MS&E 125: Intro to Applied Statistics

Feature Engineering

Professor Udell

Management Science and Engineering
Stanford

May 10, 2023

Announcements

Outline

Supervised learning setup

- input space $\mathcal{X}$
  - $x \in \mathcal{X}$ is called the **covariate**, **feature**, or **independent variable**
- output space $\mathcal{Y}$
  - $y \in \mathcal{Y}$ is called the **response**, **outcome**, **label**, or **dependent variable**
- given $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$
  - $\mathcal{D}$ is called the **data**, **examples**, **observations**, **samples** or **measurements**
- we will find some $h \in \mathcal{H}$ so that (we hope!)

$$h(x_i) \approx y_i, \quad i = 1, \dots, n$$

Supervised learning

different names for different $\mathcal{Y}$s:

- **classification:** $\mathcal{Y} = \{-1, 1\}$
- **regression:** $\mathcal{Y} = \mathbf{R}$
- **multiclass classification:** $\mathcal{Y} = \{\text{car}, \text{pedestrian}, \text{bike}\}$
- **ordinal regression:**
  $\mathcal{Y} = \{\text{strongly disagree}, \dots, \text{strongly agree}\}$

Regression

examples where $\mathcal{Y} = \mathbf{R}$:

- predict credit score of applicant
- predict temperature at Stanford a year from today
- predict height of child given height of parents
- predict price of house given location, square footage, ...
- predict demand for electricity given temperature

Regression

examples where $\mathcal{Y} = \mathbf{R}$:

- predict credit score of applicant
- predict temperature at Stanford a year from today
- predict height of child given height of parents
- predict price of house given location, square footage, ...
- predict demand for electricity given temperature

careful: are all real number valid predictions?

Outline

Linear models

To fit a linear model ($=$ linear in parameters β)

- pick a transformation $\phi : \mathcal{X} \to \mathbf{R}^p$
- predict y using a linear function of $\phi(x)$

$$\hat{y} = \phi(x)^T \beta = \sum_{i=1}^{p} \beta_i (\phi(x))_i$$

Feature engineering

How to pick $\phi : \mathcal{X} \to \mathbf{R}^d$?

- so response y will depend linearly on $\phi(x)$
- so number of features p is not too big

Feature engineering

How to pick $\phi : \mathcal{X} \to \mathbf{R}^d$?

- so response y will depend linearly on $\phi(x)$
- so number of features p is not too big

if you think this looks like a hack, you're right!

Feature engineering

examples:

- adding offset
- standardizing features
- polynomials
- transforming Booleans, ordinals, nominals
- handling missing values
- ensuring positive predictions
- transforming images, text, location
- concatenating data
- all of the above

`https://xkcd.com/2048/`

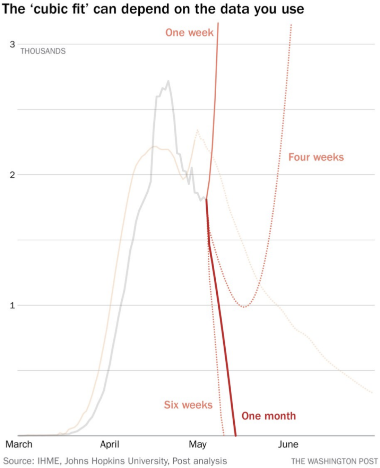
Outline

- Supervised learning
- Feature engineering
- Polynomial transformations
- Boolean, nominal, ordinal
- Missing values
- Nonlinear transformations
- Location
- Text, images, ...

Fitting a polynomial

- $\mathcal{X} = \mathbf{R}$
- let
  $$\phi(x) = (1, x, x^2, x^3, \dots, x^{p-1})$$
  be the vector of all monomials in x of degree $< p$
- now $\hat{y} = \beta^T \phi(x) = \beta_1 + \beta_2 x + \beta_3 x^2 + \cdots + \beta_p x^{p-1}$

IMHE and the cubic fit



https://www.washingtonpost.com/politics/2020/05/05/
white-houses-self-serving-approach-estimating-deadliness-

Demo: crime

`https://colab.research.google.com/github/stanford-mse-125/demos/blob/main/crime.ipynb`

Model evaluation

how should we measure how good a model is?

