University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 6: Feature Engineering

Slides adapted from Chris Ketelsen, Jordan Boyd-Graber, and Noah Smith

Administrivia

- HW 2 will be released today
- Prelims for distance sections
- Final projects!!!
- HW1 feedback initial summary (14 responses)

Learning Objectives

- Understand why features matter
- Understand feature engineering techniques

Outline

- Features matter
- Feature engineering techniques

Outline

- Features matter
- Feature engineering techniques

Scary words

Outline for CSCI 5622
We've already covered stuff in blue!

- Problem formulations: classification, regression
- Supervised techniques: decision trees, nearest neighbors, perceptron, linear models, neural networks, support vector machine, kernel methods
- Unsupervised techniques: clustering, linear dimensionality reduction, topic modeling
- "Meta-techniques": ensembles, expectation-maximization, variational inference
- Understanding ML: limits of learning, practical issues, bias & fairness
- Recurring themes: (stochastic) gradient descent

Today: (More) Best Practices

You already know:

- Separating training and test data
- Hyperparameter tuning on development data

Understanding machine learning is partly about knowing algorithms and partly about the art of mapping abstract problems to learning tasks.

Features/representations

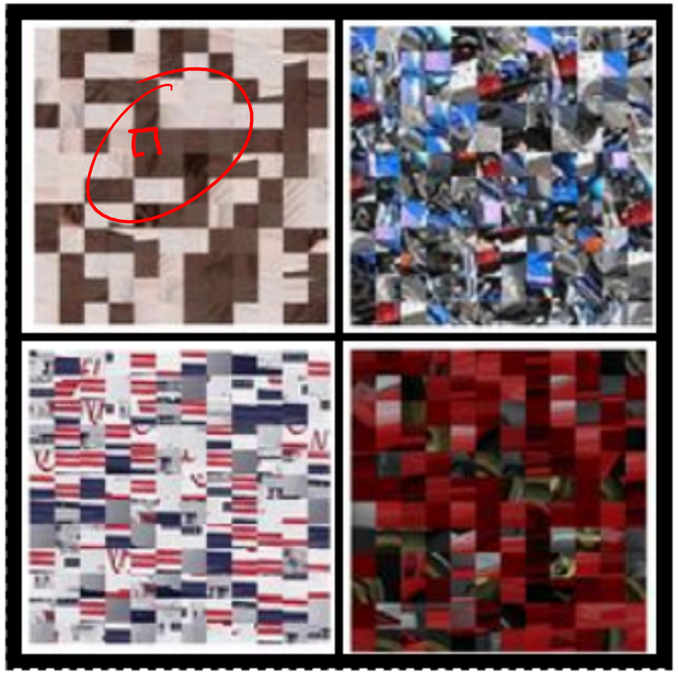
x

Features Matter

Pixel representation

Features Matter

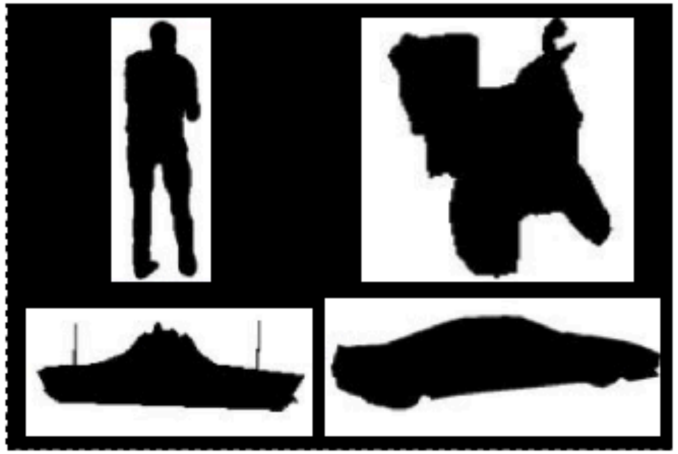
Patch representation



Face

Roses

Shape representation



Features Matter

Original

Features Matter

Bag of words

a, algorithms, and, applications, arthur, artificial, building, by, can, coined, computational, computer, computing, construction, data, decisions, designing, detection, difficult, driven, email, employed, evolved, example, explicit, explores, filtering, following, from, good, in, include, infeasible, inputs, instructions, intelligence, intruders, is, learn, learning, machine, make, making, model, name, network, of, on, or, overcome, pattern, performance, predictions, program, programming, range, recognition, sample, samuel, static, strictly, study, such, tasks, that, the, theory, through, vision, was, where, with

Original

The name machine learning was coined in 1959 by Arthur Samuel.[1] Evolved from the study of pattern recognition and computational learning theory in artificial intelligence,[3] machine learning explores the study and construction of algorithms that can learn from and make predictions on data[4] - such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions,[5]:2 through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders, and computer vision.

