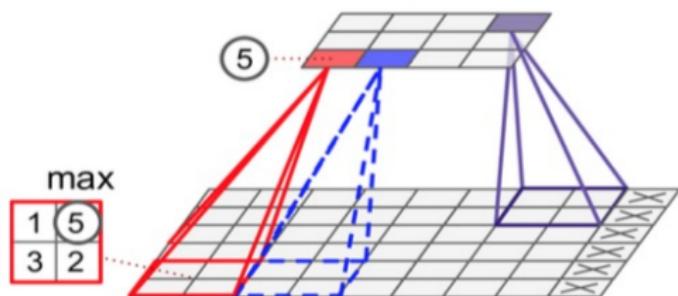


FIGURE 10.6. Schematic showing how a convolutional neural network classifies an image of a tiger. The network takes in the image and identifies local features. It then combines the local features in order to create compound features, which in this example include eyes and ears. These compound features are used to output the label “tiger”.

Source: James et al. (2021)

CNN: The pooling step 2

Step 2: Reduce the size of the array, using a **pooling** algorithm.



2x2 pooling layer, no padding



Source: Géron (2017)

The pooling step reduces the dimensionality of each feature map but retains the most important information⁶⁰

Pooling can be of different types: Max, Average, Sum etc.

⁶⁰It is also called subsampling or downsampling step

CNN: The pooling step 2

The function of pooling is to progressively reduce the spatial size of the input representation. In particular, pooling:

- makes the input representations smaller and more manageable
- reduces the number of weights and links in the network, therefore, controlling overfitting
- makes the network invariant to small transformations, distortions and translations in the input image⁶¹
- helps us arrive at an almost scale invariant representation of our image⁶²

⁶¹a small distortion in input will not change the output of Pooling – since we take the maximum/average value in a local neighborhood

⁶²This is very powerful since we can detect objects in an image no matter where they are located

CNN: The classification step 3

Step 3: Make a final prediction with a **fully-connected** network

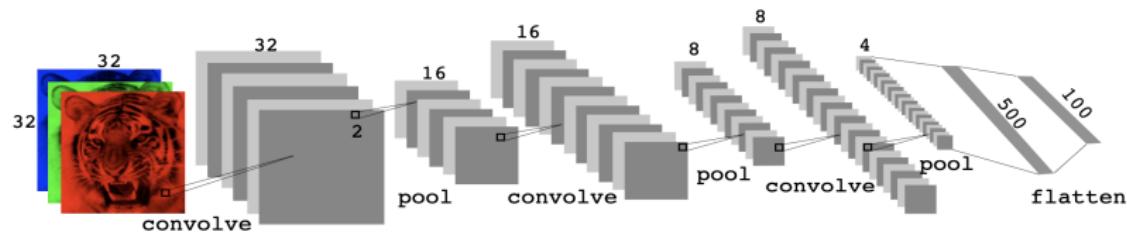


FIGURE 10.8. Architecture of a deep CNN for the **CIFAR100** classification task. Convolution layers are interspersed with 2×2 max-pool layers, which reduce the size by a factor of 2 in both dimensions.

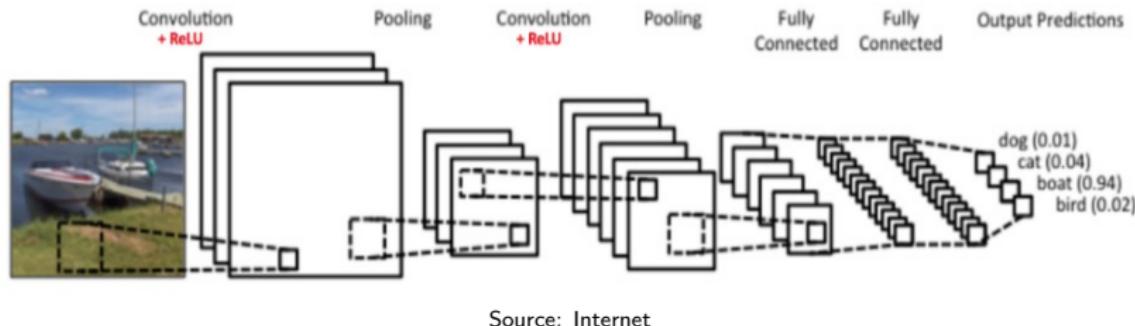
Source: James et al. (2021)

Feature extraction: use even more steps (hidden-layers) to extract the useful features from the images. The more convolution steps you have, the more complicated features your network will be able to learn to recognize.

Classification: The purpose of the Fully Connected layer is to use the high-level features for classifying the input image into classes

CNN: Intuitive principle

A hierarchy of representations with increasing level of abstraction:



Source: Internet

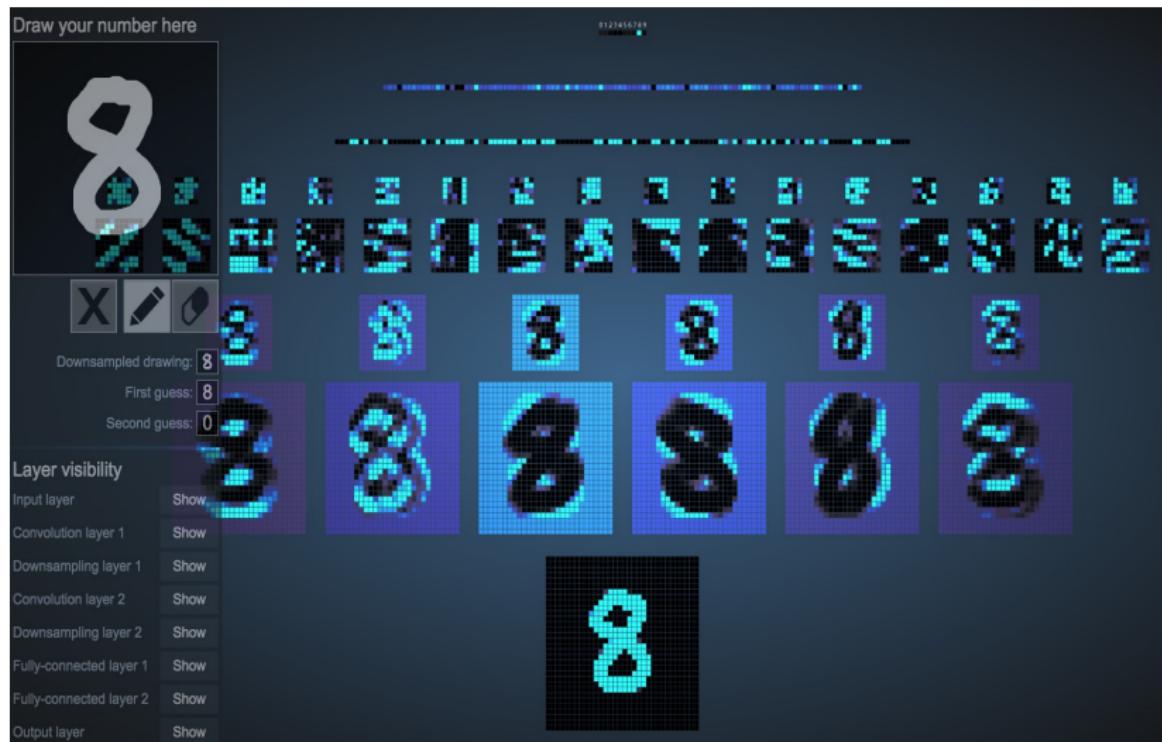
- Extract local features that depend on small subregions of the image
- Information from these features are merged to detect higher-order features
→ construction of complex objects from elementary parts

Image recognition: pixel → edge → texton → motif → part → object

Text: character → word → word group → clause → sentence → story

Speech: sample → spectral band → sound → ... → phoneme → word

CNN: An example on number recognition



To understand how ConvNet works, play with this animation

[link](#)

CNN: Performance in practice

TABLE 11.1. Test set performance of five different neural networks on a handwritten digit classification example (Le Cun, 1989).