- (root) mean squared error (RMSE)
- mean absolute error (MAE)
- coefficient of determination (R^2)

Mean square error

mean square error is minimized by the least squares estimator

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

equal to the sum of the residuals squared

Root mean square error

root mean square error is the square root of the mean square error

$$\hat{\sigma}^2 = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

(the residual standard error is similar, but normalizes by the **residual degrees of freedom** $n - p - 1$ instead of n)

Mean absolute error

mean absolute error is the mean of the absolute value of the residuals

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

often makes more sense than RMSE when we care about quality of the predictions
(e.g., if we will pay a linear penalty for being wrong)

Coefficient of determination

coefficient of determination $R^2 \in [0, 1]$ is the fraction of the variance in the data that is explained by the model

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} = 1 - \frac{\text{MSE}}{\text{Var}(y)} = 1 - \frac{\text{SSR}}{\text{SST}}$$

lingo:

- SSR is the **sum of squares of the residuals**
- SST is the **total sum of squares**

for a model with an intercept, R^2 is the square correlation between the predicted and true values of y

$$R^2 = [\rho(y, \hat{y})]^2$$

Outline

Notation: boolean indicator function

define

$$\mathbb{1}(\text{statement}) = \begin{cases} 1 & \text{statement is true} \\ 0 & \text{statement is false} \end{cases}$$

examples:

- $\mathbb{1}(1 < 0) = 0$
- $\mathbb{1}(17 = 17) = 1$

Boolean variables

- $\mathcal{X} = \{\text{true}, \text{false}\}$
- let $\phi(x) = \mathbb{1}(x)$

Nominal values: one-hot encoding

- nominal data: *e.g.*, $\mathcal{X} = \{\text{apple}, \text{orange}, \text{banana}\}$
- let

$$\phi(x) = [\mathbb{1}(x = \text{apple}), \mathbb{1}(x = \text{orange}), \mathbb{1}(x = \text{banana})]$$

- called **one-hot encoding**: only one element is non-zero

Nominal values: one-hot encoding

- nominal data: *e.g.*, $\mathcal{X} = \{\text{apple}, \text{orange}, \text{banana}\}$
- let

$$\phi(x) = [\mathbb{1}(x = \text{apple}), \mathbb{1}(x = \text{orange}), \mathbb{1}(x = \text{banana})]$$

- called **one-hot encoding**: only one element is non-zero

extension: sets

Demo: crime

https://colab.research.google.com/github/stanford-mse-125/demos/blob/main/crime.ipynb

Nominal values: the long tail

- **problem:** too many nominal categories
- **solution:**

Nominal values: the long tail

- **problem:** too many nominal categories
- **solution:**
  - cluster the categories by some known ontology (eg, "squamous cell carcinoma" $\rightarrow$ "cancer")

Nominal values: the long tail

- **problem:** too many nominal categories
- **solution:**
  - cluster the categories by some known ontology (eg, “squamous cell carcinoma” $\rightarrow$ “cancer”)
  - lump the least common categories into a single category: “Other”

Nominal values: the long tail

- **problem:** too many nominal categories
- **solution:**
  - cluster the categories by some known ontology (eg, "squamous cell carcinoma" $\rightarrow$ "cancer")
  - lump the least common categories into a single category: "Other"
  - feature hashing

Nominal values: the long tail

- **problem:** too many nominal categories
- **solution:**
  - cluster the categories by some known ontology (eg, "squamous cell carcinoma" $\rightarrow$ "cancer")
  - lump the least common categories into a single category: "Other"
  - feature hashing
  - ... be creative!

Nominal values: look up features!

why not use other information known about each item?

- $\mathcal{X} = \{\text{apple}, \text{orange}, \text{banana}\}$
  - price, calories, weight, ...
- $\mathcal{X} =$ zip code
  - average income, temperature in July, walk score, ...
- ...

database lingo: **join** tables on nominal value

Ordinal values: real encoding

- ordinal data: *e.g.*, $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- let

$$\phi(x) = \begin{cases} 1, & x = \text{Stage I} \\ 2, & x = \text{Stage II} \\ 3, & x = \text{Stage III} \\ 4, & x = \text{Stage IV} \end{cases}$$

- default encoding

Ordinal values: real encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- use real encoding ϕ to transform ordinal data
- fit linear model with offset to predict y as $\beta_0 + \beta_1\phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.