Outline

- Features matter
- Feature engineering techniques

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors?

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- K-nearest neighbors? ☹

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹
- Perceptron?

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- K-nearest neighbors? ☹
- Perceptron? ☺

Irrelevant Features

$$\mathrm{E}[f \mid Y] = \mathrm{E}[f]$$

One irrelevant feature isn't a big deal; what we're worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- K-nearest neighbors? ☹
- Perceptron? ☺

What about *redundant* features ϕ_j and $\phi_{j'}$ such that $\phi_j \approx \phi_{j'}$?

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Example: $\phi(x) = [\![$the word *the* occurs in document $x]\!]$

Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of D, remove it.

Example: $\phi(x) = [\![$the word *the* occurs in document $x]\!]$

Generalization: if a feature has variance (in D) **lower** than some threshhold value, remove it.

$$\text{sample\_mean}(\phi; D) = \frac{1}{N}\sum_{n=1}^{N}\phi(x_n) \qquad \text{(call it "}\bar{\phi}\text{")}$$

$$\text{sample\_variance}(\phi; D) = \frac{1}{N-1}\sum_{n=1}^{N}\left(\phi(x_n) - \bar{\phi}\right)^2 \qquad \text{(call it "Var}(\phi)\text{")}$$

Technique: Feature Normalization

Center a feature:

$$\phi(x) \to \phi(x) - \bar{\phi}$$

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

$$\phi(x) \to \frac{\phi(x)}{\sqrt{\mathrm{Var}(\phi)}} \qquad \text{"variance scaling"}$$

$$\phi(x) \to \frac{\phi(x)}{\max_n |\phi(x_n)|} \qquad \text{"absolute scaling"}$$

$\sim \mathcal{N}(0,1)$

$[-1, 1]$

Technique: Feature Normalization

Center a feature:

$$\phi(x) \rightarrow \phi(x) - \bar{\phi}$$

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

(Standard Scaler(Logistic Regression()))

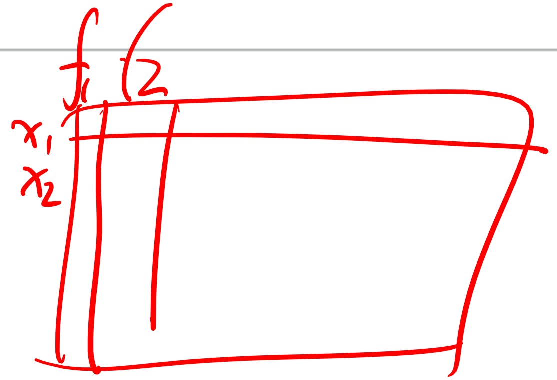
$$\phi(x) \rightarrow \frac{\phi(x)}{\sqrt{\mathrm{Var}(\phi)}} \quad \text{"variance scaling"}$$

$$\phi(x) \rightarrow \frac{\phi(x)}{\max_n |\phi(x_n)|} \quad \text{"absolute scaling"}$$

pipeline

Remember that you'll need to normalize test data before you test!

Technique: Example Normalization



We have been talking about normalizing columns.

We can also normalize rows. l_2 normalization is commonly used for bag of words.

$$\boldsymbol{x} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} = \frac{\boldsymbol{x}}{\sqrt{\sum_j \boldsymbol{x}[j]^2}}$$

Techniques: Creating New Features

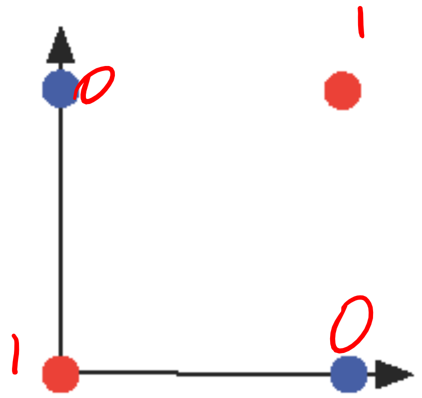
1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

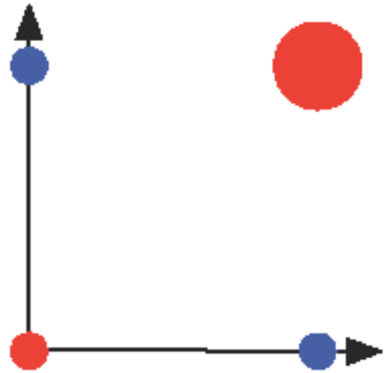


The classic "xor" problem: these points are *not* linearly separable.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

