

Introduction to Text Mining

Part VIII: Text Mining using Classification and Regression

Henning Wachsmuth

<https://cs.upb.de/css>

Classification and Regression: Learning Objectives

Concepts

- Understand how to prepare dataset for supervised learning.
- Get to know how to employ classification within text mining.
- Get to know how to employ regression within text mining.

Text analysis techniques

- Learn how to train a machine learning algorithm for text classification.
- Understand how to engineer features for a given text analysis task.
- Learn how to score texts with linear regression.

Covered text analyses

- Named entity recognition
- Sentiment polarity classification
- Sentiment scoring

Outline of the Course

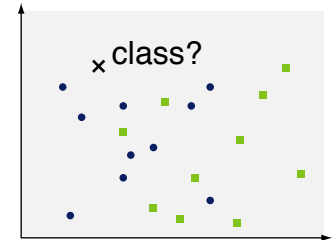
- I. Overview
- II. Basics of Linguistics
- III. Text Mining using Rules
- IV. Basics of Empirical Research
- V. Text Mining using Grammars
- VI. Basics of Machine Learning
- VII. Text Mining using Clustering
- VIII. Text Mining using Classification and Regression
 - What Is Text Mining using Classification and Regression?
 - Dataset Preparations
 - Supervised Classification
 - Supervised Regression
- IX. Practical Issues
- X. Text Mining using Sequence Labeling

What Is Text Mining
using Classification and Regression?

Classification and Regression

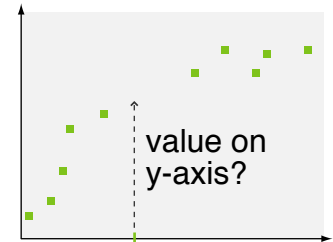
What is classification?

- The task to assign an instance to the most likely of a set of $k > 1$ predefined classes.
- Class values (aka labels) are interpreted as nominal.
This also holds if they in fact have an order or even a distance.



What is regression?

- The task to assign an instance to the most likely value from a real-valued scale.
- The values are continuous, possibly with upper and lower bounds.



Classification vs. regression

- Classification predicts group membership, regression a “missing” value.
- Many prediction tasks can be tackled with either of both.
- Via thresholding, we can map regression to classification problems.

Classification and Regression

Supervised Learning

Supervised (machine) learning

- Aims to find statistical patterns in annotated (ground-truth) data that connect input information to target output information.
- Derives a model $y : X \rightarrow C$ from training pairs of instances $\mathbf{x}^{(i)}$ and the associated output information $c(\mathbf{x}^{(i)})$.

Properties of supervised learning

- The learning process is guided by an external “teacher”.
- The optimization criterion relates to the target function in a given task or domain, i.e., the goal is to approximate the function best possible.

Classification and regression as supervised learning

- **Classification.** Can be done with hand-crafted rules (as seen earlier), but supervised learning is mostly the more successful way.
- **Regression.** In principle, real values can be calculated based on rules (or arithmetics), but real *regression* is inseparable from statistics.

Classification and Regression

Classification and Regression using Supervised Learning

Supervised classification (regression analog)

- Patterns in the representations of instances are learned based on their correlation with output classes.

Notice that machine learning does not “care” about the actual meaning of classes.

- The resulting models can assign arbitrary instances to output classes.

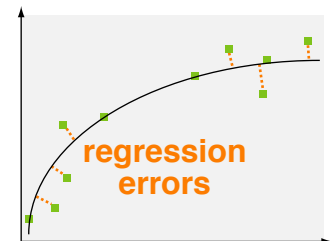
Evaluation of supervised classification

- The goal is to classify as many instances as possible correctly (from all or particular classes).
- **Measures.** Accuracy, precision, recall, F_1 -score.



Evaluation of supervised regression

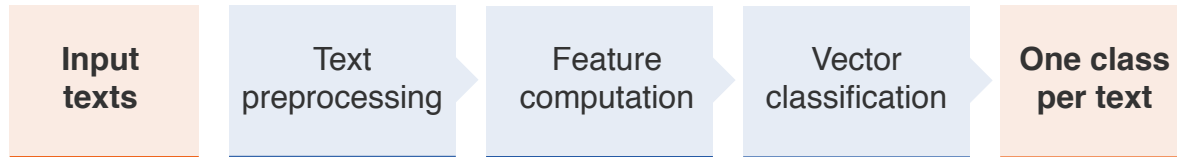
- The goal is to minimize the average difference between predicted and correct values.
- **Measures.** Mean absolute error, mean squared error.



Text Mining using Classification and Regression

Classification in text mining (regression analog)

- **Input.** Usually plain texts or text spans, in training with class information.
- **Output.** A class for each text + a model that maps from texts to classes.



Roles in text mining

- **Classification.** Decide about span boundaries, span types, text labels, relations between spans, relation types, ...
- **Regression.** Assign scores, rating, probabilities, ...

Applications in text mining

- **Classification.** Ubiquitous text analysis technique (see next slide).
- **Regression.** Scoring is the most common use case.

Text Mining using Classification and Regression

Text Analyses Overview from Lecture Part I

Lexical and syntactic

- Tokenization → Classification
- Sentence splitting → Classification
- Paragraph detection → Classification
- Stemming
- Lemmatization → Classification
- Part-of-speech tagging → Classification
- Similarity computation
- Spelling correction
- Phrase chunking → Classification
- Dependency parsing → Classification
- Constituency parsing → Classification
- ... and some more

Semantic and pragmatic

- Term extraction
- Numerical entity recognition → Classification
- Named entity recognition → Classification
- Reference resolution → Classification
- Entity relation extraction → Classification
- Temporal relation extraction → Classification
- Topic detection
- Authorship attribution → Classification
- Sentiment analysis → Classification
- Discourse parsing → Classification
- Spam detection → Classification
- Argument mining → Classification
- ... and many many more

Notice

- The label **Classification** denotes here that the standard approach to the respective analysis involves classification.

Text Mining using Classification and Regression

Dataset Preparation

Why dataset preparation?

- Text corpora usually contain annotations for the task to be studied.
- Not always, these annotations match with the task instances required for supervised classification and regression.

Creation of task instances

- Particularly, “negative” instances often need to be created for learning.

“[Jaguar]_{ORG} is named after the animal jaguar.”

- Also, annotations may have to be mapped to other task instances.

Ratings 1–2 → “negative”, 3 → ignore, 4–5 → “positive”

Balancing of datasets

- For learning, a balanced distribution of the target variable in the training set is often preferable.
- Different ways to oversample or undersample a dataset exist.

Dataset Preparations

Dataset Preparations

Text corpus (recap)

- A collection of real-world texts with known properties, compiled to study a language problem.
- The texts are often annotated for some target variable.



Characteristics of corpus annotations

- Often cover only the instances of a concept, not “negative” instances.
In entity corpora, it is only marked what is an entity.
- Define the maximum specificity level at which a problem can be studied.
If different organization types are marked, they can be abstracted into one class.
- Include values from a potentially application-specific scale.
[1, 4] is the typical range of essay grading in the US.
- Do not to always match across corpora for the same problem.
Hotel reviews with scores $\in [1, 5]$ vs. movie reviews with scores $\in [0, 2]$
- Are possibly distributed neither balanced nor representative.

Dataset Preparations

Negative Instances

Why “negative” instances?

- In many classification tasks, one class is in the focus.
- Other classes may not be annotated, or are more specific than needed.
False token boundaries, spans that are *not* entities, different neutral sentiments, ...

Defining negative instances

- What is seen as a “negative” instance is a design decision.
- The decision should be based on what a classifier should be used for.
- Trivial cases may distract classifiers from learning relevant differences.

Example: Negative instances in person entity recognition

“[tim]_{PER} works in [cupertino]_{LOC}. [san fran]_{LOC} is his home. as a cook, he cooks all day.”

- All other named entities? → Can only distinguish entity *types* then.
- All other content words? → Verbs will never be person names (somewhat trivial).
- All other noun phrases? → Seems reasonable.

Dataset Preparations

Mapping of Target Variable Values

What is the mapping of target variable values?

- The alteration of the target classes or values in a given corpus/dataset.
- May require expert knowledge about the target variable.

Why mapping?

- Corpus annotations may be more fine-grained than needed.

Sentiment scores: $[1, 2] \rightarrow \text{“negative”}$, $]2, 4[\rightarrow \text{“neutral”}$, $[4, 5] \rightarrow \text{“positive”}$

- Some values or differences may not be relevant in a given application.

Polarities: $\text{“negative”} \rightarrow \text{“negative”}$, ignore “neutral” , $\text{“positive”} \rightarrow \text{“positive”}$

- The ranges of the “same” target variable may not match across corpora.

