Features

Sanjay Lall and Stephen Boyd

EE104
Stanford University
Records and embedding
Raw data

- raw data pairs are \((u, v)\), with \(u \in U\), \(v \in V\)
- \(U\) is set of all possible input values
- \(V\) is set of all possible output values
- each \(u\) is called a record
- typically a record is a tuple, or list, \(u = (u_1, u_2, \ldots, u_r)\)
- each \(u_i\) is a field or component, which has a type, e.g., real number, Boolean, categorical, ordinal, word, text, audio, image, parse tree (more on this later)
- e.g., a record for a house for sale might consist of

  (address, photo, description, house/apartment?, lot size, \ldots, \# bedrooms)
Feature map

- learning algorithms are applied to \((x, y)\) pairs,

\[
x = \phi(u), \quad y = \psi(v)
\]

- \(\phi: \mathcal{U} \rightarrow \mathbb{R}^d\) is the feature map for \(u\)

- \(\psi: \mathcal{V} \rightarrow \mathbb{R}\) is the feature map for \(v\)

- feature maps transform records into vectors

- feature maps usually work on each field separately,

\[
\phi(u_1, \ldots, u_r) = (\phi_1(u_1), \ldots, \phi_r(u_r))
\]

- \(\phi_i\) is an embedding of the type of field \(i\) into a vector
Embeddings

- embedding puts the different field types on an equal footing, i.e., vectors
- some embeddings are simple, e.g.,
  - for a number field \((\mathcal{U} = \mathbb{R})\), \(\phi_i(u_i) = u_i\)
  - for a Boolean field, \(\phi_i(u_i) = \begin{cases} 1 & u_i = \text{TRUE} \\ -1 & u_i = \text{FALSE} \end{cases}\)
- others are more sophisticated
  - text to TFID histogram
  - word2vec (maps words into vectors)
  - pre-trained ImageNet NN (maps images into vectors)

(more on these later)
More embeddings

- color to $\phi = (R, G, B)$

- geolocation data: $\phi(\mathbf{u}) = (\text{Lat}, \text{Long})$ in $\mathbb{R}^2$ or embed in $\mathbb{R}^3$ (if data points are spread over planet)

- day of week:
Faithful embeddings

a \textit{faithful} embedding satisfies

\begin{itemize}
  \item $\phi(u)$ is near $\phi(\tilde{u})$ when $u$ and $\tilde{u}$ are ‘similar’
  \item $\phi(u)$ is not near $\phi(\tilde{u})$ when $u$ and $\tilde{u}$ are ‘dissimilar’
\end{itemize}

\begin{itemize}
  \item lefthand concept is \textit{vector distance}
  \item righthand concept depends on field type, application
\end{itemize}

\begin{itemize}
  \item interesting examples: names, professions, companies, countries, languages, ZIP codes, cities, songs, movies
  \item we will see later how such embeddings can be constructed
\end{itemize}
Standardized embeddings

usually assume that an embedding is *standardized*

- entries of \( \phi(u) \) are centered around 0
- entries of \( \phi(u) \) have RMS value around 1
- roughly speaking, entries of \( \phi(u) \) ranges over \( \pm 1 \)

- with standarized embeddings, entries of feature map

\[
\phi(u_1, \ldots, u_r) = (\phi_1(u_1), \ldots, \phi_r(u_r))
\]

are all comparable, i.e., centered around zero, standard deviation around one

- \( \text{rms}(\phi(u) - \phi(\tilde{u})) \) is reasonable measure of how close records \( u \) and \( \tilde{u} \) are
Standardization or $z$-scoring

- suppose $\mathcal{U} = \mathbb{R}$ (field type is real numbers)
- for data set $u^1, \ldots, u^n \in \mathbb{R}$

\[
\bar{u} = \frac{1}{n} \sum_{i=1}^{n} u^i \quad \text{std}(u) = \left( \frac{1}{n} \sum_{i=1}^{n} (u^i - \bar{u})^2 \right)^{\frac{1}{2}}
\]

- the $z$-score or standardization of $u$ is the embedding

\[
x = \text{zscore}(u) = \frac{1}{\text{std}(u)} (u - \bar{u})
\]

- ensures that embedding values are centered at zero, with standard deviation one
- $z$-scored features are very easy to interpret: $x = \phi(u) = +1.3$ means that $u$ is 1.3 standard deviations above the mean value
Standardized data matrix

- suppose all $d$ (real) features have been standardized
- columns of $n \times d$ feature matrix $X$ have zero mean, RMS value one
- $(1/n)X^TX = \Sigma$ is the feature correlation matrix
- $\Sigma_{ii} = 1$ (since each column of $X$ has RMS value 1, and so norm $\sqrt{n}$)
- $\Sigma_{ij}$ is correlation coefficient of $i$th and $j$th raw features
Log transform

- old school rule-of-thumb: if field $u$ is positive and ranges over wide scale, embed as $\phi(u) = \log u$ (or $\log(1 + u)$) (and then standarize)

- examples: web site visits, ad views, company capitalization

- interpretation as faithful embedding:
  - 20 and 22 are similar, as are 1000 and 1100
  - but 20 and 120 are not similar
  - i.e., you care about fractional or relative differences between raw values

(here, log embedding is faithful, affine embedding is not)

- can also apply to output or label field, i.e., $y = \psi(v) = \log v$ if you care about percentage or fractional errors; recover $\hat{v} = \exp(\hat{y})$
Example: House price prediction

- we want to predict house selling price $v$ from record $u = (u_1, u_2)$
  - $u_1 = \text{area (sq. ft.)}$
  - $u_2 = \# \text{ bedrooms}$

- we care about relative error in price, so we embed $v$ as $\psi(v) = \log v$ (and then standardize)

- we standardize fields $u_1$ and $u_2$

\[
x_1 = \frac{u_1 - \mu_1}{\sigma_1}, \quad x_2 = \frac{u_2 - \mu_2}{\sigma_2}
\]

- $\mu_1 = \bar{u}_1$ is mean area
- $\mu_2 = \bar{u}_2$ is mean number of bedrooms
- $\sigma_1 = \text{std}(u_1)$ is std. dev. of area
- $\sigma_2 = \text{std}(u_2)$ is std. dec. of $\# \text{ bedrooms}$

(means and std. dev. are over our data set)
Example: House price regression model

- regression model: \( \hat{y} = \theta_1 + \theta_2 x_1 + \theta_3 x_2 \)
- in terms of original raw data:
  \[
  \hat{v} = \exp \left( \theta_1 + \theta_2 \frac{u_1 - \mu_1}{\sigma_1} + \theta_3 \frac{u_2 - \mu_2}{\sigma_2} \right)
  \]
- \( \exp \) undoes log embedding of house price
Vector embeddings for real field

- we can embed a field $u$ into a vector $x = \phi(u) \in \mathbb{R}^k$
- useful even when $\mathcal{U} = \mathbb{R}$ (real field)
- polynomial embedding:
  \[
  \phi(u) = (1, u, u^2, \ldots, u^d)
  \]
- piecewise linear embedding:
  \[
  \phi(u) = (1, (u)_-, (u)_+)
  \]
  where $(u)_- = \min(u, 0), (u)_+ = \max(u, 0)$
- regression with these features yield polynomial and piecewise linear predictors
Whitening

- analog of standardization for raw data $U = \mathbb{R}^k$
- start with raw data, $n \times d$ matrix $U$
- $\bar{u} = U^T \frac{1}{n}$ is vector of column means
- $\tilde{U} = U - 1\bar{u}^T$ is de-meaned data matrix
- $\tilde{U} = QR$ is its QR factorization
- $X = \sqrt{n}Q = \sqrt{n}\tilde{U}R^{-1}$ defines embedding $x^i = \phi(u^i)$
  - columns of $X$ have zero mean and RMS value one
  - columns of $X$ are orthogonal
  - features are uncorrelated
  - feature correlation matrix is $\Sigma = I$
Categorical data

- data field is *categorical* if it only takes a finite number of values

- *i.e.*, $\mathcal{U}$ is a finite set $\{\alpha_1, \ldots, \alpha_k\}$

- examples:
  - TRUE/FALSE (two values, also called Boolean)
  - APPLE, ORANGE, BANANA (three values)
  - MONDAY, ..., SUNDAY (seven values)
  - ZIP code (40000 values)

- *one-hot embedding for categoricals*: $\phi(\alpha_i) = e_i \in \mathbb{R}^k$

  - $\phi($APPLE$) = (1, 0, 0)$,  $\phi($ORANGE$) = (0, 1, 0)$,  $\phi($BANANA$) = (0, 0, 1)$

- standardizing these features handles *unbalanced* data
**Ordinal data**

- Ordinal data is categorical, with an order

- Example: *Likert scale*, with values
  
  STRONGLY DISAGREE, DISAGREE, NEUTRAL, AGREE, STRONGLY AGREE

- Can embed into \( \mathbb{R} \) with values \(-2, -1, 0, 1, 2\)

- Or treat as categorical, with one-hot embedding into \( \mathbb{R}^5 \)

- Example: number of bedrooms in house

  - Can be treated as a real number
  
  - Or as an ordinal with (say) values 1, \ldots, 6
Feature engineering
How feature maps are constructed

- start by embedding each field

\[ \phi(u_1, \ldots, u_r) = (\phi_1(u_1), \ldots, \phi_r(u_r)) \]

- can then standardize, if needed

- use *feature engineering* to create new features from existing ones
Creating new features

- product features: $x_{\text{new}} = x_i x_j$ (models interactions between features)
- max features: $x_{\text{new}} = \max(x_i, x_j)$ (can also use min)
- positive/negative parts:
  \[ x_{\text{new}}^+ = (x_i)^+ = \max(x_i, 0), \quad x_{\text{new}}^- = (x_i)^- = \min(x_i, 0) \]
- random features:
  - choose random matrix $R$
  - new features are $(Rx)^+$ or $(Rx)^-$
Un-embedding
Un-embedding

- we embed \( v \) as \( y = \psi(v) \), \( \psi : \mathcal{V} \rightarrow \mathbb{R} \)
- we need to ‘invert’ this operation, and go from \( \hat{y} \) to \( \hat{v} \)
- when the inverse function exists, we use \( \psi^{-1} : \mathbb{R} \rightarrow \mathcal{V} \)
- example: log embedding \( y = \log v \) has inverse \( v = \exp y \)
- prediction stack:
  1. **embed**: given record \( u \), feature vector is \( x = \phi(u) \)
  2. **predict**: \( \hat{y} = g(x) \)
  3. **un-embed**: \( \hat{v} = \psi^{-1}(\hat{y}) \)
- final predictor is \( \hat{v} = \psi^{-1}(g(\phi(u))) \)
Un-embedding

- in many cases, the inverse of $\psi$ function doesn’t exist
- for example, embedding a Boolean or ordinal into $\mathbb{R}$
- for the purposes of un-embedding, we define

$$\psi^{-1}(y) = \arg\min_{v \in \mathcal{V}} ||y - \psi(v)||$$

i.e., we choose the value of $v$ for which $\psi(v)$ is closest to $y$

- example: embed $\text{TRUE} \mapsto 1$ and $\text{FALSE} \mapsto -1$

- un-embed via

$$\psi^{-1}(y) = \begin{cases} \text{TRUE} & \text{if } y > 0 \\ \text{FALSE} & \text{otherwise} \end{cases}$$
Example: Un-embedding one-hot

- **one-hot embedding**: \( \phi(u) = e_u \) for \( U = \{1, \ldots, d\} \)

- un-embed

\[
\phi^{-1}(x) = \arg\min_u ||x - e_u||_2 = \arg\max_u x_u
\]