Ordinal values: real encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- use real encoding ϕ to transform ordinal data
- fit linear model with offset to predict y as $\beta_0 + \beta_1 \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.
Q: What is β_0? β_1?

Ordinal values: real encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- use real encoding ϕ to transform ordinal data
- fit linear model with offset to predict y as $\beta_0 + \beta_1 \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years: $\beta_0 = 6$, $\beta_1 = -2$.

Ordinal values: real encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- use real encoding ϕ to transform ordinal data
- fit linear model with offset to predict y as $\beta_0 + \beta_1 \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years: $\beta_0 = 6$, $\beta_1 = -2$.

Q: How long does the model predict a person with Stage IV cancer will survive?

Ordinal values: real encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- use real encoding ϕ to transform ordinal data
- fit linear model with offset to predict y as $\beta_0 + \beta_1\phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years: $\beta_0 = 6$, $\beta_1 = -2$.

Q: How long does the model predict a person with Stage IV cancer will survive?

A. 6 years

B. 2 years

C. 0 years

D. -2 years

Ordinal values: boolean encoding

- ordinal data: *e.g.*, $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- let

$$\phi(x) = [\mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

Ordinal values: boolean encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- define transformation $\phi : \mathcal{X} \to \mathbf{R}$ (with offset) as

$$\phi(x) = [1, \mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

- fit linear model with offset to predict y as $\beta^T \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.

Ordinal values: boolean encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- define transformation $\phi : \mathcal{X} \to \mathbf{R}$ (with offset) as

$$\phi(x) = [1, \mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

- fit linear model with offset to predict y as $\beta^T \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.
Q: What is β?

Ordinal values: boolean encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- define transformation $\phi : \mathcal{X} \to \mathbf{R}$ (with offset) as

$$\phi(x) = [1, \mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

- fit linear model with offset to predict y as $\beta^T \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.

Q: What is β?

A: $\beta_0 = 4$, $\beta_1 = -2$, β_2 and β_3 not determined

Ordinal values: boolean encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- define transformation $\phi : \mathcal{X} \to \mathbf{R}$ (with offset) as

$$\phi(x) = [1, \mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

- fit linear model with offset to predict y as $\beta^T \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.
Q: What is β?
A: $\beta_0 = 4$, $\beta_1 = -2$, β_2 and β_3 not determined
Q: How long does the model predict a persion with Stage IV cancer will survive?

Ordinal values: boolean encoding

- $\mathcal{X} = \{\text{Stage I}, \text{Stage II}, \text{Stage III}, \text{Stage IV}\}$
- $\mathcal{Y} = \mathbf{R}$, number of years lived after diagnosis
- define transformation $\phi : \mathcal{X} \to \mathbf{R}$ (with offset) as

$$\phi(x) = [1, \mathbb{1}(x \geq \text{Stage II}), \mathbb{1}(x \geq \text{Stage III}), \mathbb{1}(x \geq \text{Stage IV})]$$

- fit linear model with offset to predict y as $\beta^T \phi(x)$

Suppose model predicts a person diagnosed with Stage II cancer will survive 2 more years, and a person diagnosed with Stage I cancer will survive 4 more years.

Q: What is β?

A: $\beta_0 = 4$, $\beta_1 = -2$, β_2 and β_3 not determined

Q: How long does the model predict a persion with Stage IV cancer will survive?

A: can't say without more information

Outline

Supervised learning

Feature engineering

Polynomial transformations

Boolean, nominal, ordinal

Missing values

Nonlinear transformations

Location

Text, images, . . .