Sentiment scores: $\{1, 2\} \rightarrow 0$, $\{3\} \rightarrow 1$, $\{4, 5\} \rightarrow 2$

- The values may just not be those desired (purely “aesthetical”).

Polarities: $\text{“bad”} \rightarrow \text{“negative”}$, $\text{“medium”} \rightarrow \text{“neutral”}$ $\text{“good”} \rightarrow \text{“positive”}$

Dataset Preparations

Balancing Datasets

What is dataset balancing?

- The alteration of the distribution of a dataset with respect to some target variable, such that the distribution is uniform afterwards.
- Balancing is more common for classification than for regression, since in regression there is often no fixed set of target values to balance.

A solution for regression is *binning*, i.e., to balance the distribution for certain intervals.

When to balance?

- Balancing a training set prevents machine learning from being biased towards majority classes (or values).
- Validation and test sets should usually not be balanced, since analysis algorithms are usually evaluated on *representative* distributions.

How to balance?

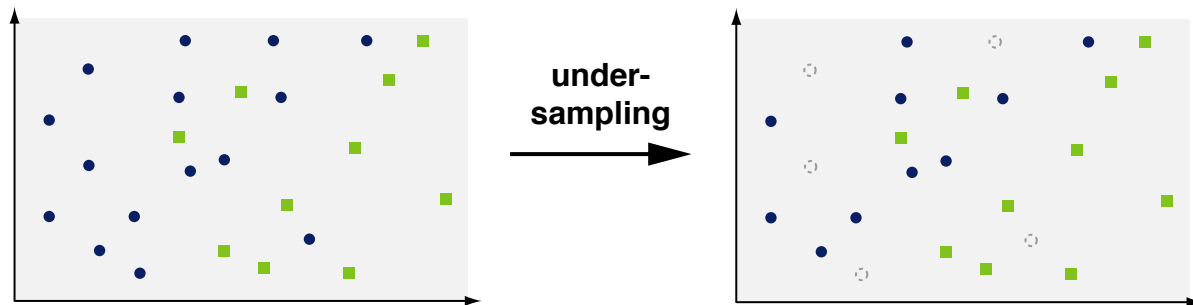
- **Undersampling.** Removal of instances from majority classes.
- **Oversampling.** Addition of instances from minority classes.

Dataset Preparations

Undersampling

How to balance with undersampling?

- Removing instances of all non-minority classes until all classes have the size of the minority class.
- Instances to be removed are usually chosen randomly.



Pros and cons

- **Pro.** All remaining data is real.
- **Pro.** Downsizing of a dataset makes training less time-intensive.
- **Con.** Instances that may be helpful in learning are discarded, i.e., potentially relevant information is lost.

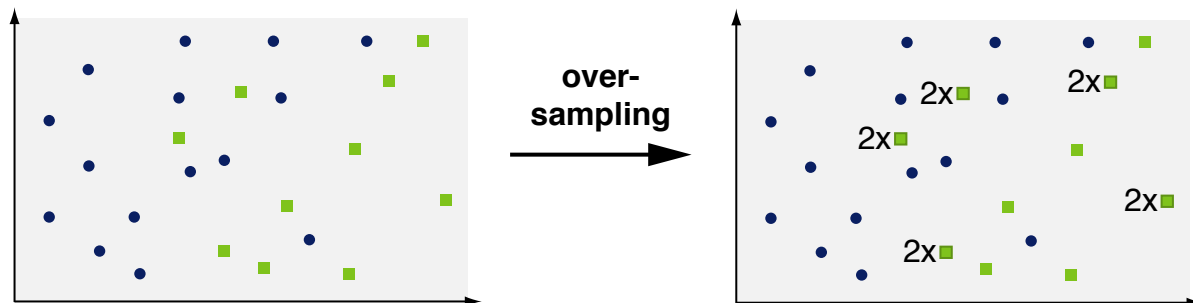
Dataset Preparations

Oversampling

How to balance with oversampling?

- Adding instances of all minority classes until all classes have the size of the majority class.
- Usually, the instances to be added are random duplicates.

Where reasonable, an alternative is to create artificial instances using interpolation.



Pros and cons

- **Pro.** No instance is discarded, i.e., all information is preserved.
- **Con.** Upsizing of a dataset makes training more time-intensive.
- **Con.** The importance of certain instances is artificially boosted, which may make features discriminative that are actually noise.

Dataset Preparations

Undersampling vs. Oversampling

Example: A sentiment training set

- 1000 positive, 500 negative, 100 neutral instances.
- After undersampling? → 100+100+100 instances
- After oversampling? → 1000+1000+1000 instances



What to use?

- When more than enough data is available, undersampling is preferable.
- When the class imbalance is low, oversampling is rather unproblematic.
- When the class imbalance is high, no really good choice exists.

Alternatives to balancing?

- Many machine learning optimization procedures can penalize wrong predictions of minority instances more than majority instances.
- Conceptually, this is the more sound way of preventing the bias.
- Practically, it makes the learning process more complex, which is why balancing is often used.

Supervised Classification

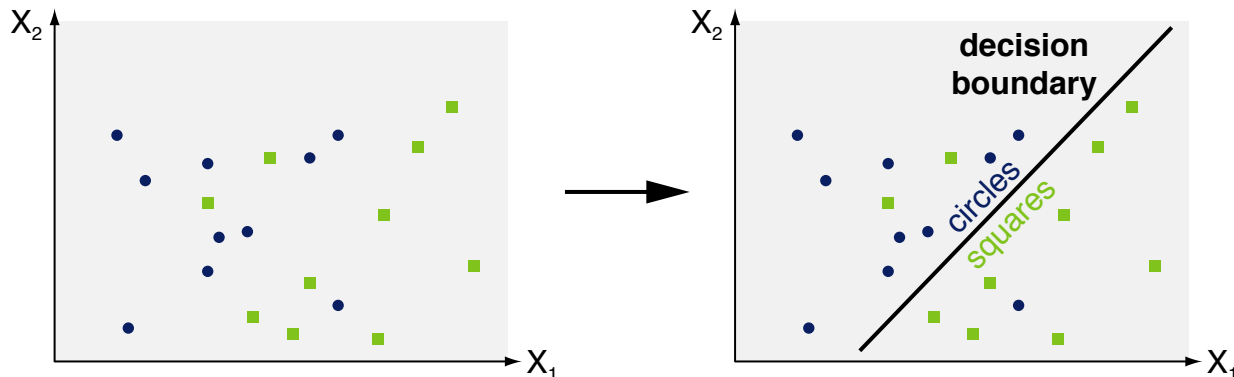
Supervised Classification

What is supervised classification?

- Supervised classification learns to predict for an instance the most likely of a set of $k > 1$ predefined discrete classes.

Learning phase (training)

- **Input.** A set of known instances $\mathbf{x}^{(i)}$ with correct output class $c(\mathbf{x}^{(i)})$.
- **Output.** A model $X \rightarrow C$ that maps any instance to its output class.



Prediction phase

- **Input.** A set of unknown instances $\mathbf{x}^{(i)}$ without output classes.
- **Output.** The output class $c(\mathbf{x}^{(i)})$ for each instance.

Supervised Classification

Feature-based Classification

Feature-based representation (recap)

- A feature vector is an ordered set of values of the form $\mathbf{x} = (x_1, \dots, x_m)$, where each feature x_i denotes a measurable property of an input, $m \geq 1$.

We consider only real-valued features here.

- Each instance is mapped to a vector $\mathbf{x}^{(j)} = (x_1^{(j)}, \dots, x_m^{(j)})$ where $x_i^{(j)}$ denotes the value of feature x_i .

We consider only values normalized to the range $[0, 1]$ here.

Text mining using feature-based classification

- The main challenge is to engineer features that help solve a given task.
- In addition, a suitable classification algorithm needs to be chosen.

Feature-based vs. neural classification (recap)

- We restrict our view to feature-based classification here.

Same for regression below.

- In neural classification, features are not explicitly represented anymore.

Supervised Classification

Classification Algorithms

Binary vs. multiple-class classification (recap)

- **Binary.** Many classification algorithms work for $k = 2$ classes only.
- **Multiple.** Handled via multiple binary classifiers, e.g., one-versus-all.

Selected supervised classification algorithms

- **Naïve Bayes.** Predicts classes based on conditional probabilities.
- **Support vector machine.** Maximizes the margin between classes.
- **Decision tree.** Sequentially compares instances on single features.
- **Random forest.** Majority voting based on several decision trees.
- **Neural network.** Learns complex function on feature combinations.

... and many more

Focus on support vector machines here

- One of the most widely used feature-based classification algorithms.
- Often, a good default choice. Much theoretical and empirical appeal.

Support Vector Machines

What is a support vector machine (SVM)?

