Multivariate discriminative probability including machine learning

Adam Boult (www.bou.lt)

November 7, 2026

Contents

Preface **2**

I Estimating discriminative probability distributions **3**

1 Bayesian parameter estimation of discriminative models **4**

2 Point variable estimates for discriminative models **7**

3 Using F-tests to compare regression models **12**

4 Test sets and validation sets **13**

5 Choosing parametric discriminative probability distributions **15**

II Supervised linear regression **18**

6 Ordinary Least Squares for prediction **19**

7 Regularising linear regression for prediction **25**

8 Choosing linear models for prediction **29**

9 Generalised linear models, the delta rule and binary classification **30**

10 Generalised linear models and multiclass classification **36**

III Supervised machine learning **41**

11 Classification trees **42**

12 Regression trees **45**

13 Bayesian trees **47**

14 Support Vector Machines (SVMs) **49**

15 Variational Bayes **51**

16 The Naive Bayes classifier **52**

17 The K-Nearest Neighbours (KNN) classifier **54**

18 Discriminant analysis **55**

19 Non-parametric regression **56**

20 Ensemble methods **59**

21 Ensemble methods for trees **63**

22 Regularising black box models **64**

23 Confidence intervals of black box models **65**

24 Interpreting black box models **66**

25 Semi-supervised learning **68**

26 Imputing missing data for prediction **69**

27 Recommenders **70**

IV SORT **72**

28 SORT 2025 **73**

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Estimating discriminative probability distributions

Chapter 1

Bayesian parameter estimation of discriminative models

1.1 Introduction

1.1.1 Generative and discriminative models

Recap

For parametric models without dependent variables we have a form:

$P(y|\theta)$

And we have various ways of estimating θ.

We can write this as a likelihood function:

$L(\theta; y) = P(y|\theta)$

Discriminative models

In discriminative models we learn:

$P(y|X, \theta)$

Which we can write as a likelihood function:

$L(\theta; y, X) = P(y|X, \theta)$

Generative models

In generative models we learn:

$P(y, X|\theta)$

Which we can write as a likelihood function:

$L(\theta; y, X) = P(y, X|\theta)$

We can use the generative model to calculate dependent probabilities.

$$P(y|X, \theta) = \frac{P(y, X|\theta)P(\theta)}{P(X, \theta)}$$

$$P(y|X, \theta) = \frac{P(y, X|\theta)}{P(X|\theta)}$$

1.2 Bayesian parameter estimation

1.2.1 Bayesian parameter estimation for dependent models

Recap

For non-dependent models we had:

$$P(\theta|y) = \frac{P(y, \theta)}{P(y)}$$

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

The bottom bit is a normalisation factor, and so we can use:

$P(\theta|y) \propto P(y|\theta)P(\theta)$

We have here:

- Our prior - $P(\theta)$
- Our posterior - $P(\theta|y)$
- Our likelihood function - $P(y|\theta)$

Bayesian regression for generative models

We know:

$$P(\theta|y, X) = \frac{P(y, \theta, X)}{P(y, X)}$$

$$P(\theta|y, X) = \frac{P(y, X|\theta)P(\theta)}{P(y, X)}$$

The bottom bit is a normalisation factor, and so we can use:

$P(\theta|y, X) \propto P(y, X|\theta)P(\theta)$

We have here:

- Our prior - $P(\theta)$
- Our posterior - $P(\theta|y, X)$
- Our likelihood function - $P(y, X|\theta)$

Bayesian regression for discriminative models

We know:

$$P(\theta|y, X) = \frac{P(y, \theta, X)}{P(y, X)}$$

$$P(\theta|y, X) = \frac{P(y|\theta, X)P(\theta, X)}{P(y, X)}$$

$$P(\theta|y, X) = \frac{P(y|\theta, X)P(\theta)P(X|\theta)}{P(y, X)}$$

We assume $P(X|\theta) = X$ and so:

$$P(\theta|y, X) = \frac{P(y|\theta, X)P(\theta)P(X)}{P(y, X)}$$

The bottom bit is a normalisation factor, and so we can use:

$P(\theta|y, X) \propto P(y|X, \theta)P(\theta)$

We have here:

- Our prior - $P(\theta)$
- Our posterior - $P(\theta|y, X)$
- Our likelihood function - $P(y|X, \theta)$

1.2.2 Prior and posterior predictive distributions for dependent variables

Prior predictive distribution

Our prior predictive distribution for $P(y|X)$ depends on our prior for θ.

$P(y|X) = \int_\Theta P(\mathbf{y}|X, \theta)P(\theta)d\theta$

Posterior predictive distribution

Once we have calculated $P(\theta|\mathbf{y}, \mathbf{X})$, we can calculate a posterior probability distribution for $P(y|X)$.

$P(y|\mathbf{x}, \mathbf{y}, \mathbf{X}) = \int_\Theta P(y|\mathbf{x}, \theta)P(\theta|\mathbf{y}, \mathbf{X})d\theta$

Chapter 2

Point variable estimates for discriminative models

2.1 Predictions and residuals

2.1.1 Predictions

Our data $(\mathbf{y}, \mathbf{X})$ is divided into $(y_i, \mathbf{x}_i)$.

We create a function $\hat{y}_i = f(\mathbf{x}_i)$.

The best predictor of y given x is:

$g(X) = E[Y|X]$

The goal of regression is to find an approximation of this function.

2.1.2 Residuals

$\epsilon_i = y_i - \hat{y}_i$

2.1.3 Residual sum of squares (RSS)

$RSS = \sum_i \epsilon_i^2$

$RSS = \sum_i (y_i - \hat{y}_i)^2$

2.1.4 Explained sum of squares (ESS)

$ESS = \sum_i (\bar{y} - \hat{y}_i)^2$

2.1.5 Total sum of squares (TSS)

$TSS = \sum_i (y_i - \bar{y})^2$

2.1.6 Relationship between prediction and probability distribution

$P(y|X, \theta)$

$\hat{y} = f(\mathbf{x})$

Through integration?

$E[y] = \int P(y|X, \theta) dy$

2.1.7 Coefficient of determination (R^2)

$R^2 = 1 - \frac{RSS}{TSS}$

2.2 Classification

2.2.1 Binary classification

Classification models are a type of regression model, where y is discrete rather than continuous.

So we want to find a mapping from a vector X to probabilities across discrete y values.

A classifier takes X and returns a vector.

For a classifier we have K classes.

2.2.2 Classification

Confusion matrix. true positve, false positive, false negative, true negative

Can use this to get

Accuracy: percentage correct

Precision: percentage of positive predictions which are correct

Recall (sensitivity): percentage of poitive cases that were predicted as positive

Specificity: percentage of negative cases preicated as negative

2.2.3 Multiclass classification

Multiclass classification

What if can be email for work, friends, family, hobby?

2.2.4 Confusion matrix

Include error types here

2.2.5 Hard and soft classifiers

A hard classifier can return a sparce vector with 1 in the relevant classification.

A soft classifier returns probabilities for each entry in the vector.

The vector represents $P(Y = k|X = x)$

2.2.6 Transforming soft classifiers into hard classifiers

We can use a cutoff.

If there are more than two classes we can choose the one with the highest score.