Missing values

handling missing values:

- remove rows/columns with missing entries

Missing values

handling missing values:

- remove rows/columns with missing entries
- (for time series) back-fill with most recent observed value

Missing values

handling missing values:

- remove rows/columns with missing entries
- (for time series) back-fill with most recent observed value
- impute with mean, median, or mode

Missing values

handling missing values:

- remove rows/columns with missing entries
- (for time series) back-fill with most recent observed value
- impute with mean, median, or mode
- fancier imputation methods (covered later in this class): matrix completion, copula models, deep learning, ...

Missing values

handling missing values:

- remove rows/columns with missing entries
- (for time series) back-fill with most recent observed value
- impute with mean, median, or mode
- fancier imputation methods (covered later in this class): matrix completion, copula models, deep learning, ...
- add new feature: Boolean indicator $\mathbb{1}$(data is missing)
  - can detect if missingness is informative
  - can complement imputation method
  - can use different indicators for different kinds of missingness (refused, missing, illegible response, ...)

Poll

In an ambulance dataset (data taken by instruments on board an ambulance), we want to predict if the patient died. The variable “heart rate” is sometimes missing. Is missingness

A. informative?

B. uninformative?

Poll

In a weather dataset, the batteries in the instruments occasionally run out before the experimenter can replace them, leaving missing data for eg temperature, humidity, or barometric pressure. Is missingness

A. informative?

B. uninformative?

Talk to your neighbor

Can you think of a dataset in which missing values would be

- informative?
- uninformative?

Outline

Nonlinear transformations

sometimes data is easy to predict with a simple but **non**linear relation, e.g.

$$\log(y) = x^T \beta$$

can transform x or (even more important) y

Nonlinear transformations

sometimes data is easy to predict with a simple but **non**linear relation, e.g.

$$\log(y) = x^T \beta$$

can transform x or (even more important) y

hints that your data might benefit from a nonlinear transform:

- y is positive and heavy-tailed? try $y \leftarrow \log(y)$
- residuals $r = y - x_i^T \beta$ are skewed (not normal)

Nonlinear transformations

sometimes data is easy to predict with a simple but **non**linear relation, e.g.

$$\log(y) = x^T \beta$$

can transform x or (even more important) y

hints that your data might benefit from a nonlinear transform:

- y is positive and heavy-tailed? try $y \leftarrow \log(y)$
- residuals $r = y - x_i^T \beta$ are skewed (not normal)

useful nonlinear transforms:

- log, exp, quantile, ...

Nonlinear transformations

sometimes data is easy to predict with a simple but **non**linear relation, e.g.

$$\log(y) = x^T \beta$$

can transform x or (even more important) y

hints that your data might benefit from a nonlinear transform:

- y is positive and heavy-tailed? try $y \leftarrow \log(y)$
- residuals $r = y - x_i^T \beta$ are skewed (not normal)

useful nonlinear transforms:

- log, exp, quantile, . . .

Q: which of these might benefit from a log transformation?

Log transform

Q: what happens if x increases by 1 in the model

$$\log(y) = \beta_0 + \beta_1 x,$$

Log transform

Q: what happens if x increases by 1 in the model

$$\log(y) = \beta_0 + \beta_1 x,$$

A: $\log(y)$ increases by β_1, so y increases by $\exp(\beta_1)$

$$\begin{aligned}
\log(y) = \beta_0 + \beta_1 x \quad &\implies \quad y = \exp(\beta_0 + \beta_1 x) \\
\log(y') = \beta_0 + \beta_1 (x+1) \quad &\implies \quad y' = \exp(\beta_0 + \beta_1 (x+1)) \\
& \quad y' = \exp(\beta_0 + \beta_1 x)\exp(\beta_1)
\end{aligned}$$

A convenient approximation

- for small x, $\exp(x) \approx 1 + x$,
- *e.g.*, $\exp(0.01) \approx 1.01$
- if x increases by 1%, then y increases by factor of $\exp(\beta_1/100)$
- so if x increases by 1%, then y increases by factor of $\approx \beta_1/100 = \beta_1\%$

Log transformations of covariates

if we instead log transform x, $\hat{y}$ increases by $\beta_1/100$ for each 1% increase in x.

- *e.g.*, if $\beta_1 = 3$, $\hat{y}$ increases by 3/100=0.03 units for every 1% increase in x.

if we instead log transform both x and y, $\hat{y}$ increases by β_1% for each 1% increase in x.