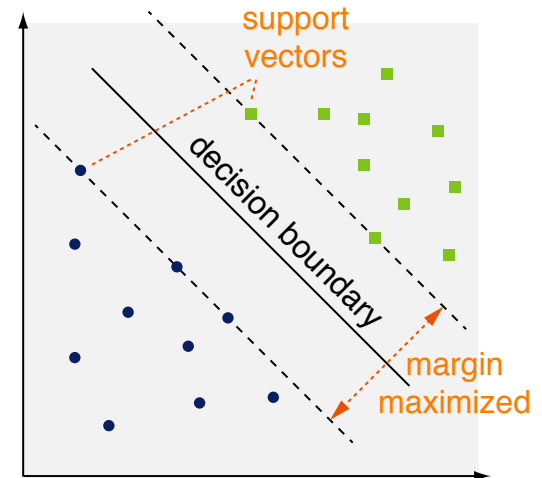
- A supervised learning method that aims to find the decision boundary which maximizes the margin between instances of two classes.
- The decision boundary is linear. Non-linear classification is possible through the *kernel trick* (see below).

Support vectors

- A (usually small) subset of training instances that fully specifies the decision boundary.
- Other instances are disregarded by an SVM.

Large-margin classification

- The distance from the boundary to all closest instances defines the margin of a classifier.
 - SVMs maximize the minimum margin of the decision boundary to the training instances.
- Some instances may be discounted as outliers/noise.



Support Vector Machines

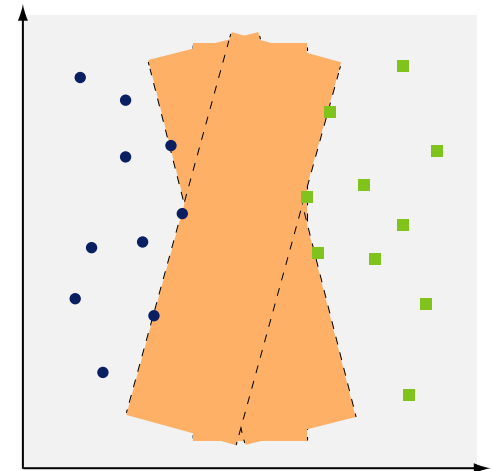
Intuition

Maximization of certainty

- Intuitively, a decision boundary in the middle between instances of two classes is better than one which is very close to instances of either.
- A large margin avoids low-certainty decisions in training, i.e., decisions for instances near the decision boundary.
- A slight instance variation will not cause a misclassification.

Better generalization

- SVMs aim to make the “separator” between classes as large as possible.
- Fewer positions for such a separator exist. This reduces memory capacity, i.e., variance.
- Hence, its ability to correctly generalize to test data is usually increased.



Support Vector Machines

SVM Classification

Decision boundary

- A hyperplane defined by a weight vector \mathbf{w} , which is perpendicular to the hyperplane (aka the *normal*), along with an intercept term b .
- All points \mathbf{x} on the hyperplane satisfy $\mathbf{w}^T \mathbf{x} = b$.

Training

- Given a set of n training pairs $(\mathbf{x}^{(i)}, y^{(i)})$, where $\mathbf{x}^{(i)}$ is an instance and $y^{(i)} = c(\mathbf{x}^{(i)}) \in \{-1, 1\}$ its class label.
All feature values are assumed to be normalized to $[0, 1]$.
- The training set uniquely defines the best separating hyperplane.

Linear SVM classifier

- The linear classifier is then: $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} - b)$
- A value of -1 indicates one class, and a value of $+1$ the other class.

Support Vector Machines

Linearly Separable Cases

Maximum-margin decision boundary

- If the training set is linearly separable, any instance $\mathbf{x}^{(i)}$ must fulfill either of:

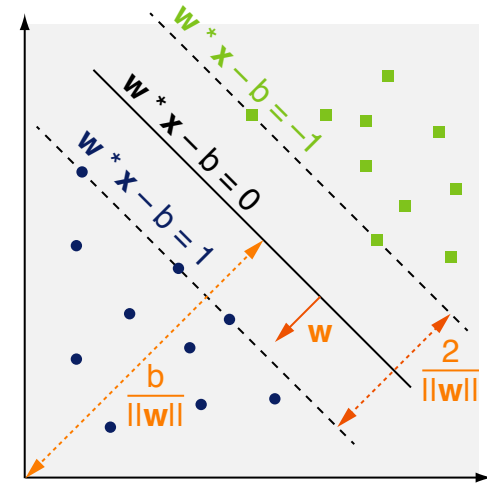
$$\mathbf{w}^T \mathbf{x}^{(i)} - b \geq 1 \quad \text{if } y^{(i)} = 1$$

$$\mathbf{w}^T \mathbf{x}^{(i)} - b \leq -1 \quad \text{if } y^{(i)} = -1$$

- This can be rewritten as:

$$\forall i \in \{1, \dots, n\} : y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} - b) \geq 1$$

- The margin is given by $\frac{2}{\|\mathbf{w}\|}$, as follows from the inequations above.
- To maximize the margin, $\|\mathbf{w}\|$ must be minimized.



Hard-margin SVM as a minimization problem

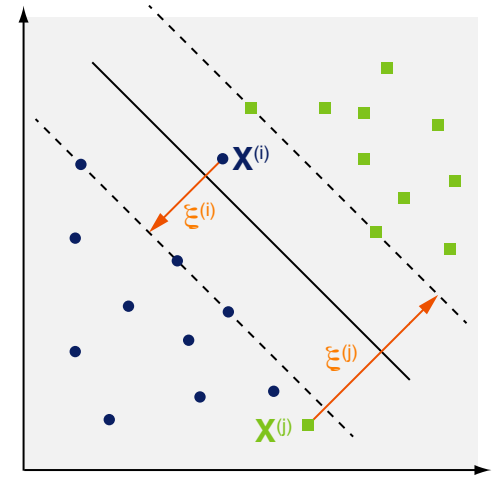
- Find \mathbf{w} and b such that $\|\mathbf{w}\|$ is minimized, and $\forall i : y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} - b) \geq 1$.
- A quadratic optimization procedure based on a training set finds the support vectors that represent the hyperplane.

Support Vector Machines

Linearly Inseparable Cases

Slack variables

- For linearly inseparable training sets, an SVM can allow instances $\mathbf{x}^{(i)}$ to be misclassified via slack variables $\xi^{(i)} \geq 0$.
- If $\xi^{(i)} > 0$, the margin of $\mathbf{x}^{(i)}$ can be less than 1 at a cost proportional to $\xi^{(i)}$.
- The SVM then trades the size of the margin against the number of correctly classified training instances.



Soft-margin SVM as a minimization problem

- Find \mathbf{w} , b , and $\xi^{(i)} \geq 0$ such that $\|\mathbf{w}\| + C \cdot \sum_i \xi^{(i)}$ is minimized, and $\forall i : y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} - b) \geq 1 - \xi^{(i)}$.
- $C > 0$ is a regularization term, used to control overfitting. It must be optimized against a validation set.

Support Vector Machines

Cost Hyperparameter Optimization

The cost hyperparameter C

- If C is small, the bulk of instances can be modeled easily at low cost.
- As C becomes large, it is unattractive to not respect training instances.
- So, the higher C , the more an SVM will fit the training data.

Typical cost optimization process

- First find best magnitude of C on the validation set.
For instance, by testing each $C \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.
- Then validate more fine-grained cost values in this magnitude.
- The best found value is used on the test set (or in the application).

Notice

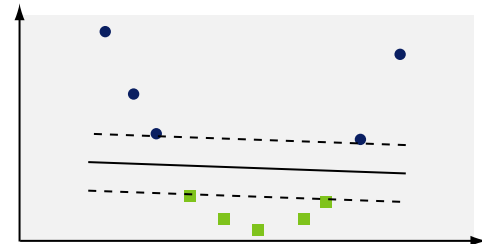
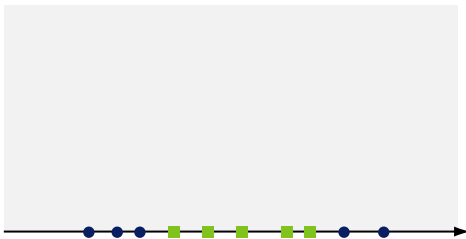
- Non-linear SVMs (see below) may have a second hyperparameter γ .
- The combination C and γ needs to be optimized, e.g., in a grid search.

Support Vector Machines

Non-Linear Classification

Non-linear SVMs

- Some data is just not linearly separable.
- Non-linear SVMs map the original feature space of such data to some higher-dimensional space where it is linearly separable.
- A linear classifier is then learned for the higher-dimensional space.



Kernel trick

- Conceptually, the mapping is a non-linear transformation $\Phi : \mathbf{x} \mapsto \phi(\mathbf{x})$.
- Practically, SVMs use a *kernel function* K that computes the dot product of two transformed instances in the original feature space.