2.3 Loss functions for point predictions

2.3.1 Minimum Mean Square Error (MMSE)

Mean estimate.

Can do for a parameter, or for a predicted estimate for y.

Linear models

MLE is same as y^2 loss

MAP is same as y^2 loss with regularisation

2.3.2 Loss functions for soft classifiers

Hinge loss

Brier score

2.3.3 Loss functions for hard classifiers

Don't want answers outside 0 and 1.

F score

F1 score

F_1 score: $\frac{2PR}{(P+R)}$

may not just care about accuracy, eg breast cancer screening

high accurancy can result from v basic model (ie all died on titanic)

Receiver Operating Characteristic (ROC) Area Under Curve (AUC)

2.4 Other

2.4.1 Estimating other priors

Estimating $P(k|T)$ - Which variables we split by, given the tree size

Estimating $P(r|T,k)$ - The cutoff, given the tree size and the variables we are splitting by

Estimating $P(\theta|T,k,r)$

2.4.2 Maximum Likelihood Estimation (MLE) for generative and discriminative models

2.4.3 Maximum A-Priori estimation (MAP) for generative models

Bayesian regression for generative models

We know:

$$P(\theta|y,X) = \frac{P(y,\theta,X)}{P(y,X)}$$

$$P(\theta|y,X) = \frac{P(y,X|\theta)P(\theta)}{P(y,X)}$$

The bottom bit is a normalisation factor, and so we can use:

$$P(\theta|y,X) \propto P(y,X|\theta)P(\theta)$$

We have here:

- Our prior - $P(\theta)$
- Our posterior - $P(\theta|y,X)$
- Our likelihood function - $P(y,X|\theta)$

2.4.4 Bayesian classifier

Classification risk

We can measure the risk of a classifier. This is the chance of misclassification.

$$R(C) = P(C(X) \neq Y)$$

The Bayesian classifier

This is the classifer $C(X)$ which minimises the chance of misclassification.

It takes the output of the soft classifier and chooses the one with the highest chance.

Chapter 3

Using F-tests to compare regression models

3.1 Hypothesis testing

3.1.1 Power of tests

3.1.2 Type I and type II errors

3.1.3 Sensitivity tests

3.2 F-test

3.2.1 F test for equal population means

3.2.2 F test for additional variables

3.3 Other

3.3.1 Cohen's d

Chapter 4

Test sets and validation sets

4.1 Test sets

4.1.1 Overfitting

Overfitting is a risk. Instead we split to test, train. risk of using too many features. more features always improve training score, not necessarily test score.

As model gets more complex, both test and train do better. however at some point, test stops doing better, overfitting

Structural risk minimisation can address this trade off. use test and training sets. train model on train, rate it on test

Structural minimisation curve has accuracy of boths sets over complexity

To avoid overfitting:

+ reduce number of features + do a model selection + use regularisation + do cross validation

Can choose other model parameters

How to evalute model?

4.1.2 K-fold cross-validation

Can do k-fold cross validation. given algo A and dataset D, divide D into k equal sized subsets

For each subset, train the model on all other subsets and test on the other subset. average error between folds

4.2 Splitting

4.2.1 K-folds

The problem of different sample sizes (sample size for validations sets is lower, different hyper parameters could be more appropriate)

4.2.2 Leave-One-Out

4.3 Hyperparameters

4.3.1 Learning rate, batching and momentum

4.4 Search methods

4.4.1 Grid-search

4.5 Validation sets

4.5.1 Validation sets

which features? remove, add?

change lambda,regularisation

change polynomial features

Chapter 5

Choosing parametric discriminative probability distributions

5.1 Sample sizes

5.1.1 Sample sizes

If you're modelling house prices using just size, getting a large sample size won't help too much

Can improve low bias models

Is data size an issue? can artificially restrict training data size and then evaluate error

Training:

+ zero error for low m + increases error as m increases, as degrees of freedom/m falls

cv:

+ error decreases as data set increases, more accurate theta

The two curves converge towards each other for v large m

When are large datasets useful?

When all features available:

Predicting house price using just size, won't benefit from more data...

If choosing correct word in sentence (to, too, two), more helpful

If human expert can do it, then more data probably helpful

Expert realtor probably couldn't do much with just size, but speaker could answer other q

Could expert do it?

Low bias algorithms do well with more data

More data good if large number of parameters, or lot of hidden units.

5.2 Overfitting

5.2.1 Overfitting

Role of lambda: high makes impact of more variables lower =¿ high bias

Low makes impacts of more variables strong =¿ high variance

Can trade off using cut off. only make positive if above 0.7

How to use? difficult, as lambda within cost!

Can do similarly to d:

Run for a range of lambda (eg 0, 0.01, 0.02, 0.04, 0.08: 10), then pick from cross validation set

Low lambda always has low cost for training set, but not for cv set..

Regularisation: add to error term the size of the term. penalised large parameters

May not fit outside sample

High bias: eg house prices and size. linear would have high bias for out of scope sample (underfitting)

High variance: making polynomial passing through all data (overfitting)

Can reduce overfitting by reducing features either manaually or using models

OR regularisation: keep all features, but reduce magnitude of theta

5.2.2 Regularisation

Make cost function include size of θ^2 values

$\min \frac{1}{2m}[\sum(h(x)-y)^2 + 1000\theta 3^2 + 1000\theta 4^2]$

or more broadly:

$\min \frac{1}{2m}[\sum ..... + \lambda \sum \theta j^2]$

Tend to not include theta 0 as convention, no regularisation

Update for linear regression is

$\theta j = \theta j - \alpha(\frac{1}{m}) * sum(h(x) - y)xj + (\lambda/m\theta j)$

$\theta j = \theta j(1 - \alpha\lambda/m) - alpha(1/m) * \sum(h(x) - y)xj$

This is the same as before, but theta j updates from a smaller θ j each time.

Normal equation needs a change

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

Now is

$(X'X + \lambda I)^{-}1Xy'$

although for theta 0, lambda zero, so indentiy matrix, but first element 0

REGULARISATION FOR REGULARISATION

add to end of $J(\theta)$:

$\frac{\lambda}{2m} \sum \theta j^2$

update for θj $j > 0$: is a as linear regression, but $h(x)$ is a different function

Part II

Supervised linear regression

Chapter 6

Ordinary Least Squares for prediction

6.1 Constructing a linear model

6.1.1 Defining linear models

Defining

One option for $f(X)$ is a linear model.

$f(X_i) = \hat{Y}_i = \beta_0 + \sum_{j=1}^{p} \beta_i X_{ij}$

The values for β are the regression coefficients.

So we have:

$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_i X_{ij} + e(X_i) + e_i$

We define the error of the estimate as:

$\epsilon_i = Y_i - \hat{Y}_i$

$\epsilon_i = e(X_i) + e_i$

So:

$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_i X_{ij} + \epsilon_i$

The linear model could be wrong for two reasons. No linear model could be appropriate, or the wrong coefficients could be provided for a linear model.

Linear regression if f is a linear function on w. NB: not linear in x necessarily. could have x^2 etc, but still linear in w.

6.1.2 Intercept

6.1.3 Modelling non-linear functions as linear

Polynomials

The function $y = x^2$ is not linear, however we can model is as linear, by including x^2 as a variable.