- *e.g.*, if $\beta_1 = 3$, $\hat{y}$ increases by 3% for every 1% increase in x.

log transformation results in **multiplicative** increases (rather than **additive**)

Outline

Location

can be given as

- latitude, longitude
- zip code
- neighborhood, county, state, country

can be transformed between these!

Location

can be given as

- latitude, longitude
- zip code
- neighborhood, county, state, country

can be transformed between these!

which makes sense for your problem?

- does nearness matter?
- are there sharp boundaries?
- are other properties of the location (eg, mean house price or crime rate) more important?

Outline

Supervised learning

Feature engineering

Polynomial transformations

Boolean, nominal, ordinal

Missing values

Nonlinear transformations

Location

Text, images, . . .

Text

$\mathcal{X}$ = sentences, documents, tweets, ...

- **bag of words** model (one-hot encoding):
  - pick set of words $\{\beta_1, \ldots, \beta_d\}$
  - $\phi(x) = [\mathbb{1}(x \text{ contains } \beta_1), \ldots, \mathbb{1}(x \text{ contains } \beta_d)]$
  - ignores order of words in sentence

Text

$\mathcal{X}$ = sentences, documents, tweets, ...

- **bag of words** model (one-hot encoding):
  - pick set of words $\{\beta_1, \ldots, \beta_d\}$
  - $\phi(x) = [\mathbb{1}(x \text{ contains } \beta_1), \ldots, \mathbb{1}(x \text{ contains } \beta_d)]$
  - ignores order of words in sentence
- **pre-trained neural networks**:
  - sentiment analysis: `https://medium.com/@b.terryjack/nlp-pre-trained-sentiment-analysis-1eb52a9d742c`
  - Universal Sentence Encoder (USE) embedding: `https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/semantic_similarity_with_tf_hub_universal_encoder.ipynb`
  - lots of others: `https://modelzoo.co/`

Neural networks: whirlwind primer

$$\mathrm{NN}(x) = \sigma(W_1\sigma(W_2 \ldots \sigma(W_\ell x))))$$

- σ is a nonlinearity applied elementwise to a vector, *e.g.*
  - ReLU: $\sigma(x) = \max(x, 0)$
  - sigmoid: $\sigma(x) = \log(1 + \exp(x))$
- each W is a matrix of parameters
- trained on very large datasets, *e.g.*, Wikipedia, YouTube

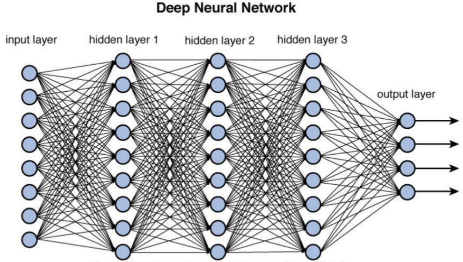
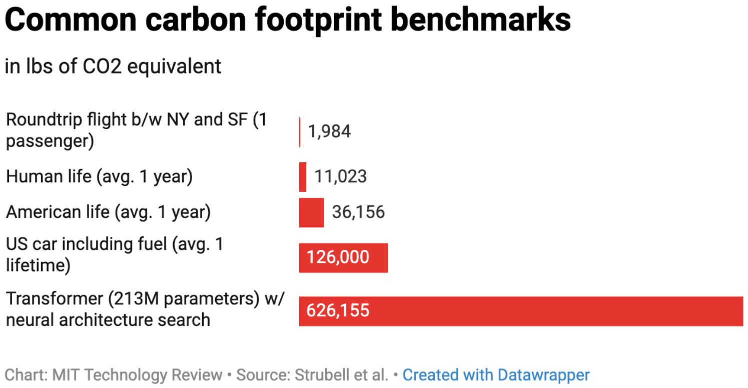


Figure 12.2 Deep network architecture with multiple layers.

Why not use deep learning?



towards a solution: `https://arxiv.org/abs/1907.10597`

Review

- linear models are linear in the **parameters** β
- can fit many different models by picking feature mapping $\phi : \mathcal{X} \to \mathbf{R}^d$