Support Vector Machines

Kernel Functions

Linear SVMs with a kernel function

- A linear SVM can be specified as a sum of dot products

$K(\mathbf{x}, \mathbf{x}^{(j)}) = \mathbf{x}^T \mathbf{x}^{(j)}$ for an instance \mathbf{x} and each support vector $\mathbf{x}^{(j)}$:

$$f(\mathbf{x}) = \text{sign}(\sum_j y^{(j)} K(\mathbf{x}, \mathbf{x}^{(j)}) - b)$$

Kernel functions of non-linear SVMs

- K above can be replaced by a kernel function that computes the dot product of two transformed instances, $\phi(\mathbf{x})$ and $\phi(\mathbf{x}^{(j)})$.

For SVM optimization, K must fulfill certain properties omitted here for simplicity.

Common kernel functions

- **Polynomial kernels.** $K(\mathbf{x}, \mathbf{x}^{(j)}) = (1 + \mathbf{x}^T \mathbf{x}^{(j)})^d$.
A quadratic kernel ($d = 2$) is often used. $d = 1$ results in a linear kernel.
- **Radial basis function.** $K(\mathbf{x}, \mathbf{x}^{(j)}) = e^{-\gamma \cdot (\mathbf{x} - \mathbf{x}^{(j)})^2}$.
A Gaussian distribution that maps to a potentially infinite feature space.

Other functions can be designed for specific data, but this beyond the scope here.

Support Vector Machines

Empirical Results

Experiments with text classification (Joachims, 2002)

- **Corpus.** Reuters-21578, 90 topics, 9603 training and 3299 test texts.
- **Features.** More than 10,000 word-based features.
- **Classification.** Four baseline algorithms and different SVM variations.
- **Optimization.** Hyperparameters (incl. #features) tuned for all algorithms.

Effectiveness results (F₁-score)

Learning Algorithm	10 Most Frequent Topics										Micro
	earn	acq	mny-fx	grain	crude	trade	intrst.	ship	wheat	corn	F ₁
Naïve Bayes	96.0	90.7	59.6	69.8	81.2	52.2	57.6	80.9	63.4	45.2	72.3
Rocchio	96.1	92.1	67.6	79.5	81.5	77.4	72.5	83.1	79.4	62.2	79.9
Decision trees	96.1	85.3	69.4	89.1	75.5	59.2	49.1	80.9	85.5	87.7	79.4
<i>k</i> nearest neighbors	97.8	91.8	75.4	82.6	85.8	77.9	76.7	79.8	72.9	71.4	82.6
Linear SVM (<i>C</i> 0.5)	98.0	95.5	78.8	91.9	89.4	79.2	75.6	87.4	86.6	87.5	86.7
Linear SVM (<i>C</i> 1.0)	98.2	95.6	78.5	93.1	89.4	79.2	74.8	86.5	86.8	87.8	87.5
RBF-SVM	98.1	94.7	74.3	93.4	88.7	76.6	69.1	85.8	82.4	84.6	86.4

Support Vector Machines

Discussion

Benefits of SVMs

- Effective in high-dimensional feature spaces.
- Comparably effective for small numbers n of training instances.
- May still be effective if number m of features is greater than n .
- Memory-efficient, since it uses only a subset of the training instances.
- Different kernel functions can be specified.

Challenges with SVMs

- If m is much greater than n , SVMs may overfit.
Simple kernel functions and enough regularization are crucial then.
- SVMs do not directly provide probability estimates.
Estimates can be calculated internally using an expensive cross-validation.
- Higher-dimensional spaces increase the generalization error of SVMs.
Enough training data should be given, when using non-linear SVMs.
- Hyperparameter optimization is a must.

Review Sentiment Analysis

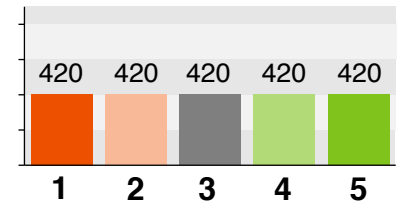
Sentiment classification of reviews

- Classification of the nominal sentiment polarity or score of a customer review on a product, service, or work of art.

Data

- 2100 English hotel reviews from TripAdvisor.
900 training, 600 validation, and 600 test reviews.
- Each review has a sentiment score from $\{1, \dots, 5\}$.

ArguAnaTripAdvisor corpus



Tasks

- **3-class sentiment.** 1–2 mapped to negative, 3 to neutral, 4–5 to positive.
Training set balanced with random undersampling.
- **5-class sentiment.** Each score interpreted as one (nominal) class.

Approach

- **Algorithm.** Linear SVM with one-versus-all multi-class handling.
Cost hyperparameter tuned on validation sets.
- **Features.** Combination of several standard and specific feature types.

Review Sentiment Analysis

Feature Engineering

What is feature engineering? (recap)

- The design and development of the feature representation of instances used to address a given task.
- Key step in feature-based machine learning.
- The representation governs what patterns can be found during learning.

Standard vs. specific features

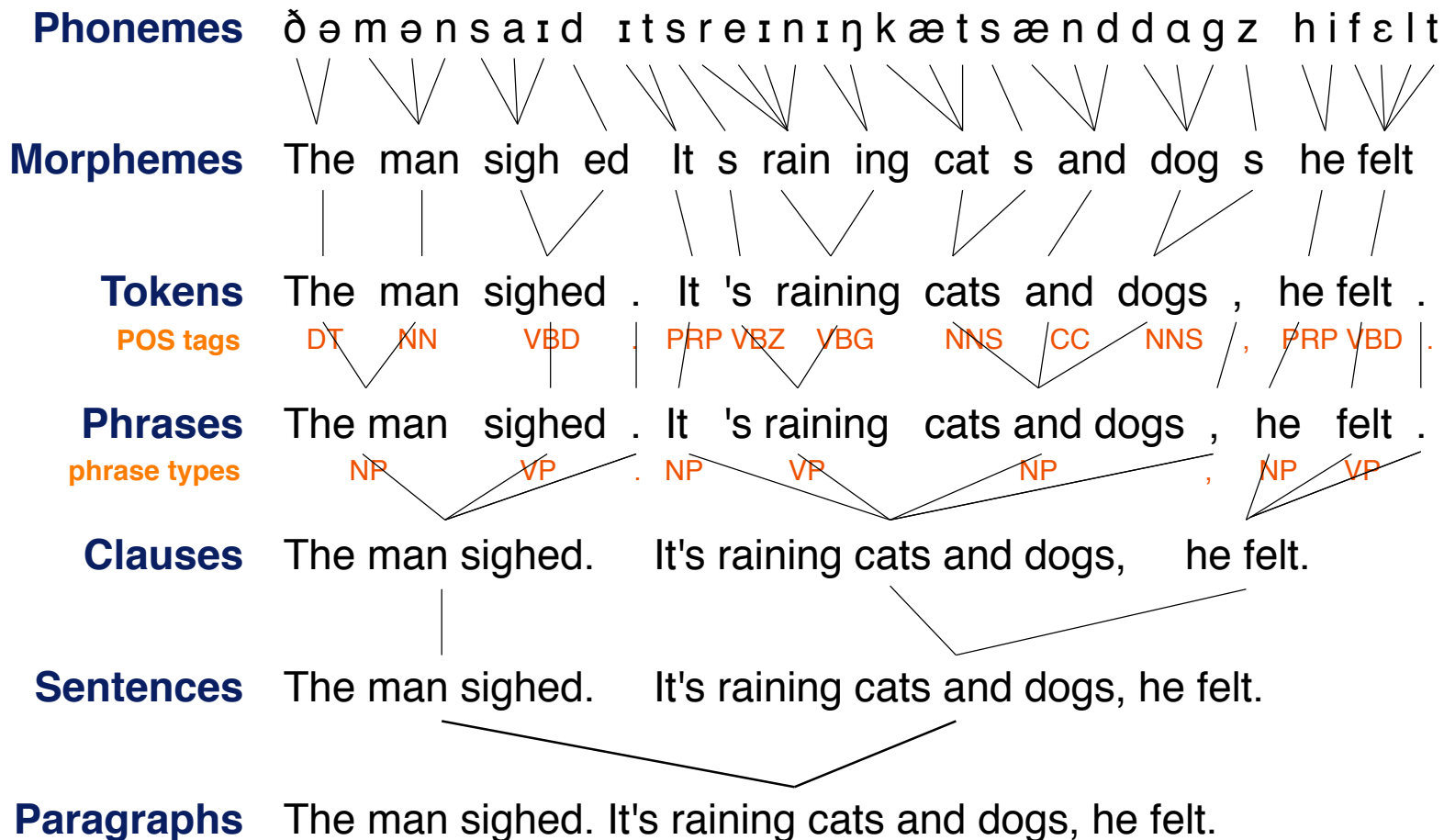
- **Standard.** Features that can be derived from (more or less) general linguistic phenomena and that may help in several tasks.
- **Specific.** Features that are engineered for a specific tasks, usually based on expert knowledge about the task.