We can expand this, and using linear models to estimate parameters for functions such as:

$y = ax^3 + bx^2 + xz$

Logarithms and exponentials

We can also transform data using logarithms and exponents.

For example we can model

$\ln y = \theta \ln x$

6.1.4 Geometric interpretation of OLS

Best Approximation Theorem

6.2 Calculating Ordinary Least Squares (OLS) estimators

6.2.1 Normal equation

Least squares

The square error is $\sum_i (\hat{y_i} - y_i)^2$.

The differential of this with respect to $\hat{\theta_j}$ is:

$2 \sum_i \frac{\delta \hat{y_i}}{\delta \hat{\theta_j}} (\hat{y_i} - y_i)$

The stationary point is where this is zero:

$\sum_i \frac{\delta \hat{y_i}}{\delta \hat{\theta_j}} (\hat{y_i} - y_i) = 0$

Linear least squares

Here, $\hat{y_i} = \sum_j x_{ij} \hat{\theta_j}$

Therefore: $\frac{\delta \hat{y_i}}{\delta \hat{\theta_j}} = x_{ij}$

And so the stationary point is where

$\sum_i x_{ij}(\sum_j x_{ij}\hat{\theta_j} - y_i) = 0$

$\sum_i x_{ij}(\sum_j x_{ij}\hat{\theta_j}) = \sum_i x_{ij}y_i$

Normal equation

We can write this in matrix form.

$X^T X\hat{\theta} = X^T y$

We can solve this as:

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

Perfectly correlated variables

If variables are perfectly correlated then we cannot solve the normal equation.

Intuitively, this is because for perfectly correlated variables there is no single best parameter, as changes to one parameter can be counteracted by changes to another.

6.2.2 Mean and variance of predictions

Bias

$\hat{y} = \theta x$

$E[\hat{y} - y] = E[\theta x - y]$

$y = \hat{y} + \epsilon$

$E[y - \hat{y}|X]$

$E[\epsilon|X]$

Unbiased so long as independent of error term.

Variance

$Var[\hat{y} - y] = Var[\theta x - y]$

$Var[y - \hat{y}|X]$

$Var[\epsilon|X]$

6.2.3 The Moore-Penrose pseudoinverse

For a matrix X, the pseudoinverse is $(X^* X)^{-1}X^*$.

For real matrices, this is: $(X^T X)^{-1}X^T$

The pseudoinverse can be written as X^+

Therefore θ is the pseudoinverse of the inputs, multiplied by the outputs. Or:

$\theta = X^+ y$

The pseudoinverse satisfies:

$XX^+X = X$

$X^+XX^+ = X^+$

6.2.4 Leverage

Introduction

Leverage measures how much the predicted value of y_i, $\hat{y}_i$, changes as y_i changes.

We have:

$\mathbf{y} = \mathbf{X}\theta + \mathbf{u}$

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

$\hat{\theta} = P_X y$

The leverage score is defined as:

$h_i = P_{ii}$

6.3 Making forecasts with OLS

6.3.1 The projection and annihilation matrices

The projection matrix

We have X.

The projection matrix is $X(X^T X)^{-1} X^T$

The projection matrix maps from actual y to predicted y

$\hat{y} = Py$

Each entry refers to the covariance between actual and fitted

$$p_{ij} = \frac{Cov(\hat{y}_i, y_j}{Var(y_j)}$$

The annihilation matrix

We can get residuals too:

$u = y - \hat{y} = y - py = (1 - P)y$

$1 - P$ is called the annihilator matrix

We can now use the propagation of uncertainty

$\Sigma^f = A\Sigma^x A^T$

To get:

$\Sigma^u = (I - P)\Sigma^y(I - P)$

Annihilator matrix is:

$M_X = I - X(X^T X)^{-1}X^T$

Called this because:

$M_X X = X - X(X^T X)^{-1}X^T X$

$M_X X = 0$

Is called residual maker

6.4 Frisch-Waugh-Lovell theorem

6.4.1 Introduction

If we have a partitioned linear regression model:

$\mathbf{y} = \mathbf{X}\theta + \mathbf{Z}\beta + \mu$

Use the annihilator matrix:

$M_X\mathbf{y} = M_X\mathbf{X}\theta + M_X\mathbf{Z}\beta + M_X\mu$

$M_X\mathbf{y} = M_X\mathbf{Z}\beta + M_X\mu$

We can then estimate β.

Frisch-Waugh-Lovell theorem says that this is the same estimate as the original regression.

6.5 Trimming

6.5.1 Introduction

OLS:

$\hat{\theta} = \frac{\sum_i (X_i - \mu_X)(y_i - \mu_y)}{\sum_i (x_i - \mu_X)^2}$

Trimming

$\hat{\theta} = \frac{n^{-1}\sum_i (X_i - \mu_X)(y_i - \mu_y)\mathbf{1}_i}{n^{-1}\sum_i (x_i - \mu_X)^2\mathbf{1}_i}$

Where:

$\mathbf{1}_i = \mathbf{1}(\hat{f}(z_i) \geq b)$

Where $b = b(n)$ is a trimming parameter, where:

$b \to 0$ as $n \to \infty$

6.6 Best linear predictor

6.6.1 Introduction

The best linear predictor is the one which minimises:

$E[Y - X\theta]$

Under what circumstances is this the same as OLS? When n¿¿p. When n is not, then other linear estimators (like LASSO) can be better.

6.7 Other

6.7.1 Cook's distance

Cook's distance measures the effect of deleting outliers. work out predictions if outlier was removed, sum all differneces in y hat

Outliers have a high Cook's distance.

6.7.2 Bayesian linear regression

In linear regression we have

$P(y|X, \theta, \sigma_\epsilon^2)$

For Bayesian linear regression we want:

$P(\theta, \sigma_\epsilon^2|y, X)$

We can use Bayes rule:

$P(\theta, \sigma_\epsilon^2|y, X) \propto P(y|X, \theta, \sigma_\epsilon^2)P(\theta|\sigma_\epsilon^2)P(\sigma_\epsilon^2)$

Chapter 7

Regularising linear regression for prediction

7.1 OLS predictions with many parameters

7.1.1 Too many variables

If there are more independent variables than samples then OLS will not work. There will be an infinite number of perfect fits.

For example if we regression genetic information on height with 1000 people, there will be too little data to fit using OLS.

This is due to colinearity.

We could also have too many variables through the use of derived variables. For example if we choose to use x, x^2, x^3 etc.

Optimal sparce regression

Optimal is $\lambda = \sigma 2\sqrt{2\log(pn)n}$

Relies on knowing σ, which we may not.

Instead we can use root LASSO.

Minimise the squareroot of the sum of squares loss (over n) , and use $\lambda = \sqrt{2\log(pn)/n}$

Doesn't have σ

Lasso biased, estimators 0 for many.

Post-LASSO

We can use LASSO for model selection, then use OLS on only those estimators.

7.2 Least Absolute Shrinkage and Selection Operator (LASSO)

7.2.1 Least Absolute Shrinkage and Selection Operator (LASSO)

Introduction

With LASSO we add a constraint to $\hat{\theta}$.

$\sum_i \hat{\theta}_i \leq t$

Regularisation of LLS. Sum of thetas are constrained to be below hyperparameter t

L1 regularisation

This is also known as sparce regression, because many weights are set to 0.