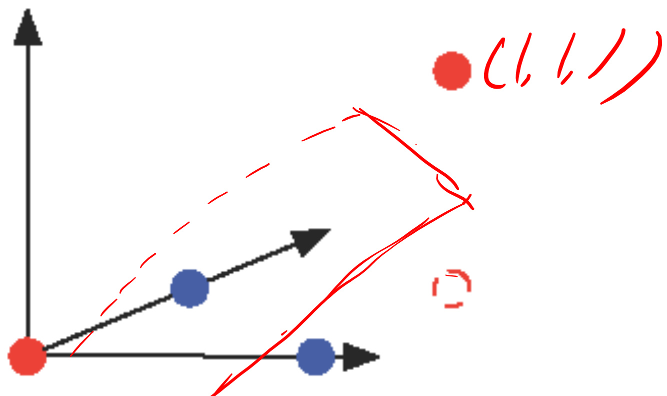


Define $\boldsymbol{x}[3] = \boldsymbol{x}[1] \wedge \boldsymbol{x}[2]$.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

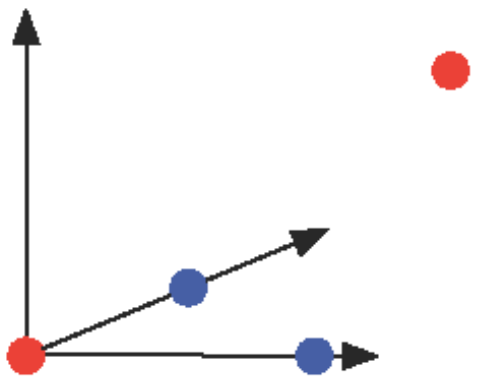


Rotating the view.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

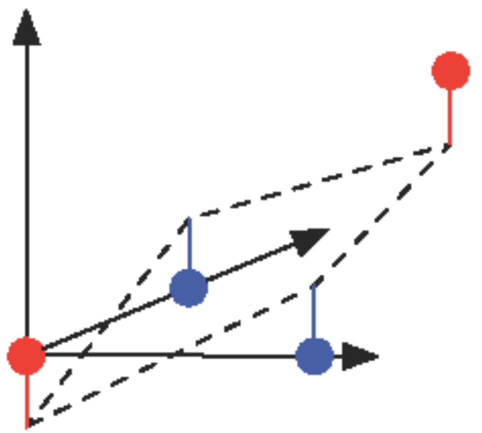
$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$



Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$



$$2 \cdot \boldsymbol{x}[1] + 2 \cdot \boldsymbol{x}[2] - 4 \cdot \boldsymbol{x}[3] - 1 = 0$$

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

Generalization: take the *product* of two features.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

   $$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.
2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.
   This is one view of what decision trees are doing!
   - Every leaf's path (from root) is a conjunction feature.
   - Why not build decision trees, extract the features and toss them into the perceptron?

Techniques: Creating New Features

1. Consider two binary features, ϕ_j and $\phi_{j'}$. A new *conjunction* feature can be defined by:

$$\phi_{j \wedge j'}(x) = \phi_j(x) \wedge \phi_{j'}(x)$$

   Generalization: take the *product* of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we'd like.
   This is one view of what decision trees are doing!
   - Every leaf's path (from root) is a conjunction feature.
   - Why not build decision trees, extract the features and toss them into the perceptron?

3. Transformations on features can be useful. For example,

$$\phi(x) \rightarrow \text{sign}(\phi(x)) \cdot \log(1 + |\phi(x)|)$$

Techniques: Creating New Features

Remember that adding features does not always bring benefits.

You could be just bring irrelevant, redundant, or features that make linearly separable datasets not linearly separable.

A more realistic but easy example

Given the following data about the locations of two cities, predict whether it is possible to drive between these two cities.

x_1 x_2 x_3 x_4 y

City 1 lat.	City 1 long.	City 2 lat.	City 2 long.	drivable
123.24	46.71	121.33	47.34	Yes
123.24	56.91	121.33	55.23	Yes
123.24	46.71	121.33	55.34	No
123.24	46.71	130.99	47.34	No

Feature engineering

Features represent the food of machine learning.

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

- Pruning
- Normalization
- Creating new features

Feature engineering

Features represent the food of machine learning.

Garbage in, garbage out.

- Pruning
- Normalization
- Creating new features

In practice, feature engineering requires a deep understanding of the problem.