Features covered here

- **Standard content features.** Token n -grams, target class features.
- **Standard style features.** POS and phrase n -grams, stylometric features.
- **Specific features.** Local sentiment, discourse relations, flow patterns.

Review Sentiment Analysis

Some General Linguistic Phenomena (Recap)



Review Sentiment Analysis

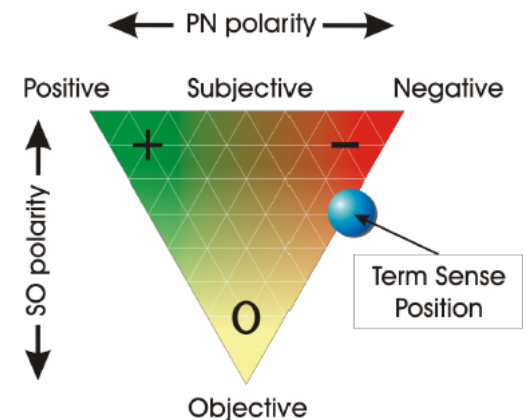
Standard Content Feature Types

Token n -grams

- **Token unigrams.** The distribution of all token 1-grams that occur in at 5% of all training texts.
- **Token bigrams.** Analog for token 2-grams.
- **Token trigrams.** Analog for token 3-grams.

Target class features

- **Core vocabulary.** The distribution of all words that occur at least three times as often in one class as in every other.
- **Sentiment scores.** The mean positivity, negativity, and objectivity of all first and average word senses in *SentiWordNet*. SentiWordNet is a lexicon with subjectivity and polarity values for words (Baccianella et al., 2010).
- **Sentiment words.** The distribution of all subjective words in SentiWordNet.



Review Sentiment Analysis

Standard Style Feature Types

Part-of-speech (POS) tag n-grams

- **POS unigrams.** The distribution of all part-of-speech 1-grams that occur in at least 5% of all training texts.
- **POS bigrams/trigrams.** Analog for part-of-speech 2-grams and 3-grams.

Phrase type n-grams

- **Phrase unigrams.** The distribution of all phrase type 1-grams that occur in at least 5% of all training texts.
- **Phrase bigrams/trigrams.** Analog for phrase type 2-grams and 3-grams.

Stylometric features

- **Character trigrams.** The distribution of all character 3-grams that occur in at least 5% of all training texts.
- **Function words.** The distribution of the top 100 words in the training set.
- **Lexical statistics.** Average numbers of tokens, clauses, and sentences.

Review Sentiment Analysis

Evaluation of the Standard Feature Types

Evaluation

- One linear SVM for each feature type in isolation and for their combination.
- Training on training set, tuning on validation set, application on test set.

Discussion

- Token unigrams best, but some other types close.
- Combination does not outperform single types.
- 60.8% accuracy does not seem satisfying.

Effectiveness results (accuracy)

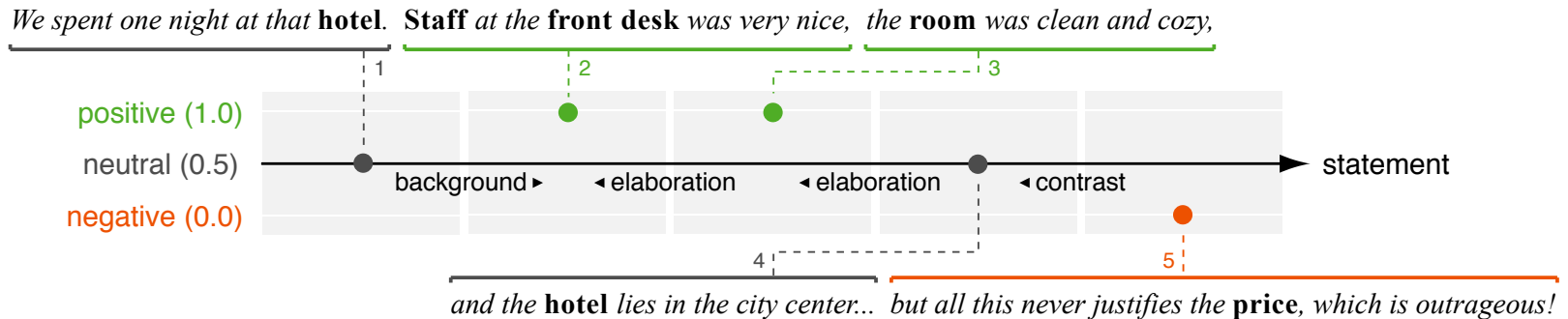
Category	Feature type	# Features	3 classes
Content	Token unigrams	426	60.8%
	Token bigrams	112	49.5%
	Token trigrams	64	24.5%
	Core vocabulary	83	41.7%
	Sentiment scores	6	59.3%
	Sentiment words	123	60.5%
	Style	POS unigrams	48
POS bigrams		70	49.0%
POS trigrams		118	45.5%
Phrase unigrams		13	48.8%
Phrase bigrams		43	52.5%
Phrase trigrams		122	50.8%
Function words		100	57.3%
Character trigrams		200	48.7%
Lexical statistics		6	42.8%
Combination of features		1534	60.8%

Review Sentiment Analysis

Review Argumentation

Example hotel review

“We spent one night at that hotel. Staff at the front desk was very nice, the room was clean and cozy, and the hotel lies in the city center... but all this never justifies the price, which is outrageous!”



A shallow model of review argumentation

- A review can be seen as a flow of local sentiments on domain concepts that are connected by discourse relations.

The domain concepts (aka aspects) are omitted in the following.

Review Sentiment Analysis

Specific Feature Types for Review Sentiment Analysis (1 of 2)

Local sentiment distribution

- The frequencies of positive, neutral, and negative local sentiment as well as of changes of local sentiments.
positive 0.4 neutral 0.4 negative 0.2 (neutral, positive) 0.25 ...
- The average local sentiment value from 0.0 (negative) to 1.0 (positive).
average sentiment 0.6
- The interpolated local sentiment at each normalized position in the text.
e.g., normalization length 9: (0.5, 0.75, 1.0, 1.0, 1.0, 0.75, 0.5, 0.25, 0.0)

Discourse relation distribution

- The distribution of discourse relation types in the text.
background 0.25 elaboration 0.5 contrast 0.25 (all others 0.0)
- The distribution of combinations of relation types and local sentiments.
background(neutral, positive) 0.25 elaboration(positive, positive) 0.25 ...

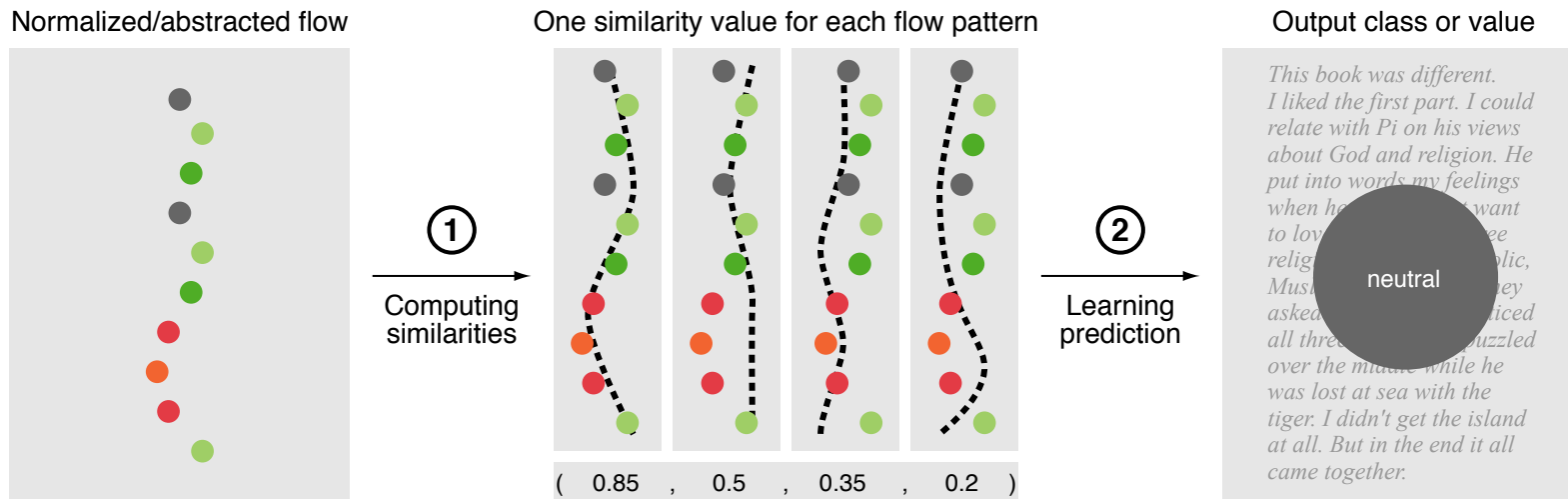
Review Sentiment Analysis

Specific Feature Types for Review Sentiment Analysis (2 of 2)

Sentiment flow patterns

- The similarity of the normalized flow of the text to each flow pattern.