This now looks like:

$w_{lasso} = \arg\min ||y - Xw||_2^2 + \lambda||w||_1$

Hyperparameter

t is a hyperparameter.

7.2.2 Optimal hyperparameter for LASSO

7.2.3 Feature scaling

7.3 Ridge regression

7.3.1 Ridge regression

Regularisation of LLS. The cost function now includes a norm on $M\theta$.

L_2 regularisation

This allows us to solve problems where there are too many features. L_1 also allows us to to do this.

Overspecified

If $n > d$ we can minimise weights subject to Xw=y. This is the same as the least norm.

Maximum A-Priori (MAP) estimator for linear regression

Maximum a priori estimation. equiv to ridge regression with a priori estimate of 0

$W_{RR} = (\lambda I + X^T X)^{-1} X^T y$

$E_[wRR] = (\lambda I + X^T X)^{-1} X^T X w$

$Var[W_{RR}] = a$

7.3.2 Elasticnet

Regularisation of LLS. Combines lasso and ridge regression.

L_1 and L_2 regularisation

7.3.3 L_p regularisation

Introduction

We can generalise this to:

$w_{l_p} = \arg\min ||y - Xw||_2^2 + \lambda ||w||_p^p$

For ridge regression there is always a solution.

For least squares there is a solution if $X^T X$ is invertible

For Lasso we must use numerical optimisation.

lasso and L_1 induces sparcity

Goal is $min||y - f(x)|| + \lambda g(w)$

Ridge regression: $g(w) = ||w||$

If $\lambda = 0$, OLS, if infinite, w goes to 0.

Normal equation changes to: $\lambda I + X^T X)^{-1} X^T y$

We can preprocess to avoid processing of 1s. shift mean of y to 0. normalise x mean 0 var 1.

7.3.4 Lava

Introduction

Alternative to ElasticNet

Each parameter is split into

$\theta_i = \rho_i + \phi_i$

There is L_2 loss on rho and L_1 loss on ϕ.

This means that large coefficients can be penalised like L_1 and small coefficients like L_2.

7.4 Tests

7.4.1 The Ramsey RESET test

The Ramsey Regression Equation Specification Error Test (RESET)

Once we have done our OLS we have $\hat{y}$.

The Ramsey RESET test is an additional stage, which takes these predictions and estimates:

$y = \theta x + \sum_{i=1}^{3} \alpha_i \hat{y^i}$

We then run an F-test on α, with the null that $\alpha = 0$.

7.4.2 The Link test

Introduction

Alternative to RESET

We have $\hat{y}$.

We regress $y = \alpha + \beta\hat{y} + \gamma\hat{y}^2$.

We test that $\gamma = 0$.

If it is not, then this suggests the model is misspecified.

7.5 Bias trade-off

7.5.1 Introduction

Trade-off between parameter accuracy and prediction accuracy.

Chapter 8

Choosing linear models for prediction

8.1 Other

8.1.1 Selection

How to restrict variables? best subset. iterate through alol and test.

not feasible for large numbers of variables. forward selection, backward selection, L1 alternatives.

removing variable does: increase bias. may reduce variance of prediction

Chapter 9

Generalised linear models, the delta rule and binary classification

9.1 Introduction

9.1.1 Link/activation functions

9.2 Estimating parameters

9.2.1 Delta rule

Introduction

We want to train the parameters $\boldsymbol{\theta}$.

We can do this with gradient descent, by working out how much the loss function falls as we change each parameter.

The delta rule tells us how to do this.

The loss function

If we have n features and m samples The error of the network is:

$E = \sum_j^m \frac{1}{2}(y_j - a_j)^2$

We know that $a_j = f(\boldsymbol{\theta x_j}) = f(\sum_i^n \theta^i x_j^i)$ and so:

$E = \sum_j \frac{1}{2}(y_j - f(\boldsymbol{\theta x_j}))^2$

$E = \sum_j \frac{1}{2}(y_j - f(\sum_i \theta^i x_j^i))^2$

Minimising loss

We can see the change in error as we change the parameter:

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta}{\delta \theta^i} \sum_j \frac{1}{2}(y_j - f(\boldsymbol{\theta x_j}))^2$$

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta}{\delta \theta^i} \sum_j \frac{1}{2}(y_j - f(\sum_i \theta^i x_j^i))^2$$

$$\frac{\delta E}{\delta \theta^i} = \sum_j (y_j - f(\boldsymbol{\theta x_j})) \frac{\delta}{\delta \theta^i}(y_j - f(\boldsymbol{\theta x_j}))$$

$$\frac{\delta E}{\delta \theta^i} = \sum_j (y_j - f(\sum_i \theta^i x_j^i)) \frac{\delta}{\delta \theta^i}(y_j - f(\sum_i \theta^i x_j^i))$$

$$\frac{\delta E}{\delta \theta^i} = \sum_j (y_j - f(\boldsymbol{\theta x_j})) \frac{\delta}{\delta \theta^i}(-f(\boldsymbol{\theta x_j}))$$

$$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - f(\sum_i \theta^i x_j^i)) \frac{\delta}{\delta \theta^i} f(\sum_i \theta^i x_j^i)$$

$$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - f(\sum_i \theta^i x_j^i)) \frac{\delta f(\sum_i \theta^i x_j^i)}{\delta \theta^i}$$

$$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - f(\sum_i \theta^i x_j^i)) \frac{\delta f(\sum_i \theta^i x_j^i)}{\sum_i \theta^i x_j^i} \frac{\sum_i \theta^i x_j^i}{\delta \theta^i}$$

$$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - f(\sum_i \theta^i x_j^i)) f'(\sum_i \theta^i x_j^i) x_j^i$$

By defining $z_j = \sum_i \theta^i x_j^i$ and $a = f$ we have:

$$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - a(z_j)) a'(z_j) x_j^i$$

Minimising loss: Other simpler attempt

We can see the change in error as we change the parameter:

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta}{\delta \theta^i} \sum_j \frac{1}{2}(y_j - f(\boldsymbol{\theta x_j}))^2$$

By defining $z_j = \sum_i \theta^i x_j^i$ and $a = f$ we have:

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta}{\delta \theta^i} \sum_j \frac{1}{2}(y_j - a(z_j))^2$$

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta E}{\delta a_j} \frac{\delta a_j}{\delta z_j} \frac{\delta z_j}{\delta \theta^i}$$

$$\frac{\delta E}{\delta \theta^i} = \frac{\delta E}{\delta a} a'(z_j) x_j$$

$\frac{\delta E}{\delta \theta^i} = \sum_j a'(z_j)x_j$

$\frac{\delta E}{\delta \theta^i} = -\sum_j (y_j - a(z_j))a'(z_j)x_j^i$

Delta

We define delta as:

$\delta_i = -\frac{\delta E}{\delta z_j} = \sum_j (y_j - a_j)a'(z_j)$

So:

$\frac{\delta E}{\delta \theta^i} = \delta_i x_{ij}$

The delta rule

We update the parameters using gradient descent:

$\Delta\theta_i = \alpha\delta^i x_{ij}$

9.2.2 Maximum likelihood

9.2.3 Identity link function

The function

$a(z) = z$

The derivative

$a'(z) = 1$

Notes

This is the same as ordinary linear regression.