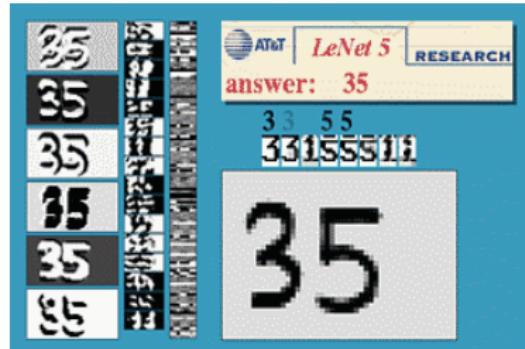
	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Source: Hastie et al. (2016)

- Convolutional Neural Networks outperform other methods
- The number of weights in Net-5 is much less than in Net-1
- ConvNet has been "a revolution in Artificial Intelligence"

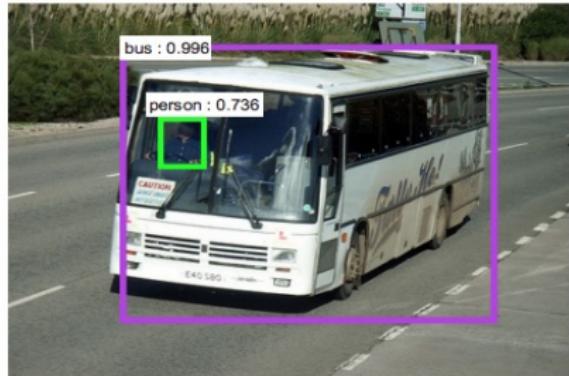
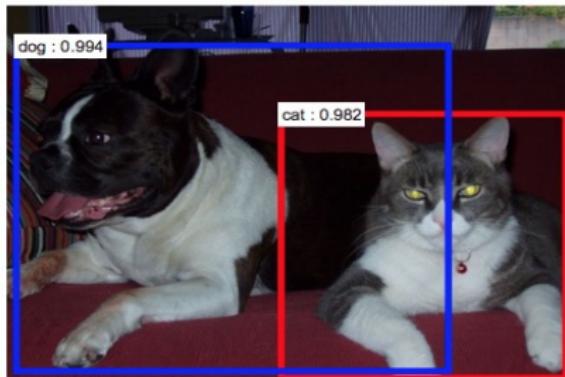
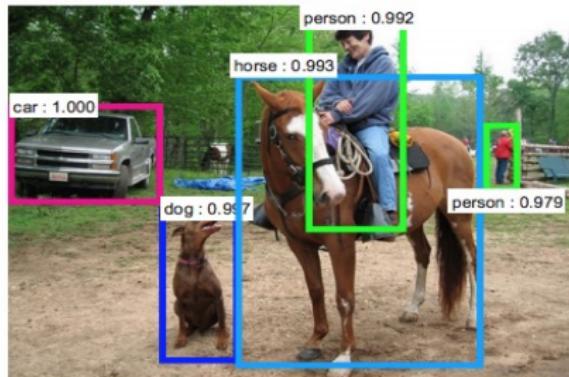
See the inaugural lesson of Yann LeCun at the College de France, in English [▶ en](#) or in French [▶ fr](#), and the review paper in Nature [▶ pdf](#)

CNN: Detection in complex cases



See this animation [link](#)

CNN: Detection in complex cases



Recurrent Neural Networks

Nature of the data

Many data sources are sequential in nature:

- In text analysis, the sequence and relative position of words capture the narrative, theme and tone → [Document classification, sentiment analysis and language translation](#)
- Time series of temperature, rainfall, wind speed, air quality and so on → [Weather forecast](#)
- Time series of market indices, stock and bond prices and exchange rates → [Financial forecasting](#)

In Recurrent Neural Network, the input object X is a sequence

Recurrent Neural Networks (RNN)

Neural network with a single hidden layer, for $t = 1, \dots, T$:

$$y_t \approx \alpha + A_t \beta$$

$$A_t = [f(\alpha_1 + X_t \delta_1), \dots, f(\alpha_M + X_t \delta_M)]$$

A linear combination of a nonlinear fct of linear combinations of X_t

Recurrent neural network:

- Each time series provides many short mini-series of input sequences $X = \{X_1, \dots, X_L\}$ of L periods, and a target Y

$$A_t = [f(\alpha_1 + X_t \delta_1 + A_{t-1} \gamma_1), \dots, f(\alpha_M + X_t \delta_M + A_{t-1} \gamma_M)]$$

- Identical weights for each sequence: α, δ, γ independent of t
- A form of weight sharing similar to the use of filters in CNN

Recurrent Neural Networks

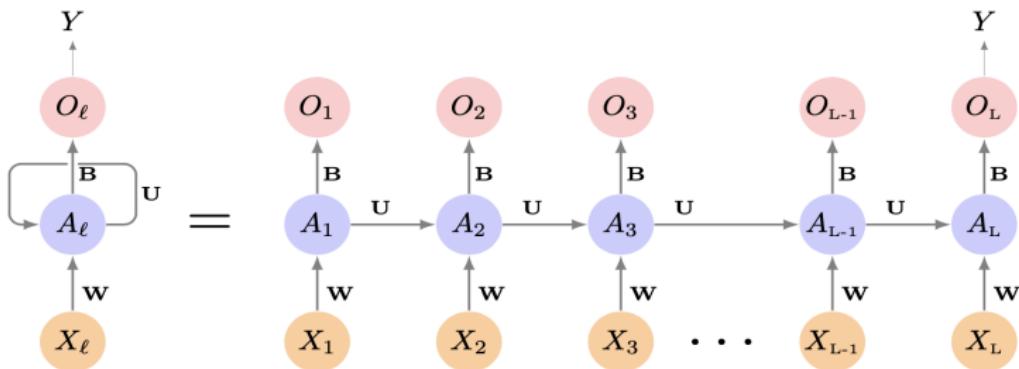


FIGURE 10.12. Schematic of a simple recurrent neural network. The input is a sequence of vectors $\{X_\ell\}_1^L$, and here the target is a single response. The network processes the input sequence X sequentially; each X_ℓ feeds into the hidden layer, which also has as input the activation vector $A_{\ell-1}$ from the previous element in the sequence, and produces the current activation vector A_ℓ . The same collections of weights \mathbf{W} , \mathbf{U} and \mathbf{B} are used as each element of the sequence is processed. The output layer produces a sequence of predictions O_ℓ from the current activation A_ℓ , but typically only the last of these, O_L , is of relevance. To the left of the equal sign is a concise representation of the network, which is unrolled into a more explicit version on the right.

RNN: Number of lags, training and test data

Recurrent neural network with one hidden layer, for $T = 1, \dots, T$:

$$y_t \approx \alpha + A_t \beta$$

$$A_t = [f(\alpha_1 + X_t \delta_1 + A_{t-1} \gamma_1), \dots, f(\alpha_M + X_t \delta_M + A_{t-1} \gamma_M)]$$

- Past values of y_t and other covariates can be used in X_t
- Select a number of lags L with care, perhaps using CV
- Extract many short series of (y, X) with a predefined length L
- Each short serie can be used to predict one value y_t
- The training data consists of n separate series of length L
- The test data consists of the remaining series of length L
- Find the set of coefficients minimizing the SSR (subject to a constraint) based on the training and test data

RNN: Historical trading time series on the NYSE

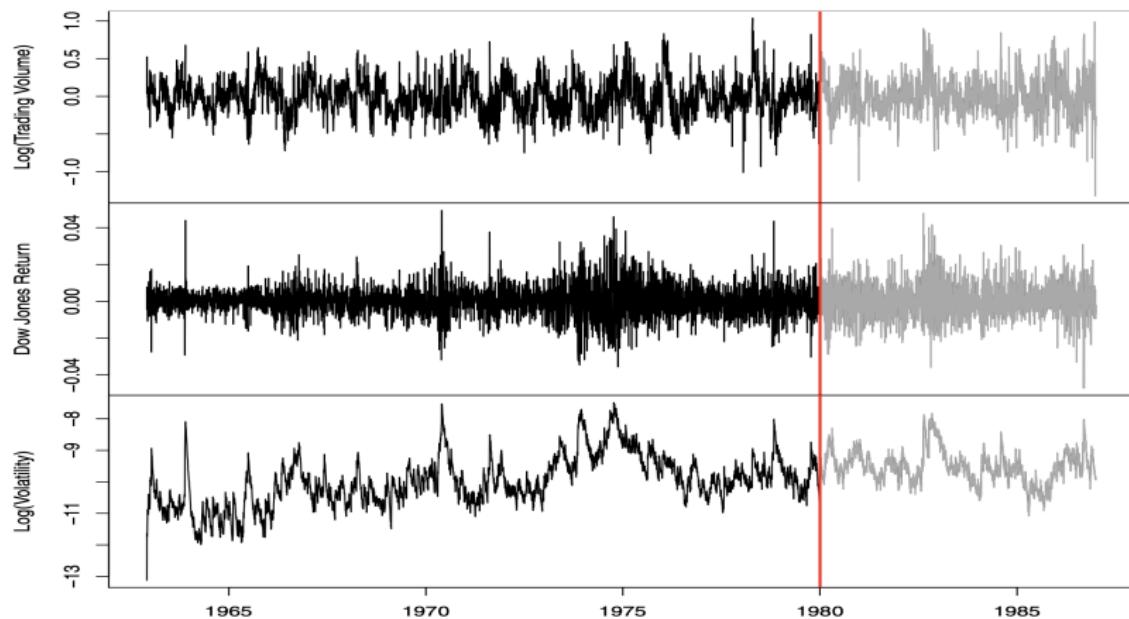
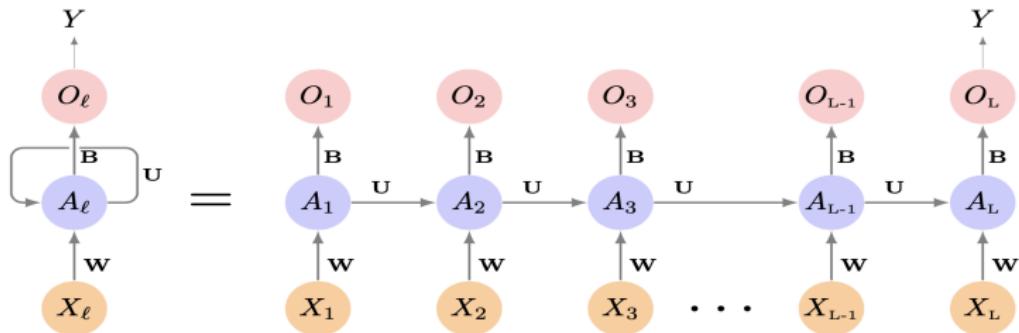


FIGURE 10.14. Historical trading statistics from the New York Stock Exchange. Daily values of the normalized log trading volume, DJIA return, and log volatility are shown for a 24-year period from 1962–1986. We wish to predict trading volume on any day, given the history on all earlier days. To the left of the red bar (January 1, 1980) is training data, and to the right test data.

RNN: Forecast trading volume based on past history



How do we represent this problem in terms of the structure displayed in Figure 10.12? The idea is to extract many short mini-series of input sequences $X = \{X_1, X_2, \dots, X_L\}$ with a predefined length L (called the *lag* in this context), and a corresponding target Y . They have the form

$$X_1 = \begin{pmatrix} v_{t-L} \\ r_{t-L} \\ z_{t-L} \end{pmatrix}, \quad X_2 = \begin{pmatrix} v_{t-L+1} \\ r_{t-L+1} \\ z_{t-L+1} \end{pmatrix}, \dots, \quad X_L = \begin{pmatrix} v_{t-1} \\ r_{t-1} \\ z_{t-1} \end{pmatrix}, \quad \text{and } Y = v_t. \quad (10.20)$$

So here the target Y is the value of `log_volume` v_t at a single timepoint t , and the input sequence X is the series of 3-vectors $\{X_\ell\}_1^L$ each consisting of the three measurements `log_volume`, `DJ_return` and `log_volatility` from day $t - L$, $t - L + 1$, up to $t - 1$. Each value of t makes a separate (X, Y)

RNN: Autocorrelation function

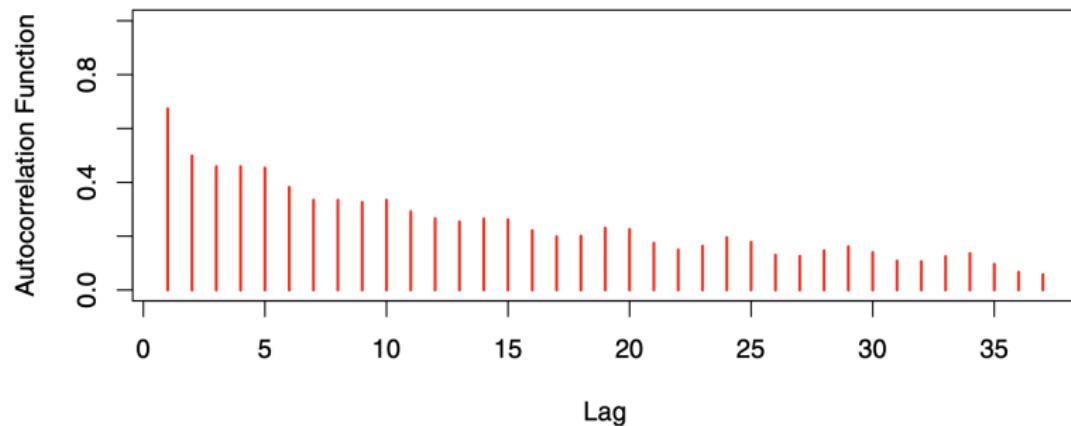


FIGURE 10.15. The autocorrelation function for `log_volume`. We see that nearby values are fairly strongly correlated, with correlations above 0.2 as far as 20 days apart.

Source: James et al. (2021)

- $T = 6051$, $L = 5$, so 6046 short series (y, X) are available
- fit the model with 12 neurons and using 4281 training series
- forecast 1765 values after January 2, 1980

RNN: Forecast of log trading volume on the NYSE

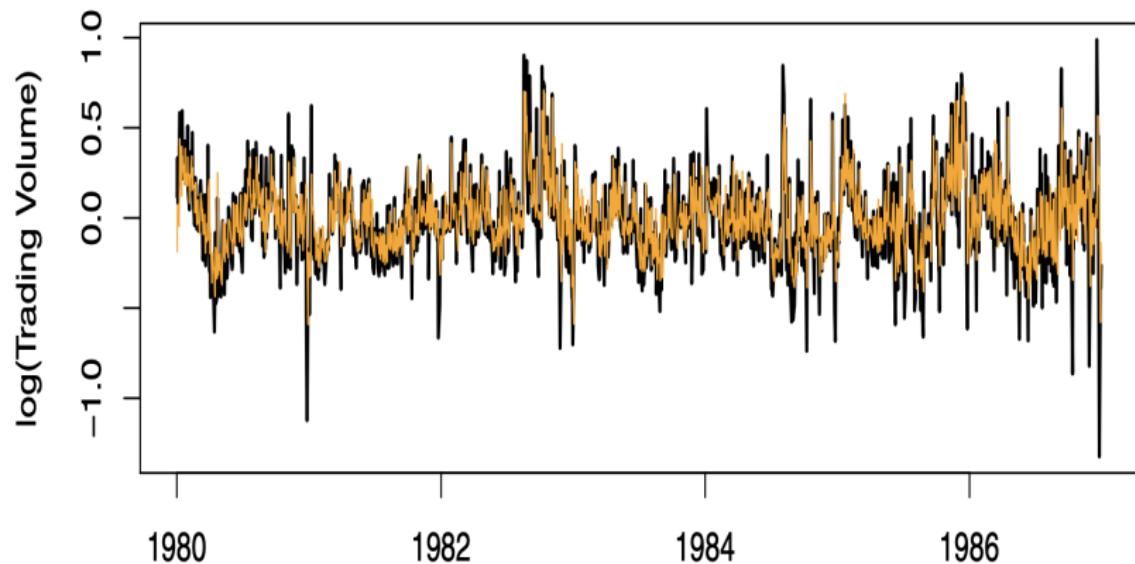


FIGURE 10.16. RNN forecast of `log_volume` on the NYSE test data. The black lines are the true volumes, and the superimposed orange the forecasts. The forecasted series accounts for 42% of the variance of `log_volume`.

RNN and AR models

Recurrent neural network with a single hidden layer:

$$y_t \approx \alpha + A_t \beta$$

$$A_t = [f(\alpha_1 + X_t \delta_1 + A_{t-1} \gamma_1), \dots, f(\alpha_M + X_t \delta_M + A_{t-1} \gamma_M)]$$

- Lag of the dependent variable y_{t-1} can be used in X_t
- With $M = 1$, f linear and $X_t = y_{t-1}$, we have an AR(L):

$$y_t \approx \beta_0 + y_{t-1} \beta_1 + \cdots + y_{t-L} \beta_{t-L}$$

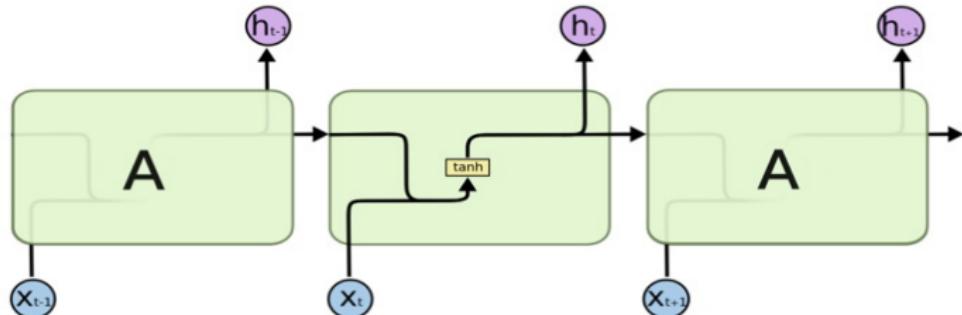
- RNN and AR models have much in common
- By combining nonlinear functions ($M > 1$ and f nonlinear), RNN add more flexibility → nonlinear and interaction effects

LSTM: Long and Short Term Memory model

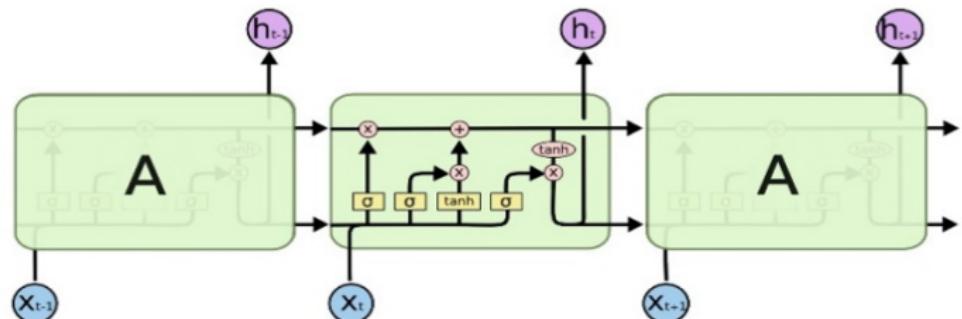
My son is a manga fan, so our next holiday will be in ...

- RNN don't predict Japan, since it doesn't remember *manga*
- RNN main limitation: short term memory
- Solution: Combine 2 hidden layers, one with short memory and the other one with longer memory
- LSTM combine a long-term state c and a short-term state h

LSTM vs. RNN



Cellule RNN Vanilla



LSTM (Long Short Term Memory)

LSTM

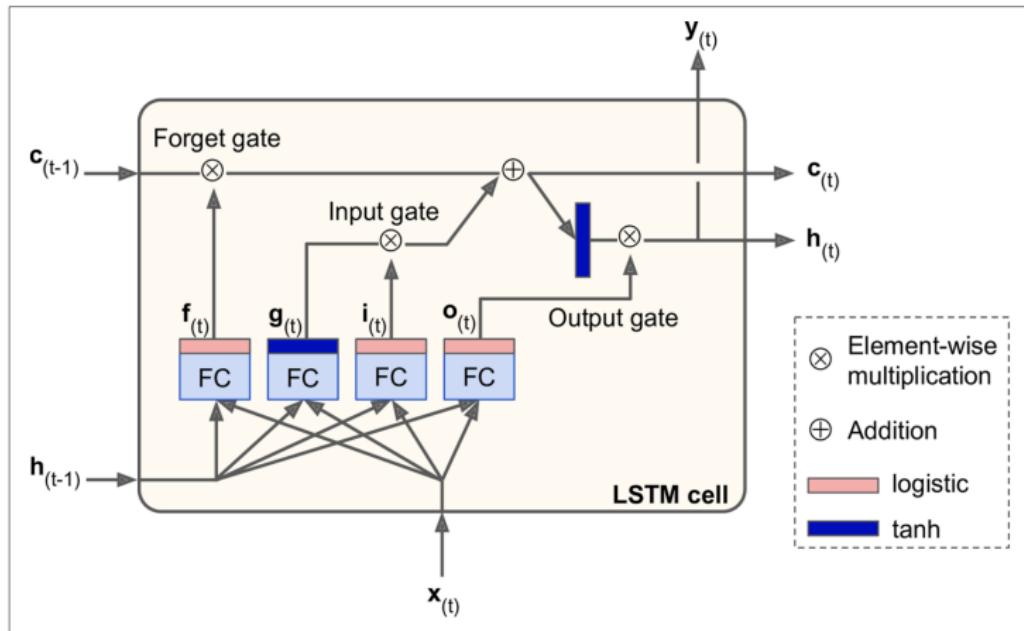


Figure 14-13. LSTM cell

Géron (2017)

- c : drop some memories \otimes and add some new memories \oplus

3. Using ML methods in Econometrics

- Misspecification detection
- Causal inference

General Principle

Machine Learning: solve the optimization problem

$$\underset{m}{\text{Minimize}} \sum_{i=1}^n \underbrace{\mathcal{L}(y_i, m(X_i))}_{\text{loss function}} + \underbrace{\lambda \|m\|_{\ell_q}}_{\text{penalization}}$$

- Choice of the **loss function**:
 - \mathcal{L} → conditional mean, quantiles, classification
 - m → linear, splines, tree-based models, neural networks
- Choice of the **penalization**:
 - ℓ_q → lasso, ridge
 - λ → over-fitting, under-fitting, cross validation

Ridge and Lasso

$$\underset{\beta}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - X_i \beta)^2 + \lambda \sum_{j=2}^p |\beta_j|^q$$

It is equivalent to minimize SSR subject to $\sum_{j=2}^p |\beta_j|^q \leq c$

- The constraint restricts the magnitude of the coefficients
- It shrinks the coefficients towards zero as $c \searrow$ (or $\lambda \nearrow$)
- Add some bias if it leads to a substantial decrease in variance
- $q = 2$: Ridge, $\hat{\beta} = (X^\top X + \lambda \mathbb{I}_n)^{-1} X^\top y$ is defined with $p \gg n$
- $q = 1$: Lasso sets some coef exactly to 0, variable selection

→ High-dimensional problems ($p \gg n$)

Random Forest, Boosting, Deep learning

$$\underset{m}{\text{Minimize}} \quad \sum_{i=1}^n (y_i - m(X_i))^2 + \lambda \int m''(x)^2 dx$$

It is equivalent to minimize SSR subject to $\int m''(x)^2 dx \leq c$

- A fully nonparametric model: $y \approx m(X_1, \dots, X_p)$
- The constraint restricts the flexibility of m
- Choice of m : Random forest, boosting or deep learning
- Similar to nonparametric econometrics (splines)
- Appropriate with many covariates (no curse of dimensionality)

→ Complex functional form

Why and how to use ML methods in Econometrics?

Pros:

- High-dimensional problems
- Complex functional forms

However,

- Black-box models
- Prediction is not causation

3. Using ML methods in Econometrics

- Misspecification detection
- Causal inference

A major criticism to econometrics

Léo Breiman (*Statistical Science*, 2001):

Upon my return, I started reading the *Annals of Statistics*, the flagship journal of theoretical statistics, and was bemused. Every article started with

Assume that the data are generated by the following model: ...

wavelet theory. Even in applications, data models are universal. For instance, in the *Journal of the American Statistical Association* (JASA), virtually every article contains a statement of the form:

Assume that the data are generated by the following model: ...

... an uncritical use of data models.

Misspecification can lead to wrong conclusions

- Let us assume that the true regression function is:

$$y = \beta_0 + \beta_1 x + \beta_3 x^3 + \varepsilon \quad (5)$$

- A parametric test of the following hypotheses:

$$H_0 : y = \beta_0 + \beta_1 x + \varepsilon \quad \text{vs.} \quad H_1 : y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$$

may not reject the null, since $\beta_2 = 0$ is true in (5)

- To the opposite, a test statistic based on

$$H_0 : y = \beta_0 + \beta_1 x + \varepsilon \quad \text{vs.} \quad H_1 : y = m(x) + \varepsilon$$

would likely reject the null

- A nonparametric model is more appropriate under H_1

How machine learning tools may help econometrics?

- Parametric model:

$$y = X\beta + \varepsilon$$

- Fully-nonparametric model:

$$y = m(X) + \varepsilon$$

- Is the parametric regression model correctly specified?
 - If no, ML methods should outperform OLS estimation
 - If yes, ML methods should not outperform OLS estimation
- ML can be used to detect misspecification

Application 1: Boston housing prices

- Boston housing dataset: 14 variables (2 dummies), 506 observations⁶³
- OLS in a linear regression model, $p=13$

$$\text{medv} = X\beta + \varepsilon$$

- Lasso with squares, cubes and pairwise interactions, $p=117$

$$\text{medv} = X\beta_1 + X^2\beta_2 + X^3\beta_3 + (X:X)\beta_4 + \varepsilon$$

- Random Forest and Boosting in a nonparametric model, $p=13$

$$\text{medv} = m(X) + \varepsilon$$

- We compute the MSE by 10-folds Cross-Validation

⁶³ $X = [\text{chas}, \text{nox}, \text{age}, \text{tax}, \text{indus}, \text{rad}, \text{dis}, \text{lstat}, \text{crim}, \text{black}, \text{rm}, \text{zn}, \text{ptratio}]$

Application 1: Boston housing prices

```
1 library(MASS); library(randomForest); library(gbm); library(glmnet)
2 data(Boston) ; nobs=nrow(Boston)
3 set.seed(12345) ; nfold=10
4 Kfold=cut(seq(1,nobs), breaks=nfold , labels=FALSE)
5 mse.test=matrix(0, nfold ,4)
6 # generate X^2 X^3 and pairwise interactions for the Lasso
7 Xcol=colnames(Boston)[-14]
8 Xsqr=paste0("I(",Xcol,"^2)", collapse="+") # squared covariates
9 Xcub=paste0("I(",Xcol,"^3)", collapse="+") # cubic covariates
10 fmla=paste0("medv~(.)^2+",Xsqr,"+",Xcub)
11 X=model.matrix(as.formula(fmla), data=Boston)[,-1]
12 y=Boston[,14]
13 mysample=sample(1:nobs) # random sampling (permutation)
14 for(i in 1:nfold){ # K-fold CV
15   cat("K-fold loop: ", i, "\r")
16   test=mysample[which(Kfold==i)]
17   train=mysample[which(Kfold !=i)]
18   # OLS, Lasso, Random Forest, Boosting
19   fit.lm <- lm(medv~., data=Boston, subset=train)
20   fit.la <- cv.glmnet(X[train,], y[train], alpha=1)
21   fit.rf <- randomForest(medv~., data=Boston, subset=train, mtry=6)
22   fit.bo <- gbm(medv~., data=Boston[train,], distribution="gaussian",
23                 interaction.depth=6)
23   # out-sample MSE
24   mse.test[i,1]=mean((Boston$medv-predict(fit.lm,Boston))[-train]^2)
25   mse.test[i,2]=mean((y-predict(fit.la,X,s="lambda.min"))[-train]^2)
26   mse.test[i,3]=mean((Boston$medv-predict(fit.rf,Boston))[-train]^2)
27   mse.test[i,4]=mean((Boston$medv-predict(fit.bo,Boston))[-train]^2)
28 }
29 mse=colMeans(mse.test) # test error
30 round(mse, digits=2)
31 [1] 23.93 14.88 10.16 10.34
```



Application 1: Boston housing prices

- Boston housing dataset:⁶⁴

$\widehat{\mathcal{R}}^{10\text{-CV}}$	OLS	LASSO $_{x^2x^3int}$	R.Forest	Boosting
MSE	23.93	14.88	10.16	10.34

- Random Forest and Boosting show impressive improvement over OLS, in terms of predictive performance
- ML models are known to capture complex functional forms
- It suggests that the parametric model lacks important nonlinear and/or interaction effects
- Lasso provides substantial improvement over OLS, but is still less performant than Random Forest and Boosting. It suggests that some nonlinearities are still not well captured.

⁶⁴14 variables (2 dummies), 78 pairwise interactions, 506 observations



GamLa: An econometric model for interpretable ML

A partially linear model:

$$y = g_1(X_1) + \dots + g_p(X_p) + Z\gamma + \varepsilon$$

with Z a matrix of pairwise interactions $Z = (X_1X_2, \dots, X_{q-1}X_q)$.

The marginal effect is:

$$\frac{\partial y}{\partial X_j} = g'_j(X_j) + c$$

where c is a constant term which depends on the other covariates.

- Combine non-linearity in X_j and linear pairwise interactions
- The linearity assumption on interaction effects represents the price to pay to keep the model interpretable.

→ GamLa = GAM + variable selection (Lasso, Autometrics)⁶⁵

⁶⁵Flachaire, Hacheme, Hué, Laurent (2022)

GamLa: An econometric model for interpretable ML

A partially linear model:

$$y = g_1(X_1) + \dots + g_p(X_p) + Z\gamma + \varepsilon$$

- Estimation based on the **Double Residuals** (DR) method:
 - ① GAM of y on X_1, \dots, X_p : compute the residuals $\hat{\eta}_y$
 - ② GAM of Z_j on X_1, \dots, X_p : compute the residuals $\hat{\eta}_{z_j}$, $\forall j$
 - ③ LASSO of $\hat{\eta}_y$ on $\hat{\eta}_{z_1}, \dots, \hat{\eta}_{z_l} \rightarrow$ obtain $\hat{\gamma}$

An application of FWL to semiparametric regression models

- Robinson (1988) shows that with DR $\hat{\gamma}_{ols}$ is \sqrt{n} -consistent, even if $\hat{g}_1(X), \dots, \hat{g}_p(X)$ are consistent at slower rates
- Flachaire, Hacheme, Hué and Laurent (2022) show that using the DR approach is crucial to select correctly the interactions⁶⁶

⁶⁶So don't use the `gamlasso` function in the R package `plsmselect`!

Application 1: Boston housing prices

- Boston housing dataset:

$\widehat{\mathcal{R}}^{10-\text{CV}}$	OLS	LASSO $_{x^2x^3int}$	R.Forest	GamLa
MSE	23.93	14.88	10.16	9.73

- GamLa shows impressive improvement over OLS, in terms of predictive performance
- GamLa performs as well as Random Forest and Boosting⁶⁷
- It suggests that parametric models are outperformed by ML models when they lack important nonlinear and/or interaction effects only

⁶⁷ Model Confidence Set (MCS) test can be used to test if the MSE are significantly different (Hansen, Lunde and Nason 2011) [► pdf](#) Pairwise AUC can be used in classification (Candelon, Dumitrescu and Hurlin 2012) [► pdf](#)

Conclusion

- Many results report that ML outperform parametric models in terms of predictive performance
- ML models outperform standard parametric model ... which are not well-specified!
- ML methods can help to detect and correct misspecification in parametric regression
- Parametric models can perform as well as ML models!

3. Using ML methods in Econometrics

- Misspecification detection
- Causal inference

Prediction is not causation

Kleinberg et al. (2015) *Prediction policy problems*

- Many policy applications where causal inference is not central
- Hips or knees replacement: costly, painful, recovery takes time
- Policy decision: predicting the riskiest patients⁶⁸

Athey (2017) *Beyond prediction: Using big data for policy problems*

- Pure prediction methods are not helpful for causal problems
- Which patients should be given priority to receive surgery?
- Estimating heterogeneity in the effect of surgery is required

⁶⁸ML are used to predict the probability that a candidate would die within a year from other causes. Identify high risk patients who shouldn't receive surgery



High-dimensional parametric framework

Inference on target regression coefficients

Our main concern is the estimation and inference on α in a high-dimensional framework:

$$y = d\alpha + X\beta + \varepsilon$$

- d is a target regressor as treatment, policy or other variable
- X may contain many variables, a few of them are important
- With sparsity, a variable selection method is used in a 1st step
- Since Lasso shrinks coefficients towards zero, coef are biased
- Correct this bias using an additional (unrestricted) estimation

→ Post-selection estimation and inference

The problem of post-selection inference

- Single Selection: OLS of y on d and the selected variables X^*

$$y = d\alpha + X^*\beta + \varepsilon$$

- Unbiased $\hat{\alpha}$... if the *true* model is selected only!
- Problem: mistakes from the variable selection can introduce omitted variable bias
 - one covariate X_j strongly correlated to d without a strong effect on y may be omitted in the variable selection process
- Ignoring variable selection uncertainty may be misleading
 - Naive post-selection estimation may be biased

Post-selection inference: Double selection

Our main concern is estimation and inference on α in

$$y = d\alpha + X\beta + \varepsilon$$

- **Double Selection:**⁶⁹

- ① Lasso of y on X : select variables important to predict y
- ② Lasso of d on X : select variables important to predict D

OLS of y on d and the union of the selected variables

$$y = d\alpha + X^{**}\beta + \varepsilon$$

- Idea: give a 2nd chance to omitted variables in the first Lasso
- $\hat{\alpha}$ is immunized against variable selection mistakes

→ valid post-selection inference in high-dimensions

⁶⁹Belloni, Chernozhukov and Hansen (2014) ▶ pdf Uniformly valid confidence set for α despite imperfect model selection, and full efficiency for estimating α



Post-selection inference: Partialling out

Our main concern is estimation and inference on α in

$$y = d\alpha + X\beta + \varepsilon$$

- **Partialling out:**

- ① Lasso of y on X : compute the residuals $\hat{\eta}_y$
- ② Lasso of d on X : compute the residuals $\hat{\eta}_d$

OLS of $\hat{\eta}_y$ on $\hat{\eta}_d$ (double residuals approach)

$$\hat{\eta}_y = \hat{\eta}_d\alpha + \varepsilon$$

- Idea: an application of the Frisch-Waugh-Lovell theorem⁷⁰

→ Partialling out and double selection are quite similar⁷¹

⁷⁰But $\hat{\alpha}_{ols}$ is different in the two models due to lasso variable selections

⁷¹From the FLW theorem, the double selection estimator of α is equal to the OLS estimator of the residuals of y on X^{**} on the residuals of d on X^{**} .

Threshold selection: Rigorous Lasso

- The choice of the penalization parameter λ is crucial
- Optimal λ for prediction and estimation are different
- CV targets prediction and lacks theoretical foundations
- Theoretical grounded and feasible selection for estimation:⁷²

$$\lambda = 2c\sqrt{n}\hat{\sigma}\Phi^{-1}(1 - \gamma/(2p))$$

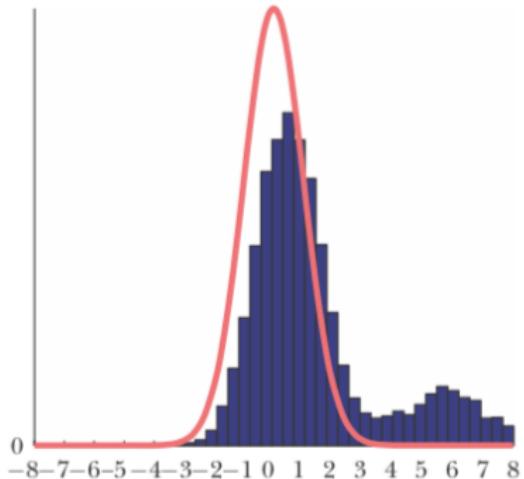
in the case of homoskedasticity

- Another selection is proposed in the heteroskedasticity case

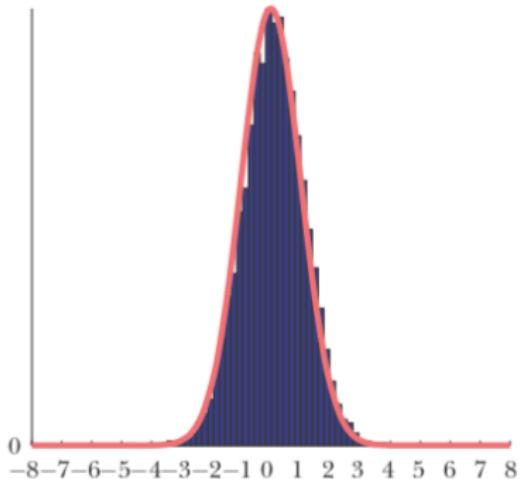
⁷²See Belloni, Chernozhukov and Hansen (2014) [pdf](#) $c = 1.1$ for post-Lasso and $c = 0.5$ for Lasso, $\gamma = .1$ by default

Bias of naive post-selection estimation

A: A Naive Post-Model Selection Estimator



B: A Post-Double-Selection Estimator



Source: Belloni, Chernozhukov, and Hansen (forthcoming).

Notes: The left panel shows the sampling distribution of the estimator of α based on the first naive procedure described in this section: applying LASSO to the equation $y_i = d_i + x_i' \theta_y + r_{y,i} + \zeta_i$ while forcing the treatment variable to remain in the model by excluding α from the LASSO penalty. The right panel shows the sampling distribution of the “double selection” estimator (see text for details) as in Belloni, Chernozhukov, and Hansen (forthcoming). The distributions are given for centered and studentized quantities.

Source: Belloni, Chernozhukov, Hansen (2014)

Application 1: Do poor countries catch up rich countries?

We are interested in the convergence hypothesis $\alpha < 0$ in

$$y = d\alpha + X\beta + \varepsilon$$

where y is the growth rate of GDP, d is the initial level of GDP and X contains many countries characteristics

- The parameter of interest is α
- We test the null hypothesis $H_0 : \alpha = 0$
- If H_0 is rejected and $\alpha < 0$: evidence of catch-up effect
- Covariate selection is crucial, since $p = 63$ and $n = 90$
- We use double selection and partialling out with rigorous Lasso
- Implementation is done with the R package `hdm`⁷³

⁷³see the vignette of the `hdm` package in R [pdf](#)

Application 1: Do poor countries catch up rich countries?

```
1 library(hdm)
2 data("GrowthData") # the 2nd column is a vector of one #
3 y=as.matrix(GrowthData)[,1,drop=F]
4 d=as.matrix(GrowthData)[,3,drop=F]
5 X=as.matrix(GrowthData)[,-c(1,2,3),drop=F]
6 # fit models
7 LS.fit=lm(y~d+X)
8 PO.fit=lassoEffect(X,y,d,method="partialling.out")
9 DS.fit=lassoEffect(X,y,d,method="double.selection")
10 # inference on coef of interest
11 LS=summary(LS.fit)$coefficients[2,]
12 PO=summary(PO.fit)$coefficients[1,]
13 DS=summary(DS.fit)$coefficients[1,]
14 rbind(ols=LS, double.selection=DS, partialling.out=PO)
```

	Estimate	Std. Error	t value	Pr(> t)
ols	-0.009377989	0.02988773	-0.31377	0.75601
double.selection	-0.050005855	0.01579138	-3.16665	0.00154
partialling.out	-0.049811465	0.01393636	-3.57420	0.00035

Application 1: Do poor countries catch up rich countries?

Inference on the parameter of interest α :

	Estimate	Std.Error	t value	Pr(> t)
OLS	-0.00938	0.02989	-0.31377	0.75601
Double selection	-0.05001	0.01579	-3.16665	0.00154
Partialling out	-0.04981	0.01394	-3.57420	0.00035

- $H_0 : \alpha = 0$ not rejected with OLS (large standard error)⁷⁴
- $H_0 : \alpha = 0$ rejected with double selection and partialling out
 - more precise estimate (smaller standard error)
 - greater magnitude of the coefficient

Poor countries tend to catch up rich countries!

- Note that Single Selection (naive post-selection) put $\alpha = 0$:

```
15 rlasso(y~d+X, post=TRUE)$coefficients [2]
```

⁷⁴It is not surprising given that $p = 63$ is comparable to $n = 90$.

Heterogeneous treatment effects: high-dimensions

If d is a treatment, we can consider heterogeneous effects as

$$y = d\alpha(X) + g(X) + \varepsilon$$

where $\alpha(X)$ and $g(X)$ are approximated by linear combinations of X or transformations of X , $\alpha(X) = Z_1\beta$ and $g(X) = Z_2\gamma$.⁷⁵

- The regression can be rewritten: $y = dZ_1\beta + Z_2\gamma + \varepsilon$
- Several variables of interest β
- Double Selection:
 - ① Lasso of y on Z_2 : select variables important to predict y
 - ② Lasso of each interaction dZ_1 on Z_2 : select important variables

OLS of y on d and the union of the selected variables

→ assess heterogeneity with many determinants

⁷⁵ Z_1 and Z_2 may include powers, b-splines, or interactions of X

Application 2: The effect of gender on wage

Several parameters of interest:

$$y = d\alpha + dX\beta + Z\gamma + \varepsilon$$

- y is the log of the wage, d is a dummy for female
- dX are the interactions between d and each covariate in X
- Z includes 2-ways interactions of the covariates $Z = [X, X:X]$
- The target variable is female d , in combination with other variables dX
- Our main interest is to make inference on α and β
 - If $\beta = 0$: homogeneous wage gender gap given by α
 - If $\beta \neq 0$: heterogeneous wage gender gap explained by X
- Data: US Census in 2012, $p = 116$ and $n = 29217^{76}$

⁷⁶for a recent application see Bach, Chernozhukov and Spindler (2021)



Application 2: The effect of gender on wage

```
1 library(hdm)
2 data(cps2012)
3 y <- cps2012$lnw
4 X <- model.matrix(~-1+female+female:(widowed+divorced+
+ separated+nevermarried+hsd08+hsd911+hsg+cg+ad+mw+so+we+
+ exp1+exp2+exp3)+(widowed+divorced+separated+nevermarried
+ +hsd08+hsd911+hsg+cg+ad+mw+so+we+exp1+exp2+exp3)^2,data
= cps2012)
5 X<-X[,which(apply(X,2,var)!=0)] #exclude constant variables
6 index.gender <- grep("female", colnames(X))
7 effects.female<-rlassoEffects(x=X,y=y, index=index.gender)
8 summary(effects.female)
```

Generic approach to generate all covariates:

```
9 Xcol=colnames(cps2012)[4:18]
10 dcol=colnames(cps2012)[3]
11 Xvar=paste(Xcol, collapse = "+")
12 Xint=paste("( ", paste(Xcol, collapse="+"), ")^2", sep="")
13 fmla=paste("~1+", dcol, "+", dcol, ":( ", Xvar, "+", Xint, sep="")
14 X<-model.matrix(as.formula(fmla), data=cps2012)
```

Application 2: The effect of gender on wage

```
15 > summary( effects.female)
16 [1] " Estimates and significance testing of the effect of
      target variables"
17
18 female              Estimate Std. Error t value Pr(>|t|) 
19 female:widowed       0.136095 0.090663 1.501 0.133325
20 female:divorced     0.136939 0.022182 6.174 6.68e-10 *** 
21 female:separated    0.023303 0.053212 0.438 0.661441
22 female:nevermarried 0.186853 0.019942 9.370 < 2e-16 *** 
23 female:hsd08         0.027810 0.120914 0.230 0.818092
24 female:hsd911        -0.119335 0.051880 -2.300 0.021435 *  
25 female:hsg           -0.012890 0.019223 -0.671 0.502518
26 female:cg            0.010139 0.018327 0.553 0.580114
27 female:ad            -0.030464 0.021806 -1.397 0.162405
28 female:mw            -0.001063 0.019192 -0.055 0.955811
29 female:so            -0.008183 0.019357 -0.423 0.672468
30 female:we            -0.004226 0.021168 -0.200 0.841760
31 female:exp1          0.004935 0.007804 0.632 0.527139
32 female:exp2          -0.159519 0.045300 -3.521 0.000429 *** 
33 female:exp3          0.038451 0.007861 4.891 1.00e-06 *** 
```

→ smaller gender gap for nevermarried or divorced female

Non-parametric framework

Homogeneous treatment effects: Partially linear model

Partially Linear Regression model

PLR model

$$y = d\alpha + g(X) + \varepsilon$$
$$d = h(X) + \eta$$

- α is the target parameter, g and h are nuisance functions⁷⁷

- Naive ML approach:

- ① ML of $y - d\hat{\alpha}$ on $X \rightarrow$ obtain $\hat{g}(X)$
- ② OLS of $y - \hat{g}(X)$ on $d \rightarrow$ obtain $\hat{\alpha}$

Initialize with $\hat{\alpha} = 0$ and iterate until convergence

- However, $\hat{\alpha}$ is biased, because \hat{g} is not a good estimate of g ⁷⁸

⁷⁷ h maybe redundant, it is the propensity score in TE litterature

⁷⁸ Since $E(y|X) \neq g(X)$, a ML fit of y on X is not a good estimate of g

Homogeneous treatment effects: Partially linear model

Partially Linear Regression model

PLR model

$$y = d\alpha + g(X) + \varepsilon$$

$$d = h(X) + \eta$$

- α is the target parameter, g and h are nuisance functions
- Double Residuals (DR):
 - ① ML of y on X : compute residuals $\hat{\eta}_y = y - \hat{g}(X)$
 - ② ML of d on X : compute residuals $\hat{\eta}_d = d - \hat{h}(X)$
 - ③ OLS of $\hat{\eta}_y$ on $\hat{\eta}_d \rightarrow \hat{\alpha}$

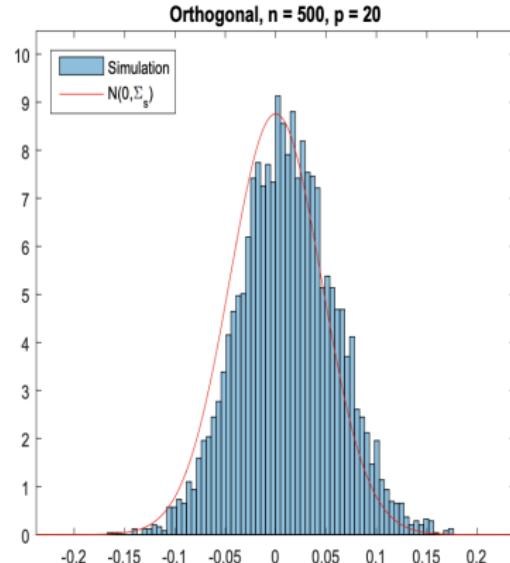
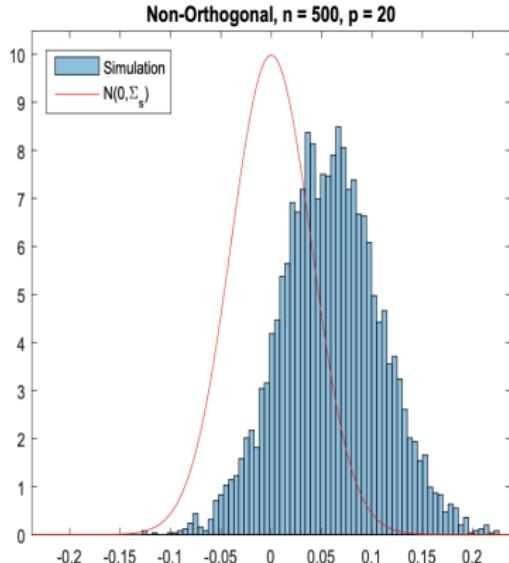
An application of FWL, or partialling out, with ML methods

- Robinson (1988) shows that with DR $\hat{\alpha}$ is \sqrt{n} -consistent, even if $\hat{g}(X)$ and $\hat{h}(X)$ are consistent at slower rates⁷⁹
- The role of DR is to immunize $\hat{\alpha}$ against ML estimates: $\hat{\alpha}$ is based on residuals $\hat{\eta}_y$ and $\hat{\eta}_d$, which are \perp to $\hat{g}(X)$ and $\hat{h}(X)$

⁷⁹Robinson considers kernel regression. Chernozukhov et al. (2018) [pdf](#)
establish that any ML method can be used, so long as it is $n^{1/4}$ -consistent

The role of double residuals (orthogonalization)

Distribution of $\hat{\alpha} - \alpha_0$



Source: Chernozhukov et al. (2018)

Non-orthogonal \equiv naive ML

Orthogonal \equiv Double Residuals

Homogeneous treatment effects: Partially linear model

Partially Linear Regression model

PLR model

$$y = d\alpha + g(X) + \varepsilon$$

$$d = h(X) + \eta$$

- α is the parameter of interest, g and h are nuisance functions
- **Cross-fitting:** split the sample into an auxiliary and a main
 - ① ML estimation of $g(X)$, $h(X)$ on auxiliary sample
 - ② Double Residuals estimation of α by OLS on main sample

Flip the roles of both samples and average the results $\frac{\hat{\alpha}_1 + \hat{\alpha}_2}{2}$

- Estimate nuisance fcts and target parameter on \neq samples
- Chernozukhov et al. (2018) show that cross-fitting is crucial to avoid overfitting

→ PLR: Double ML = Double Residuals + Cross-fitting

Heterogeneous treatment effects: Fully nonparametric

Interactive Regression Model

IRM model

$$y = m(d, X) + \varepsilon$$

$$d = h(X) + \eta$$

- d not additively separable \rightarrow very general heterogeneity in TE
- Parameter of interest: $\text{ATE} = \mathbb{E}[y_1 - y_0]$ ⁸⁰
- The estimator needs to check a **Neyman-orthogonal condition** with respect to the nuisance functions (\equiv DR in the PRL)
- So the estimator and inference are robust to small mistakes in the nuisance functions
- The **AIPW** estimator turns out to check this \perp condition:

$$\text{ATE} = \mathbb{E} \left[m(1, X) - m(0, X) + \frac{D(Y - m(1, X))}{h(X)} - \frac{(1 - D)(Y - m(0, X))}{(1 - h(X))} \right]$$

- This estimator is **doubly-robust**: to small mistakes in \hat{m} and \hat{h}

⁸⁰The observed outcome is with or without treatment: $y = y_1 d + y_0(1 - d)$



Heterogeneous treatment effects: Fully nonparametric

Interactive Regression Model

IRM model

$$y = m(d, X) + \varepsilon$$

$$d = h(X) + \eta$$

- d not additively separable → very general heterogeneity in TE
- Double Machine Learning:⁸¹
 - ① Neyman orthogonal condition → AIPW estimator
 - ② Cross-fitting → ATE and m, h estimated from \neq samples
- ATE estimation and inference with good properties
- However, no detection and analysis of heterogeneity
→ IRM: Double ML = AIPW + Cross-fitting

⁸¹Chernozhukov et al. (2018) [pdf](#) and Chernozhukov et al. (2017) [pdf](#)

Application 3: Insurance bonus on employment duration

- RCT to investigate the incentive effect of unemployment insurance (UI) bonus on unemployment duration:⁸²

Individuals in the treatment groups were offered a cash bonus if they found a job within some pre-specified period of time (qualification period), provided that the job was retained for a specified duration
- y is the log of duration of unemployment for the UI claimants
- ATE estimation and inference in a PLR and IRM models
- Pennsylvania Reemployment Bonus data set
- Implementation is done with the R package DoubleML⁸³

⁸²Individuals in the treatment groups were offered a cash bonus if they found a job within some pre-specified period of time (qualification period), provided that the job was retained for a specified duration

⁸³See [vignette](#) and Bach, Chernozhukov, Kurz, Spindler (2021) [pdf](#)

Application 3: ATE in a PLR model

```
1 library(DoubleML)
2 library(mlr3)
3 # Initialization of the Data-Backend
4 data=fetch_bonus(return_type="data.table")
5 y="inuidur1"
6 d="tg"
7 x=c("female","black","othrace","dep1","dep2","q2","q3","q4",
     "q5","q6","agelt35","agegt54","durable","lusd","husd")
8 dml_data=DoubleMLData$new(data,y_col=y,d_cols=d,x_cols=x)
9 # Initialization of the PLR Model
10 set.seed(31415) #required to replicate sample split
11 learner_g=lrn("regr.ranger",num.trees=500,min.node.size=2,
                 max.depth=5) #Random Forest from the ranger package
12 learner_m=lrn("regr.ranger",num.trees=500,min.node.size=2,
                 max.depth=5)
13 dml_plr=DoubleMLPLR$new(dml_data,
14                           ml_m = learner_m,
15                           ml_g = learner_g,
16                           score = "partialling out",
17                           n_folds = 5, n_rep = 1)
18 # Perform the ATE estimation and print the results
19 dml_plr$fit()
20 dml_plr$summary()
```

Application 3: ATE in a PLR model

```
20 > dml_plr$summary()
21 Estimates and significance testing of the effect of target
22 variables
23   Estimate Std. Error t value Pr(>|t|)
23 tg -0.07396 0.03540 -2.089 0.0367 *
24 ---
25 Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 .0.1 1
```

- Hence, we can reject $H_0 : \alpha = 0$ at the 5% significance level
- It is consistent with the findings of previous studies that have analysed the Pennsylvania Bonus Experiment
- The ATE on unemployment duration is negative and significant

Application 3: ATE in an IRM model

```
26 ## Initialization of the IRM Model
27 # Classifier for propensity score
28 learner_classif_m = lrn("classif.ranger", num.trees = 500,
29   min.node.size = 2, max.depth = 5)
30 dml_irm=DoubleMLIRM$new(dml_data,
31   ml_m = learner_classif_m,
32   ml_g = learner_g,
33   score = "ATE", #or "ATTE"
34   n_folds = 10, n_rep = 1)
35 # Perform the estimation and print the results
36 dml_irm$fit()
37 dml_irm$summary()
```

```
37 Estimates and significance testing of the effect of target
38 variables
39 Estimate Std. Error t value Pr(>|t|)
40 tg -0.07345 0.03549 -2.069 0.0385 *
41 Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 .0.1 1
```

The estimated coefficient is very similar to the estimate of the ATE in a PLR model and the conclusions remain unchanged.

Estimation of heterogeneity: Causal Forest

Causal Random Forest:⁸⁴

- Random Forest is modified to estimate the CATE directly
- Grow a tree and evaluate its performance based on TE heterogeneity rather than predictive accuracy
- The idea is to find leaves where the treatment effect is constant but different from other leaves
- Split criterion: maximize heterogeneity in TE between leaves
- Honest tree: build tree and estimate CATE from \neq samples

→ valid estimation and confidence intervals for CATE⁸⁵

⁸⁴Wager and Athey (2018) [pdf](#), Athey, Tibshirani and Wager (2019) [pdf](#)

⁸⁵RF predictions are asymptotically unbiased and Gaussian, but cv rates below \sqrt{n} and they do not account for the uncertainty due to sample splitting



Detection and analysis of heterogeneity: Generic ML

Generic Machine Learning:⁸⁶

- Do not attempt to get valid estimation and inference on the CATE itself, but on features of the CATE
- Obtain ML proxy predictor of CATE (auxiliary set) and target features of CATE based on this proxy predictor (main set)

Main interests:

- Test if there is evidence of heterogeneity (BLP)
- ATE for the 20% most (least) affected individuals? (GATES)
- Which covariates are associated to TE heterogeneity? (CLAN)

→ valid estimation and inference on *features* of CATE

⁸⁶Chernozhukov, Demirer, Duflo and Fernàndez-Val (2020) 

Generic ML: Proxies of CATE

The main idea is to compute imperfect predictions of CATE and to use them as proxies to make inferences on features of CATE:

- Split the sample into a main set and auxiliary set (50/50 split)
- Fit $y \approx m(1, X)$ with treated group from the auxiliary sample
- Fit $y \approx m(0, X)$ with control group from the auxiliary sample
- Compute $\widehat{S}(X_i) = \hat{m}(1, X_i) - \hat{m}(0, X_i)$ from the main sample
- $\widehat{S}(X)$ is used to learn about treatment effect heterogeneity

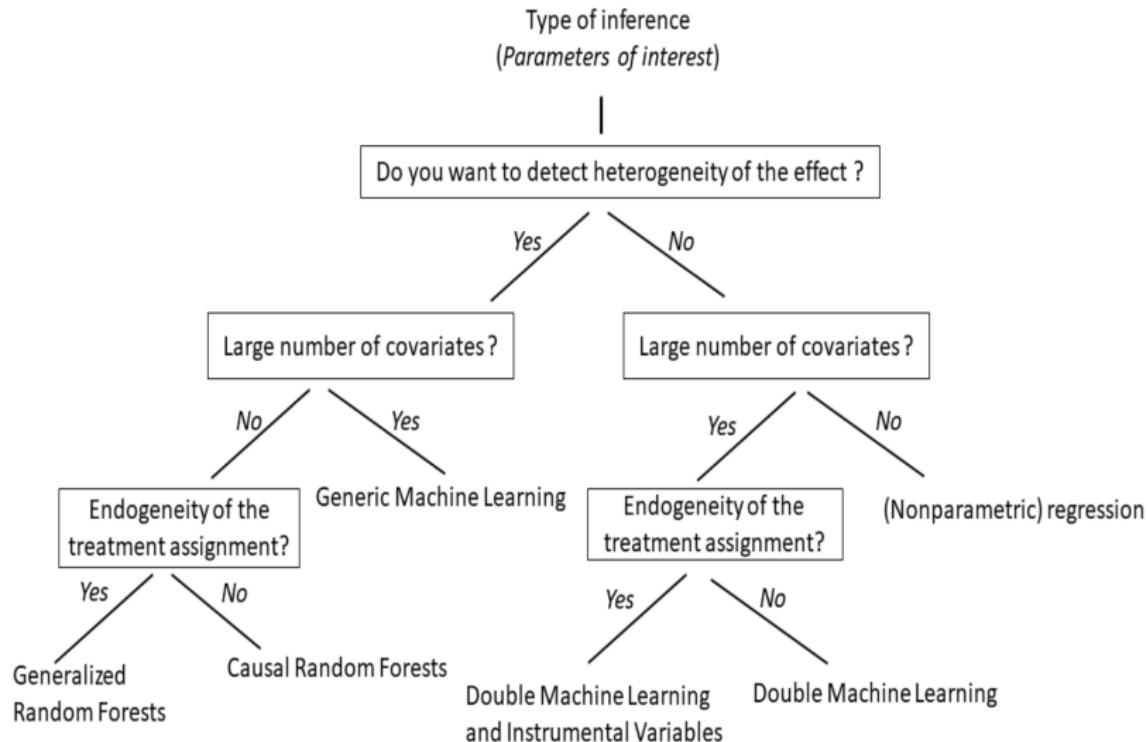
To control the uncertainty due to data splitting, this process is done many times → cross-fitting⁸⁷

The $\widehat{S}(X_i)$ are imperfect predictions of CATE; → proxies⁸⁸

⁸⁷We randomly split the sample M times. The parameter estimates, confidence bounds, and p -values reported are the medians across M splits.

⁸⁸ $CATE_i = \mathbb{E}[y_1 - y_0 | X_i] = m(1, X_i) - m(0, X_i)$

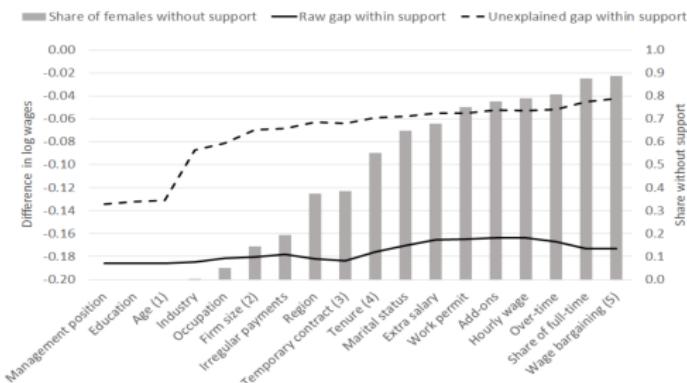
Causal Machine Learning: A brief roadmap



Source: Gaillac and L'Hour (2021)

Underlying assumptions

- Standard hypotheses: SUTVA, CIA and CSC
- Common support condition (CSC): $0 < P(d_i = 1 | X_i = x) < 1$
 - ML estimation often provides better predictions
 - Adding covariates makes matching more difficult



Strittmatter and Wunsch (2021) The gender pay gap revisited with big data: Do methodological choices matter?

- Trimming in experiments vs. decomposition methods
 - Beware of CSC when moving away from RCT framework

Conclusion

The impact of ML for public policy evaluation:

- Dealing with many covariates ($p \gg n$)
- Relying less on a priori specification
- Take care of heterogeneity
- However, do not forget underlying assumptions! (CSC)

Technical literature, where implementation becomes easier

- Double Lasso: R package `hdm`
- Double Machine Learning: R package `DoubleML`
- Generic Machine Learning: R package `GenericML`
- Generalized Random Forest: R package `grf`

An effervescent empirical and theoretical literature

Selected references in Causal ML

- Athey (2017) Beyond prediction: Using big data for policy problems, Science
- Athey (2018) The impact of machine learning on economics
- Athey, Tibshirani and Wager (2019) Generalized random forest, Ann. Statis.
- Bach, Chernozhukov and Spindler (2021) Closing the U.S. gender wage gap requires understanding its heterogeneity, arXiv:1812.04345
- Belloni, Chernozhukov and Hansen (2014) Inference on treatment effects after selection amongst high-dimensional controls, REStud
- Chernozhukov, Chetverikov, Demirer, Duflo, Hansen, Newey and Robins (2018) Double/debiased ML for treatment and structural parameters. Econometrics J.
- Chernozhukov, Demirer, Duflo and Fernàndez-Val (2020) Generic ML inference on heterogenous treatment effects in randomized experiments, arXiv:1712.04802
- Gaillac and L'Hour (2020) Machine Learning for Econometrics, Lecture notes
- Kleinberg, Ludwig, Mullainathan and Obermeyer (2015) Prediction Policy Problems, AER P&P
- L'Hour (2020), L'économétrie en grande dimension. INSEE M2020-01
- Strittmatter (2020) What is the value added by using causal machine learning methods in a welfare experiment evaluation.
- Strittmatter and Wunsch (2021) The gender pay gap revisited with big data: Do methodological choices matter? arXiv:2102.09207
- Wager and Athey (2018) Estimation and inference of heterogeneous treatment effects using random forests. JASA

References

- Berk (2016) *Statistical Learning from a Regression Perspective*. Springer Texts in Statistics, ch.3-7
- Charpentier (2018), Classification from scratch [▶ website](#)
- Charpentier, Flachaire and Ly (2018), Econometrics and Machine Learning, *Economics and Statistics*, 505 [▶ english](#) [▶ french](#)
- Efron and Hastie (2016) *Computer Age Statistical Inference*, Cambridge University Press, ch.17-19 [▶ pdf](#)
- Hastie, Tibshirani and Friedman (2009) *The Elements of Statistical Learning*. Springer, ch.7, 9-12, 15-16 [▶ website](#) [▶ pdf](#)
- Hastie, Tibshirani and Wainwright (2015) *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, [▶ website](#) [▶ pdf](#)
- James, Witten, Hastie and Tibshirani (2021) *An Introduction to Statistical Learning*. Springer, ch.2,5,6,8,9 [▶ website](#) [▶ pdf](#)
- Watt, Borhani and Katsaggelos (2016) *Machine Learning Refined*. Cambridge University Press, ch.1-6