Details on the patterns in the lecture part on clustering.



Content and style features

- Content.** Token n -grams, sentiment scores.
- Style.** Part-of-speech n -grams, character trigrams, lexical statistics.

Subset of the standard features above, here as baseline features.

Review Sentiment Analysis

Evaluation of the Specific Feature Types

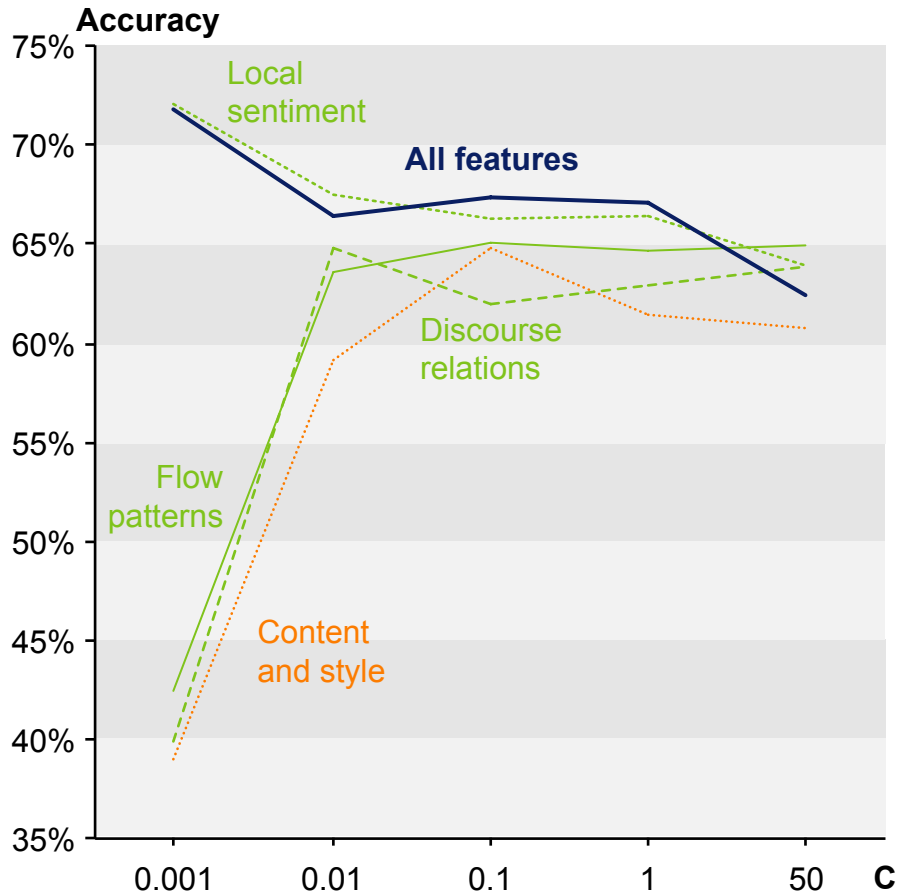
Evaluation

- One linear SVM for each feature type in isolation and for their combination.
- Training on training set, tuning on validation set, application on test set.
- Both 3-class and 5-class.

Cost hyperparameter tuning

- C values tested for SVM. 0.001, 0.01, 0.1, 1.0, 50.0
- Best C used on test set.
- Results shown here for the 3-class task only.

Validation accuracy depending on C



Review Sentiment Analysis

Results and Discussion for the Specific Features

Effectiveness results on test set (accuracy)

Feature type	# Features	3 Classes	5 Classes
Local sentiment distribution	50	69.8%	42.2%
Discourse relation distribution	75	65.3%	40.6%
Sentiment flow patterns	42	63.1%	39.7%
Content and style features	1026	58.9%	43.2%
Combination of features	1193	71.5%	48.1%
Random baseline		33.3%	20.0%

Discussion

- **Content and style features.** A bit weaker than in the experiment above, due to slight differences in the experiment setting.
- **Sentiment flow patterns.** Impact is more visible across domains (later).
- **Combination of features.** Works out this time, so more complementary.
- The 5-class accuracy seems insufficient.
- Classification misses to model the ordinal relation between classes; regression might be better.

Supervised Regression

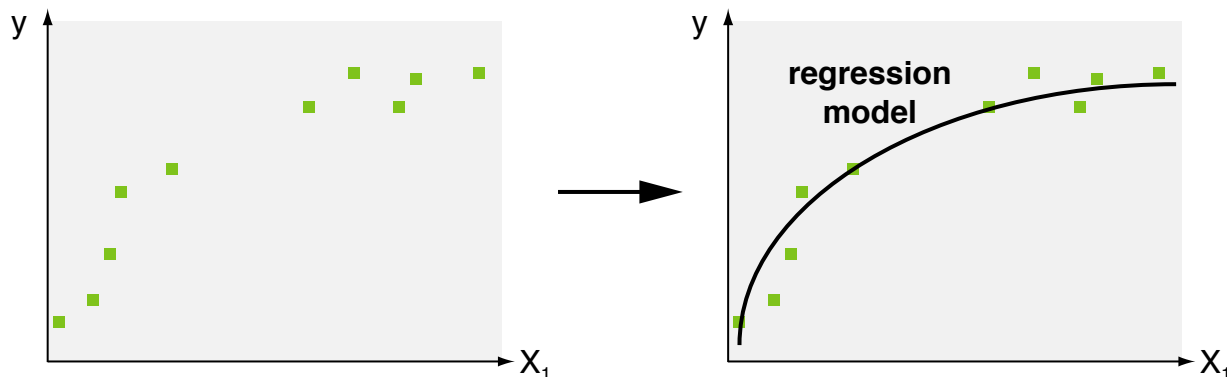
Supervised Regression

What is supervised regression?

- Supervised regression learns to predict for an instance the most likely value from a real-valued scale.

Learning phase (training)

- **Input.** A set of known instances $\mathbf{x}^{(i)}$ with correct output value $y = c(\mathbf{x}^{(i)})$.
- **Output.** A model $X \rightarrow C$ that maps any instance to its output value.



Prediction phase

- **Input.** A set of unknown instances $\mathbf{x}^{(i)}$ without output value.
- **Output.** The output value $y = c(\mathbf{x}^{(i)})$ for each instance.

Supervised Regression

Regression Algorithms

Selected supervised regression algorithms

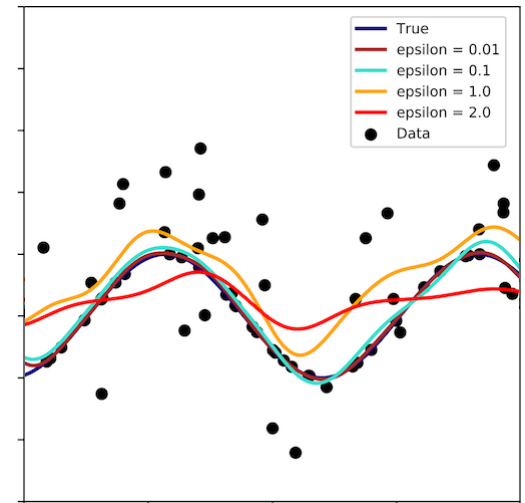
- **Linear regression.** Predict output values using a learned linear function.
- **Support vector regression.** Maximize the flatness of a regression model.

Focus on linear regression here

- Shows the general idea of regression more clearly.
- Despite its simplicity, often effective and well-interpretable.

Support vector regression

- The ideas of support vectors and kernels apply to regression, too.
- In a “tube” of some size ϵ around the model, regression errors are ignored.
- Instead of the margin in classification, the flatness of a regression model is maximized to reduce overfitting.



Linear Regression

What is linear regression?

- A supervised learning method that learns to predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = \theta_0 + \sum_{j=1}^m \theta_j \cdot x_j = \theta_0 + \theta_1 \cdot x_1 + \dots + \theta_m \cdot x_m$$

- The weight vector $\mathbf{w} = (\theta_0, \theta_1, \dots, \theta_m)$ is learned from training instances.

Training of a linear regression model

- Given a set of n training pairs $(\mathbf{x}^{(i)}, y^{(i)})$, where $\mathbf{x}^{(i)}$ is an instance and $y^{(i)} = c(\mathbf{x}^{(i)})$ its correct output value.
- **Goodness of fit.** The goal is to minimize the regression error of a model.
- **Residuals.** The regression error is assessed based on the differences between observed values $y^{(i)}$ and predicted values $y(\mathbf{x}^{(i)})$.
- **Cost function.** Usually, the regression error is quantified as the *residual sum of squares*.