9.3 Link/activation functions: Classification

9.3.1 The binomial data generating process

Introduction

For linear regression our data generating process is:

$y = \alpha + \beta x + \epsilon$

For linear classification our data generating process is:

$z = \alpha + \beta x + \epsilon$

And set y to 1 if $z > 0$

Or:

$y = \mathbf{I}[\alpha + \beta x + \epsilon > 0]$

Probability of each class

The probability that an invididual with characteristics x is classified as 1 is:

$P_1 = P(y = 1|x)$

$P_1 = P(\alpha + \beta x + \epsilon > 0)$

$P_1 = \int \mathbf{I}[\alpha + \beta x + \epsilon > 0] f(\epsilon) d\epsilon$

$P_1 = \int \mathbf{I}[\epsilon > -\alpha - \beta x] f(\epsilon) d\epsilon$

$P_1 = \int_{\epsilon=-\alpha-\beta x}^{\infty} f(\epsilon) d\epsilon$

$P_1 = 1 - F(-\alpha - \beta x)$

Example: The logistic function

Depending on the probability distribution of *epsilon* we have different classifiers.

For the logistic case we then have

$$P(y = 1|x) = \frac{e^{\alpha+\beta x}}{1 + e^{\alpha+\beta x}}$$

9.3.2 Perceptron (step function)

The function

If the sum is above 0, $a(z) = 1$. Otherwise, $a(z) = 0$.

The derivative

This has a differential of 0 at all point except 0, where it is undefined.

Notes

This function is not smooth.

These is the activation function used in the perceptron.

Perceptron data needs to be linearly separable to train.

Even if linearly separable, doesn't necessarily get the best outcome?

9.3.3 Perceptron

Perceptron: one node neural network. is one or zero depednign if weightd inputs enough. therefore is classiication

If error, update weights

Only works if linearly separable. ie can draw linear line completely separting all inputs

Neural network has more layers

Works if data is linear

How to treat node inputs: raw, sigmoid, 0,1

For all of these want the cost function have only one solution, like least squares doe. not guaranteed for all

For logistic, want to make it convex. loss = -log(f(x)) or -log(1-f(x)) depending on correct y. this is convex

How to create node inputs: sigmoid, binary cutoff

9.3.4 Logistic function (AKA sigmoid, logit)

The function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The range of this activation is between 0 and 1.

The derivative

$$\sigma'(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$\sigma'(z) = \sigma(z)\frac{1 + e^{-z} - 1}{1 + e^{-z}}$$

$$\sigma'(z) = \sigma(z)[1 - \sigma(z)]$$

Notes

9.3.5 Probability unit (probit)

The function

The cumulative distribution function of the normal distribution.

$\Phi(z)$

The derivative

The normal distribution:

$\Phi'(z) = \phi(z)$

9.3.6 tanH

9.3.7 ArcTan

9.3.8 Radial Basis Function (RBF) activation function

$a(x) = \sum_i a_i f(||x - c_i||)$

$a(x) = \sum_i a_i f e^{||x-c_i||^2}$

9.3.9 Linear probability model

$p = xB$. can be outside $[0, 1]$.

9.4 Generalised Additive Models (GAMs)

9.4.1 Introduction

Chapter 10

Generalised linear models and multiclass classification

10.1 Multinomial classification

10.1.1 The multinomial data generating process

Introduction

In the binomial case we had:

$z_i = \alpha + \beta x_i + \epsilon_i$

And set y_i to 1 if $z_i > 0$

In the multinomial case we have m alternatives

$z_{ij} = \alpha + \beta x_{ij} + \epsilon_{ij}$

And set $y_{ij} = 1$ if $z_{ij} > z_{ik} \forall k \neq j$

Generalised version

We can rewrite this as:

$z_{ij} = v_{ij} + \epsilon_{ij}$

Where:

$v_{ij} = \alpha + \beta x_{ij}$

In this case v does not depend on j, but in other formulations it could.

Probabilities

$P_{ij} = P(y_{ij} = 1|x_{ij})$

$P_{ij} = P(z_{ij} > z_{ik} \forall k \neq j)$

$P_{ij} = P(\epsilon_{ik} < v_{ij} - v_{ik} + \epsilon_{ij} \forall k \neq j)$

The form of the multinomial model: Intercepts

Previously we described the multinomial model

$z_{ij} = v_{ij} + \epsilon_{ij}$

Where:

$v_{ij} = \alpha + \beta x_{ij}$

The probability of j being chosen is.

$P_{ij} = P(\epsilon_{ik} < v_{ij} - v_{ik} + \epsilon_{ij} \forall k \neq j)$

Interceps in v cancel out. Therefore in the basic model there is no need to use

$v_{ij} = \alpha + \beta x_{ij}$

We can instead use:

$v_{ij} = \beta x_{ij}$

The form of the multinomial model: Conditional model

We have :

$v_{ij} = \beta x_{ij}$

What do we include in x_{ij}?

We can include observable characteristics for each product:

$v_{ij} = \alpha_j + \beta x_j$

One of the α_j must be normalised to 0, as only differences matter. We cannot tell the difference if all α are raised by the same amount.

For consistency with other models we can write this as:

$v_{ij} = \beta x_{ij}$

Even though this does not vary from individual to individual.

Here β represents average preferences for each product characteristic.

The form of the multinomial model: The multinomial model

We have differing characteristics for each individual:

$v_{ij} = \beta x_i$

However this adds a constant for each product. For this to discriminate we need varying coefficients.

$v_{ij} = \beta_j x_i$

As we only observe differences, one of the β_j must be normalised to 0.

We can rewrite this.

$v_{ij} = \sum_k \beta_k \delta_{kj} x_i$

$v_{ij} = \beta z_{ij}$

The original x_i is dense and contains data about the individual.

z_{ij} is sparce and only has entries in the {j} section.

Here β represents how the

The form of the multinomial model: Combined multinomial and conditional model

If we have observations of the characteristics of both individuals and alternatives we can write:

$v_{ij} = \beta_m m_{ij} + \beta_c c_{ij}$

$v_{ij} = \beta x_{ij}$

Here β represents both:

- Average preferences for customer characteristics (conditional)
- How preferences change as individual characteristics change (multinomial)

10.1.2 Extreme IID multinomial

IID

The probability of j being chosen is:

$P_{ij} = P(\epsilon_{ik} < v_{ij} - v_{ik} + \epsilon_{ij} \forall k \neq j)$

If these are independent then we have:

$P_{ij} = \prod_{k \neq j} P(\epsilon_{ik} < v_{ij} - v_{ik} + \epsilon_{ij})$

$P_{ij} = \prod_{k \neq j} F_\epsilon(v_{ij} - v_{ik} + \epsilon_{ij})$

We do not know ϵ_{ij} so we have to integrate over possibilities.

$P_{ij} = \int [\prod_{k \neq j} F_\epsilon(v_{ij} - v_{ik} + \epsilon_{ij})] f_\epsilon(\epsilon_{ij}) d\epsilon_{ij}$

Extreme values

We have:

$P_{ij} = \int [\prod_{k \neq j} F_\epsilon(v_{ij} - v_{ik} + \epsilon_{ij})] f_\epsilon(\epsilon_{ij}) d\epsilon_{ij}$

If ϵ is extreme value type-I this gives us:

$$P_{ij} = \frac{e^{v_{ij}}}{\sum_k e^{v_{ik}}}$$

Independence of irrelevant alternatives

Consider the ratio two probabilities:

$$\frac{P_{ij}}{P_{im}} = \frac{e^{v_{ij}}}{e^{v_{im}}}$$

This means that changes to any other products do not affect relative odds.

This can be undesirable. For example removing one option may cause unbalanced substitution.

For example raising the price of buses may cause more substitution to trains than helicopter, for a commute.

10.1.3 Estimating multinomial logit models

Estimating with individual level data.

Estimating with market share level data.

10.1.4 Nested logit

The probability of j being chosen is:

$P_{ij} = P(\epsilon_{ik} < v_{ij} - v_{ik} + \epsilon_{ij} \forall k \neq j)$

If the error terms are not IID this is more difficult to calculate.

We divide the J alternatives into nests. Within each of these we assume IID error terms, but allow variation between them.

For example we could have a nest of public/private transport. We could have a nest of types of product, and within that the firms offering the product.

The nested logit model does 2 or more seqential IID logit models. One to select the nest, and the other to select the alternative within the nest.

10.1.5 Mixed logit (random coefficients)

Introduction

In our standard model we have:

$z_{ij} = \beta x_{ij} + \epsilon_{ij}$

If we allow the parameters to vary for each individual we have:

$z_{ij} = \beta_i x_{ij} + \epsilon_{ij}$

The probability of choosing j now depends on the distribution of β.

In the IID case we had:

$$P_{ij} = \frac{e^{\beta x_{ij}}}{\sum_k e^{\beta x_{ik}}}$$

Rather than evaluate this at a single point β we integrate.

$$P_{ij} = \int \frac{e^{\beta x_{ij}}}{\sum_k e^{\beta x_{ik}}} f(\beta) d\beta$$

If β is degenerate this reduces to the standard logit model.

10.1.6 Multinomial probit

This relaxes the IID and extreme value assumption.

Errors have a normal variance-covariance matrix.

10.1.7 Softmax

The softmax function is often used in the last layer of a classification network.

It takes a vector of dimension k and returns another vector of the same size. Only, this time all numbers are between 0 and 1 and the values sum to 1.

The softmax function is based on the sigmoid function.

$$a_j(z) = \frac{e^{z_j}}{\sum_i e^{z_i}}$$

10.1.8 Temperature for Softmax

Part III

Supervised machine learning

Chapter 11

Classification trees

11.1 Simple decision trees

11.1.1 Tree traversal

11.1.2 Training decision trees with information gain

We can train a decision tree by starting out with the most simple tree - all outcomes in same node.

We can then do a greedy search to identify which split on the node is best.

We can then iterate this process on future nodes.

Training with information gain

We split nodes to increase maximum entropy.

Entropy is:

$E = -\sum_i^n p_{i=1} \log_2 p_i$

Where we are summing across all nodes.

Information gain

The gain in entropy is the original entropy - weighted by size entropy of each branch

Information gain ratio

11.1.3 Training decision trees with Gini impurity

11.1.4 Pruning decision trees

Training a decision tree until there is only one entry from the training set will result in overfitting.

We can use pruning to regularise trees.

Pruning

Reduced error pruning

From bottom, replace each node with a leaf of the most popular class. Accept if no reduction in accuracy.

Cost complexity pruning

Take full tree T_0

Iteratively find a subtree to replace with a leaf. Cost function is accuracy and number of leaves.

Remove this generating T_{i+1}

When we have just the root, choose a single tree using CV.

Growing and pruning

Generally we would split the data up. Grow the tree with one set and then prune with the other.

We can split our data up and iterate between growing and pruning.

When pruning, for each pair of leaves we test to see if they should be merged.

If our two sets are A and B we can do:

- A: Grow
- B: Prune
- B: Grow
- A: Prune

And repeat this process.

Partial regression trees

Once we have built a tree, we keep a single leaf and disard the rest.

11.1.5 Decision trees with many classes

Training decision trees with many classes

11.2 Other

11.2.1 Training with unbalanced data

Unbalanced dataset: more of one class than others.

Can reduce sample of majorit, or synthetically generate minority.

Chapter 12

Regression trees

12.1 Regression trees

12.1.1 Classic regression trees

In a classical regression tree, we follow a decison process as before, but the outcome is real number.

Within each leaf, all inputs are assigned that same number.

Training

With a regression problem we cannot split nodes the same way as we did for classification.

Instead by split by the residual sum of squares.

12.1.2 Training decision trees with Mean Squared Error (MSE)

12.1.3 Mixed regression trees

In classical trees all items in a leaf are assigned the same values. In this model, all are given θ for a parametric model.

This makes the resulting trees smoother.

We have some $\hat{y}_i = f(\mathbf{x}_i, \theta) + \epsilon$

The approach generalises classic regression trees. There the estimate was $\bar{y}$. Here it's a regression.

Training

At each node we do OLS. If the R^2 of the model is less than some constant, we find a split which maximises the minimum of the two new R^2.

12.1.4 Classifying with probabilistic decision trees

Previously our decision tree classifier was binary.

We can instead adapt the mixed tree model and using a probit model at each leaf.

Chapter 13

Bayesian trees

13.1 Bayesian trees

13.1.1 Priors of trees

Priors for simple trees

We can define a tree as a set of nodes: T.

For each node we define a splitting variable k and a splitting threshold r.

Our prior is $P(T, k, r)$.

We split this up to:

$P(T, k, r) = P(T)P(k, r|T)$

$P(T, k, r) = P(T)P(k|T)P(r|T, k)$

So we want to estimate:

- $P(T)$ - The number of nodes.
- $P(k|T)$ - Which variables we split by, given the tree size.
- $P(r|T, k)$ - The cutoff, given the tree size and the variables we are splitting by.

Priors for mixed trees

If at the leaf we have a parametric model, our prior is instead:

$P(T, k, r, \theta) = P(T)P(k|T)P(r|T, k)P(\theta|T, k, r)$

We then need to additionally estimate $P(\theta|T, k, r)$.

13.1.2 The pinball prior

We can generate a tree with a fixed number of leaves, according to our prior.

As we start the tree we assocaite the root node with a count of all leaves.

As we split a node, the remaining leaf counts are divided between the directions. If there is only one leaf left, we do no further splitting.

13.1.3 Estimating other priors

13.1.4 Bayesian CART

Our prior

Call the collective parameters of the tree $\Theta = (T, k, r)$ and θ.

Collectively our prior is defined by $P(\Theta)$ and $P(\theta)$

Bayes' theorem

We want to know the posterior given our data X.

$$P(\Theta|X) = \frac{P(X|\Theta)P(\Theta)}{P(X)}$$

$$P(\Theta|X) \propto P(X|\Theta)P(\Theta)$$

Expanded posterior

We know explore $P(X|\Theta)$

$P(X|\Theta) = \int P(X|\theta, \Theta)P(\theta)d\theta$

This means our posterior is:

$P(\Theta|X) \propto P(\Theta) \int P(X|\theta, \Theta)P(\theta)d\theta$

Estimation

This can be estimated with MCMC.