Linear Regression

Optimization

Residual sum of squares (RSS)

- The sum of squared differences between predicted output values $y(\mathbf{x}^{(i)})$ and correct output values $y^{(i)}$ over all instances $\mathbf{x}^{(i)}$.

$$RSS(\mathbf{w}) = \sum_{i=1}^n (y^{(i)} - y(\mathbf{x}^{(i)}))^2 = \sum_{i=1}^n (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$$

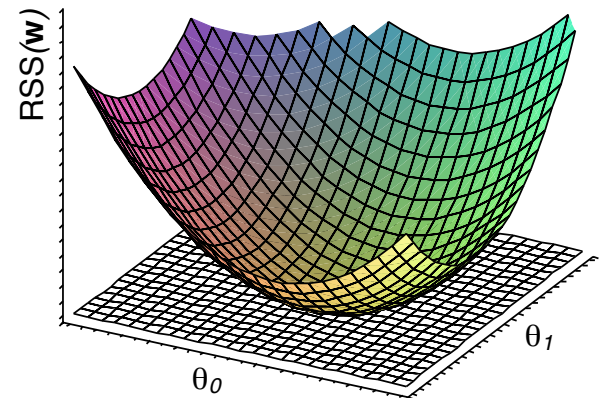
- RSS is independent of the type of model function y (linear or other) as well as of the dimensionality of $\mathbf{x}^{(i)}$.

Minimization of RSS

- The best weight vector $\hat{\mathbf{w}}$ is found by minimizing $RSS(\mathbf{w})$ on the training set.

$$\hat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^{m+1}} RSS(\mathbf{w}),$$

- If y is linear, $RSS(\mathbf{w})$ is a convex function with a single, global optimum.



Linear Regression

Regularization and RSS Optimization

Regularization

- To avoid overfitting, a regularization term is usually added to the cost function, which prevents the model from becoming too complex.

$$RSS(\mathbf{w}) = \sum_{i=1}^n (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \lambda \cdot \sum_{j=0}^m \theta_j^2$$

- The regularization term restricts the absolute values of the weights \mathbf{w} .
- The hyperparameter λ trades the goodness of fit against the simplicity of the model.

RSS Optimization

- Regression methods based on the minimization of squared residuals are called *methods of least squares*.
- Various methods have been devised, particularly for linear models.
- **General types.** Batch methods and iterative methods.

Linear Regression

Batch Gradient Descent and Stochastic Gradient Descent

Linear regression with batch gradient descent

- In each step, y is adapted to all training instances.
- The adaptation is based on the gradient of $RSS(\mathbf{w})$ at the current point.
- Stepwise heads towards the minimum of $RSS(\mathbf{w})$ until convergence.
- **Pro.** Guarantees to find the (single) minimum of $RSS(\mathbf{w})$.
- **Con.** Needs complete data and is thus hardly applicable to big data.

Linear regression with stochastic gradient descent (SGD)

- Adapts the model iteratively to each single instance.
- The adaptation is repeated for some number of *epochs* k .
- The higher k , the more the model will be fit to the data.
- **Pro.** Linear in n and k and, thus, particularly appropriate for big data.
- **Pro.** Can continually learn *online*, whenever given new training data.
- **Pro.** Can make use of stochastic sampling methods.
- **Con.** Does not guarantee to find the minimum.

Linear Regression

Learning Rate

Learning rate of gradient descent

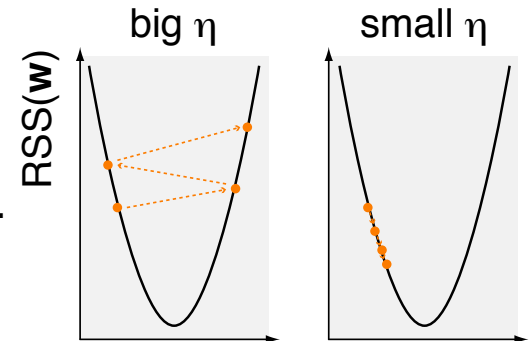
- To fit the weights in w , gradient descent adapt them based on the gradient of the residuals.

The gradient is the correspondent of derivate for multiple dimensions.

- The learning rate η is a hyperparameter that governs the step size of the adaptation.

Large or small learning rate?

- If η is big, the minimization may fail to converge.
- If η is small, the optimization may take long.



Adaptative learning rate

- A solution is to stepwise decrease the learning rate during optimization.
- Below, this is left out for simplicity.

Linear Regression

Pseudocode of Linear Regression with SGD

Signature

- **Input.** n training instances X of the form (\mathbf{x}, y) , a learning rate η , and a number of epochs k .
- **Output.** A vector \mathbf{w} with one weight θ_j for each feature $x_j \in \mathbf{x}$, $1 \leq j \leq m$.

linearRegressionWithSGD (List<Instance> X , double η , int k)

```
1.   List<double> w ← getNRandomValues(-1, 1)
2.   for int j ← 1 to k do
3.       for each Instance  $(\mathbf{x}^{(i)}, y^{(i)})$  in  $X$  do
4.           double  $y(\mathbf{x}^{(i)}) \leftarrow \mathbf{w}^T \mathbf{x}^{(i)} = \theta_0 + \theta_1 \cdot x_1^{(i)} + \dots + \theta_m \cdot x_m^{(i)}$ 
5.           List<double> gradient  $\leftarrow \frac{\partial}{\partial \theta_j} (((y^{(i)} - y(\mathbf{x}^{(i)}))^2 + \frac{\lambda}{n} \cdot \sum_{j=0}^m \theta_j^2)$ 
6.            $\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \text{gradient}$ 
7.   return w
```

Notice

- Differs from the pseudocode in lecture part VI, due to focus on RSS, inclusion of regularization, and similar.

Review Sentiment Analysis

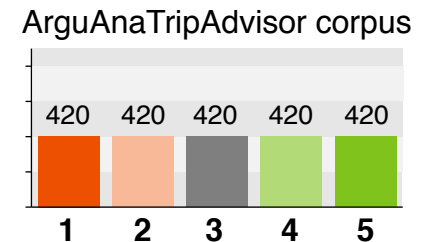
Sentiment Scoring

Sentiment scoring of reviews

- Regression of the numeric sentiment score of a customer review on a product, service, or work of art.

Data (as above)

- 2100 English hotel reviews from TripAdvisor.
900 training, 600 validation, and 600 test reviews.
- Each review has a sentiment score from $\{1, \dots, 5\}$.



Tasks

- **5-class sentiment.** Each score used as numeric value.

Approach

- **Algorithm.** Linear regression with stochastic gradient descent (SGD).
Epoch hyperparameter tuned on validation sets, other parameters fixed.
- **Features.** Combination of standard and specific features (as above).

Review Sentiment Analysis

Evaluation of Sentiment Regression

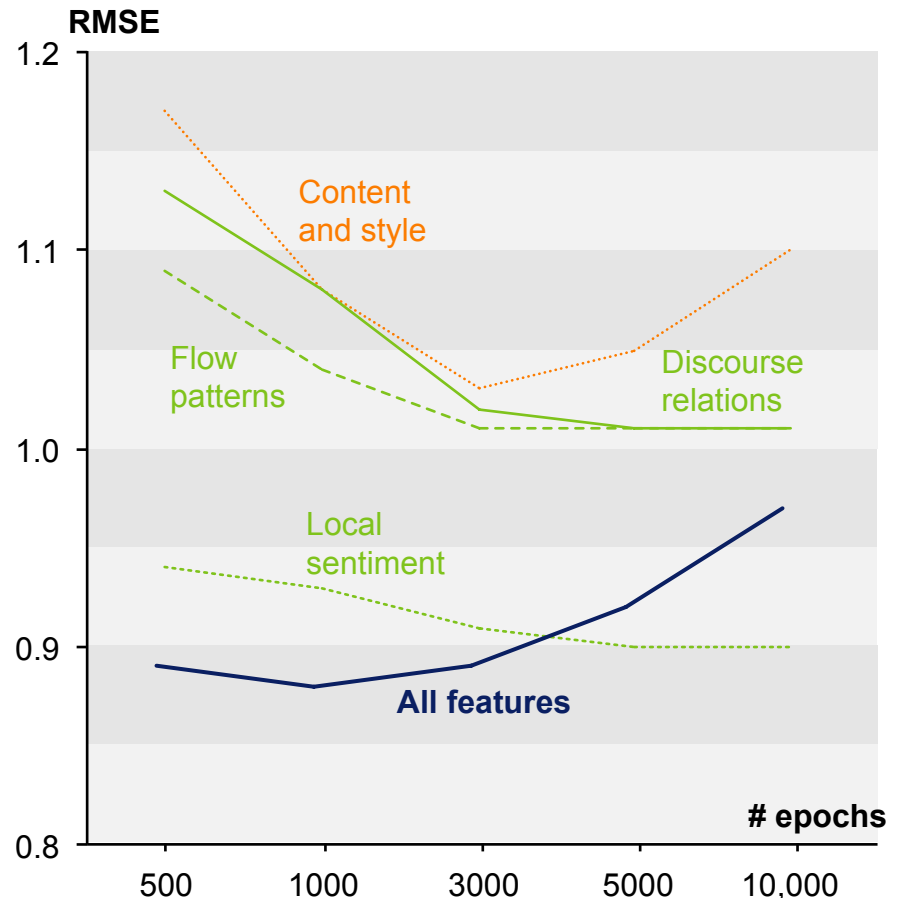
Evaluation

- Linear regressor for each feature type in isolation and for their combination.
- Training on training set, tuning on validation set, application on test set.
- **Main measure.** RMSE: root mean squared error.