Chapter 14

Support Vector Machines (SVMs)

14.1 Linear Support Vector Classifiers (SVCs)

14.1.1 Hard-margin SVC

Linear separators

We want to create a hyperplane to separate classes.

For classification problem (x, y)

Hyperplane is wx-b=0

Hard margin

If data is linearly separable then a hyperplane exists such that all data can be correctly classified

There are an infinit number that could work.

We select two parallel with distance between as large as possible. the region between these two is the margin

The maximum margin hyperplane is the one between the two margin planes

We can rescale the two hyperplanes to:

wx-b=1

wx-b=-1

The distance between the two parrallel hyperplanes is $\frac{2}{||w||}$

So we minimise $||w||$ conditional an all points being correctly classified

$y_i(wx_i - b) \geq 1$

We select w and b to solve this.

14.1.2 Support vectors

Support vectors are those that make up the classifer boundry.

14.1.3 Estimating the SVC using quadratic equations

14.1.4 Soft-margin SVC

Soft margin

Soft margin

Data may not be linearly separable, so we introduce a hinge loss function

$Max(0, 1 - y_i(wx - b))$

We then minimise

$\lambda||w||^2 + [\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(wx_i - b)]$

This introduces λ as a parameter.

14.1.5 Regularising the SVC

14.2 Multiple classes

14.2.1 Support vector classifiers for multiple classes

14.3 Non-linear support vector classifiers

14.3.1 The dot product SVC

14.3.2 The kernel trick

We can use kernels as an alternative to the dot product.

14.3.3 The radial basis function (RBF) SVC

Chapter 15

Variational Bayes

15.1 Variational Bayes

15.1.1 Introduction

Chapter 16

The Naive Bayes classifier

16.1 Naive Bayes

16.1.1 The Naive Bayes posterior

Bayes theorem

Consider Bayes' theorem

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|y)P(y)}{P(x_1, x_2, ..., x_n)}$$

Here, y is the label, and $x_1, x_2, ..., x_n$ is the evidence. We want to know the probability of each label given evidence.

The denominator, $P(x_1, x_2, ..., x_n)$, is the same for all, so we only need to identify:

$$P(y|x_1, x_2, ..., x_n) \propto P(x_1, x_2, ..., x_n|y)P(y)$$

The assumption of Naive Bayes

We assume each x is independent. Therefore:

$$P(x_1, x_2, ..., x_n|y) = P(x_1|y)P(x_2|y)...P(x_n|y)$$

$$P(y|x_1, x_2, ..., x_n) \propto P(x_1|y)P(x_2|y)...P(x_n|y)P(y)$$

16.1.2 The Naive Bayes classifier

Calculating the Naive Bayes estimator

With the Naive Bayes assumption we have:

$$P(y|x_1, x_2, ..., x_n) \propto P(x_1|y)P(x_2|y)...P(x_n|y)P(y)$$

We now choose y which maximises this.

This is easier to calculate, as there is less of a sample restriction.

This is used when evidence is also in classes, as the chance of any individual outcome on a continuous probability is 0.

Estimating $P(y)$

We can easily calculate $P(y)$, by looking at the frequency across the sample.

Estimating $P(x_1|y)$

Normally, $P(x_1|y) = \frac{n_c}{n_y}$, where:

- n_c is the number of instances where the evidence is c and the label is y.
- n_y is the number of instances where the label is y.

Regularising the Naive Bayes estimator

To reduce the risk of specific probabilities being zero, we can adjust them, so that:

$P(x_1|y) = \frac{n_c + mp}{n_y + m}$, where:

- p is the prior probability. If this is unknown, use $\frac{1}{k}$, where k is the number of classes.
- m is a parameter called the equivilant sample size.

16.1.3 Text classification using Naive Bayes

Naive Bayes and text classification

Naive Bayes can be used to classify text documents. The x variables can be appearances of each word, and y can be the document classification.

Chapter 17

The K-Nearest Neighbours (KNN) classifier

17.1 K-Nearest Neighbours (KNN)

17.1.1 K-nearest neighbours

K-nearest neighbours is a non-parametric classifer. For a point with an unknown class we identify the classes of the K-nearest neighbours and assign the most common class.

For this we need to find the distance between observations. We can do this using norms.

Weightings of the dependent variables is important here.

Practicality

Requires space to store

N samples, d features

times: O(n.d)

17.1.2 Choosing K for K-Nearest Neighbours

Chapter 18

Discriminant analysis

18.1 Discriminant analysis

18.1.1 Linear Discriminant Analysis (LDA)

Assume data is mixed gaussian. Use this to estimate classes.

18.1.2 Kernel Fisher discriminant analysis

Use LDA with kernel feature spaces.

Chapter 19

Non-parametric regression

19.1 Kernel regression

19.1.1 Kernel regression

Introduction

For parametric regression we have:

$y = f(X)$

Where the form of $f(X)$ is fixed, such as for linear regression.

For non-parametric regression we have:

$y = m(X)$

Where $m(X)$ is not fixed.

We can estimate $m(X)$ using kernel regression.

$$m(X) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{i=1}^{n} K_h(x - x_i)}$$

We know this because we have:

$$E(y|X) = \int y f(y|x) dy = \int y \frac{f(x, y)}{f(x)} dy$$

We then use kernel density estimation for both.

19.2 Splines

19.2.1 Multivariate Adaptive Regression Splines (MARS)

A linear model looks like:

$\hat{y} = c + \sum_i x_i \theta_i$

MARS instead produces a linear model for subsets of X.

$\hat{y} = c + \sum_i B_j(x_i, a_j)\theta_i$

Where:

- $B_j = max(0, x_i - a_j)$; or
- $B_j = -max(0, a_j - x_i)$

This is trained using a forward pass and a backward pass.

Forward pass

Backward pass

19.2.2 Bayesian splines

19.3 Other

19.3.1 Local regression

19.3.2 LOWESS

19.3.3 LOESS

19.3.4 Kernel regression

Quantile regression

In other supervised?

Normally we return a central estimate, commonly the mean.

Quantile regression returns an estimate of the ith quartile instead.

Goal is to find xth quartile of variance.

Linear quantile regression

Tree quantile regression

19.3.5 Principal component regression

Do PCA on X.

Do OLS with this.

Transform parameters by reversing PCA procedure on parameters.

19.3.6 Partial least squares regression

This expands on principal component regression.

Both X and Y are mapped to new spaces.

Chapter 20

Ensemble methods

20.1 Combining learners

20.1.1 Majority voting

Combining classifiers

If we generate different classifiers then each can give different predictions for the same input.

If we have different predictions how should we proceed?

Using one model or multiple models

It is not obvious that we should want to use more than one model. If one model was superior to another then there may be no benefit to using the information from the additional model.

However if the errors in different models are varied, then combining multiple models can lead to better performance as each individual model can have unique information.

Majority voting

One approach to using different predictions is to use majority voting.

If we have a collection of hard classifiers then we choose the classification with the most votes.

Condorcet's Jury Theorem

Consider a collection of classifiers. For each classifier there is a chance p_i of the classification being correct.

If the voters are independent

If voters are independent, and the chance of one vote being right is greater than 0.5, then the more voters, the better.

A weak model can still be useful, if it is independent.

20.1.2 Averaging regression predictions

If we have multiple predictions, we can take an average of these, possibly weighted.

We have m regressors $g_j(\mathbf{x}_i)$.

Our output is:

$h(\mathbf{x}_i) = \sum_j w_j g_j(\mathbf{x}_i)$

20.1.3 Stacking and SuperLearner

Introduction

With stacking we take the predictions from each of our classifiers, and then train a new model using these predictions as inputs.

Hard and soft inputs

Hard classification (0 or 1) and soft classification (between 0 and 1) can be used as inputs.

Cross validation

We can select hyper-parameters using cross validation, however there is an issue of using cross validation twice on the same data. Once for the underyling classifiers, and again for the stacked model.

SuperLearner

We have m models.

Part 1: Train each of them on all data.

Part 2: Split the data into k sets

For each set, associate all other data as training

For each fold:

- Fit each model on the training data
- Predict on other
- Create weighted predictor. choose weightings to minimise error

Part 3: Use these weightings on the original (unrestricted data) model

20.2 Generating learners with boosting

20.2.1 L_2 boosting

20.2.2 Adaboost

Introduction

Boosting is a way to create multiple learners for use in an ensemble predictor.

The goal is to create many predictors, which may not be themselves very accuracte, but have a high degree of independence.

AdaBoost

AdaBoost is a popular algorithm for boosting.

It works by:

- Creating a set of weak learners using different restrictions on features in the training data.
- Choosing the weak learner that most reduces the error of the combined learners, and give it a weighting which most reduces the error of the combined learners.
- Creating a new weighting for the dataset, where ones poorly predicted (by the combination of learners) are given high weights.
- Repeating the process a fixed number of times.

20.2.3 Gradient boosting

Gradient boosting does not iterative change the weights for the learners. Instead, it trains on different errors.

While AdaBoost trains to reduce the absolute error for each weak classifier, gradient boosting trains on the difference between the actual classiffication and the current classification.

20.3 Generating learners with Bootstrapped AGGregation (bagging)

20.3.1 Bagging

Introduction

Bagging, or Bootstrap AGGregation, is a way of generating weak learners.

Bootstrapping

Bootstraping refers to taking samples with repacement from the training set. This is how the datasets for each of the weak learners are formed.

How to do bagging

We take samples from the training set, with replacement, and train each of these separately. This gives us our weak learners.

Chapter 21

Ensemble methods for trees

21.1 Ensemble methods for trees

21.1.1 Gradient tree boosting

This applies gradient boosting to tree.

21.1.2 Multiple Additive Regression Trees (MART)

21.1.3 Bayesian Additive Regression Trees (BART)

21.1.4 Extra trees

21.1.5 Random forests

These use bagging techniques with random trees.

At each node, rather than sample the whole data we sample a random selection.

Get d dimensions, and sample m of them at each node.

Choose $m \leq \sqrt{d}$

21.1.6 Regression forests

Chapter 22

Regularising black box models

22.1 Regularising classifiers

22.1.1 Label smoothing

Regualrising of classifiers assume some incorrect (similar for regression?)

Chapter 23

Confidence intervals of black box models

23.1 Confidence intervals of black box models

23.1.1 Introduction

ensemble confidence intervals non-parametric confidence intervals semi-parametric confidence intervals

one-sided, 2 sided confidence intervals

jackknife for bagging confidence interval

23.2 Bootstrapping moments of ensemble statistics

23.2.1 Bootstrapping mean and variance

Sample size selection

23.2.2 Bootstrapping confidence intervals

Need to know estimate is unbiased for this.

Chapter 24

Interpreting black box models

24.1 Interpreting black box models

24.1.1 Introduction

partial dependence plots can be used on black box models

Interpretation: if its interpretable then you can adjust an interpretable part manually part way?

Transparency of models: Sparse linear models are more transparent. Decomposition: Can each part of the model have input, output, parameters which can be interpreted? Complex feature selection means loss of this. Complex models. Boosting. Loses Can we say formal things about performance? We can for linear models (?), but not for others For agents, we can't validate their behaviour. We can for manually defined rules. We can for interpretive models. Post-hoc interpretability

LIME Local Interpretable Model-Agnostic Explanations. Page on explainable models, h3 in that on locally explainable models

Explaining models: We may want to understand how it works. Black box algorithms are hard to understand.

This is important if the algorithm is used in high stakes cases, or where data is different to the static case used for training.

We can create explainable models (sparse linear models)

Take black box models and make them explainable.

SLAM algorithms?

Local explanation? Saliency maps?

Why do we care about transparency?

Chapter 25

Semi-supervised learning

25.1 Introduction

25.1.1 Introduction

We seek to make a discriminative model using both labelled an unlabelled data.

25.1.2 Generative models

25.1.3 Low density separation

Chapter 26

Imputing missing data for prediction

26.1 Introduction

26.1.1 Deleting observations

Deleting whole row if missing data (bias if not random)

26.1.2 Linear interpolation

26.1.3 Missing Completely At Random (MCAR)

Different to MAR

26.1.4 Missing at Random (MAR)

26.1.5 Other

Interpolation: mean, conditional (in IID, different for time series)

Chapter 27

Recommenders

27.1 Non-negative matrix factorisation

27.1.1 Non-negative matrix factorisation

27.2 Recommenders

27.2.1 The recommendation problem

We have a collection of users and a collection of products. We want to recommend products to customers.

Once a customer "consumes" something, we get feedback. however for vast majority of items, not consumed. goal: predict feedback score beforehand to make good recommendations.

We have a matrix of how much each customer "likes" things. however this is v sparce.

27.3 Approaches

27.3.1 Content filtering

We have metadata on customers and products. Eg stated preferences, genres, actors etc.

We use this to create recommendations.

27.3.2 Collaborative filtering

We look at the actions of customers. We then recommend things based on customers who are similar.

Doesn't need stated preferences, or metadata on content.

Needs data on customers (behavours, etc)

2 steps:

- Look for similar users
- Use ratings from those users to make predictions for missing items

27.3.3 Cold start

When you start you have no data on views. This is the cold start problem.

Part IV

SORT

Chapter 28

SORT 2025

28.1 Introduction

28.1.1 Introduction

somewhere near quantisation: mixture of experts (MoE) compression page on Mixture of Experts models generally? + confirm - are these specific to LLMs, or neural networks more generally? see to be more general?

LightGBM Light Gradient Boosting Machine

gram matrix to get kernel (stats stuff) page on boosting. xgboost? page on one-shot, zero-shot and few-shot learning.

28.1.2 linear regression for inference

ordinary linear regression for inference split out: + gauss markov

28.1.3 estimating discriminative

Point variable estimates for discriminative models: + split out

page on Multilevel regression with poststratification (MRP)

28.1.4 supervised machine learning

split out ensemble methods: + boosting and bagging

something on additive models around non-parametric regression

split out non-parametric regression. many of those should be in linear

28.1.5 advanced inference

xlearner tlearner

28.1.6 supervised linear regression

ordinary least squares for prediction split out + frisch-waugh-lovell + bayesian linear regression to bayrsian bit above + least trimmed squares + linear models + impact of outliers (leverage and cook's distance)

page on feature engineering