Hyperparameter tuning

- **Epochs tested for SGD.** 0.5k, 1k, 3k, 5k, 10k.
- Best epochs for test set.
- Learning rate set to 10^{-5} , same for regularization.

Validation RMSE depending on epochs



Review Sentiment Analysis

Results and Discussion of Sentiment Regression

Effectiveness results on test set

Feature type	# Features	MAE	RMSE
Local sentiment distribution	50	0.77	0.99
Discourse relation distribution	75	0.82	1.01
Sentiment flow patterns	42	0.86	1.07
Content and style features	1026	0.90	1.11
Combination of features	1193	0.73	0.93
Random baseline		1.20	1.41

Discussion

- **Content and style features.** Worst in regression, unlike in classification. Indicates that they sometimes fail strongly.
- **Local sentiment distribution.** Consistently most effective in hotel review sentiment analysis.
- **Combination of features.** Improves over single features type here, too.
- The best MAE and RMSE values do not seem fully convincing.

Review Sentiment Analysis

Ground-truth vs. Predicted Scores

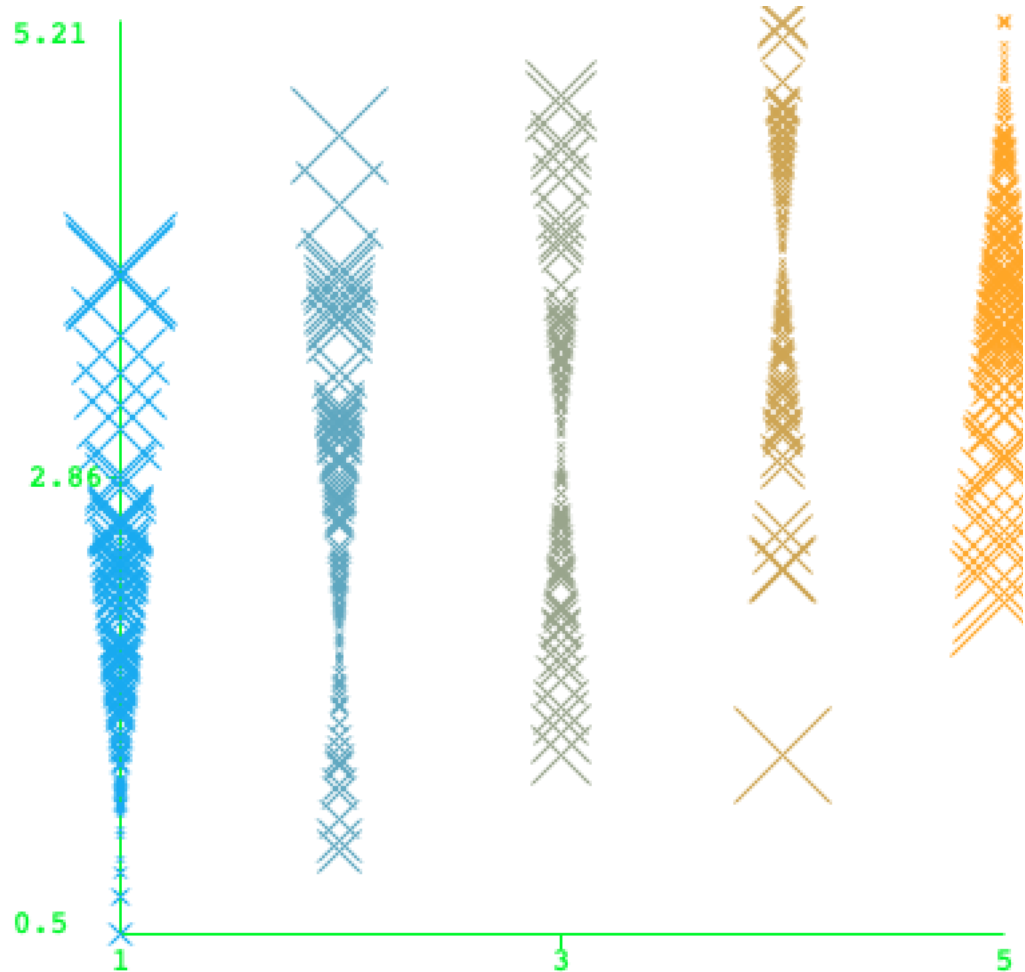
Visualization

- Scatter plot of the ground-truth scores in the test set vs. predicted scores of the best regressor.

Observations

- General tendency of regressor okay.
- Clear outliers exist for all scores.
- Few high and low predicted scores.

Effect of the regression towards the mean.



(cross size reflects regression error)

Review Sentiment Analysis

Main Features in the Evaluated Regression Model

Features with highest positive weights

- 0.6457 First local sentiment in text
- 0.2768 Proportion of neutral clauses
- 0.2186 elaboration(positive, positive)
- 0.2001 Last local sentiment in text
- 0.1682 # Clauses per sentence
- 0.1681 SentiWordnet objectivity score
- 0.1477 (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

Features with highest negative weights

- -0.2438 elaboration(negative, negative)
- -0.1246 Local sentiment change (negative, neutral)
- -0.0858 Character trigram “d s”
- -0.0800 sequence(neutral, neutral)
- -0.0714 Character trigram “t o”
- -0.0714 sequence(negative, neutral)
- -0.0691 Token bigram “. i”

Review Sentiment Analysis

Impact of Preprocessing

Evaluation on ground-truth local sentiment

- The specific features rely on the output of local sentiment classifiers.
Subjectivity accuracy 78.1%, polarity accuracy 80.4%
- Using the ground-truth local sentiment available in the given data, the impact of errors in this output can be assessed.

Effectiveness results with ground-truth local sentiment

Feature type	# Features	MAE	RMSE
Local sentiment distribution	50	0.61	0.77
Discourse relation distribution	75	0.68	0.84
Sentiment flow patterns	42	0.67	0.86
Content and style features	1026	0.90	1.11
Combination of features	1193	0.61	0.75
Random baseline		1.20	1.41

Discussion

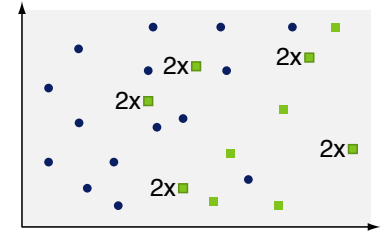
- Notable error reduction, but room for further improvement remains.

Conclusion

Summary

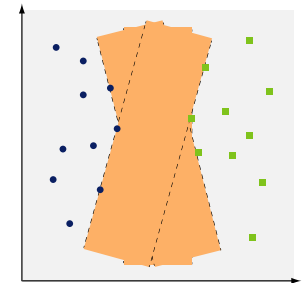
Text mining using classification and regression

- Decisions about and assessment of text properties.
- Usually done with supervised learning.
- Datasets may have to be prepared for learning.



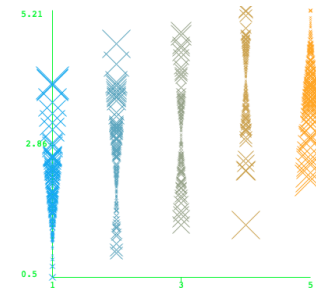
Supervised classification

- Learning to predict nominal labels of texts / text spans.
- One of the most used learning algorithms is the SVM.
- Feature engineering is key to high effectiveness.



Supervised regression

- Learning to predict numeric values of texts / text spans.
- Simple linear regression is effective and interpretable.
- Also here: Feature engineering is key.



References

Some content and examples taken from

- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). Introduction to Information Retrieval. Cambridge University Press.
- Henning Wachsmuth (2015): Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining. LNCS 9383, Springer.
- Ian H. Witten and Eibe Frank (2005): Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition.