

# Statistical Natural Language Processing

## Part VI: NLP using Classification and Regression

Henning Wachsmuth

<https://ai.uni-hannover.de>

# Learning Objectives

## Concepts

- How to prepare datasets for supervised learning
- How to employ classification within NLP
- How to employ regression within NLP

## Methods

- Classification of a text with support vector machines
- Engineering of features for a given text analysis task
- Scoring of texts with linear regression

## Tasks

- Sentiment polarity classification
- Sentiment scoring

# Outline of the Course

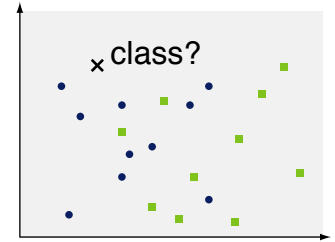
- I. Overview
- II. Basics of Data Science
- III. Basics of Natural Language Processing
- IV. Representation Learning
- V. NLP using Clustering
- VI. NLP using Classification and Regression
  - Introduction
  - Data Preparation
  - Supervised Classification
  - Supervised Regression
  - Conclusion
- VII. NLP using Sequence Labeling
- VIII. NLP using Neural Networks
- IX. NLP using Transformers
- X. Practical Issues

# Introduction

# Classification and Regression

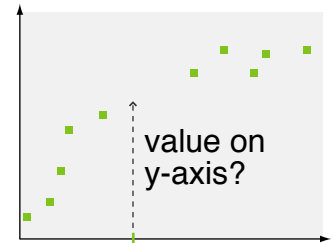
## Classification

- 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.  
Even in case the values actually have an order or distance



## Regression

- The task to assign an instance to the most likely value from a real-valued scale
- Values are continuous and (conceptually) not bounded  
Even in case the values actually come from a bounded range



## Why using supervised learning?

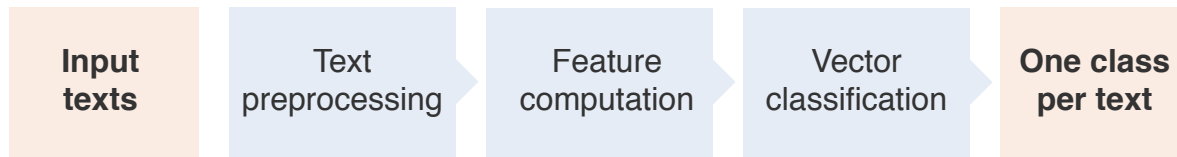
- Hand-crafted rules/arithmetics could be used to predict labels/values.
- With sufficient data, supervised learning is mostly more effective.
- We restrict our view to *feature-based* methods in this lecture part.  
Neural methods follow in Lecture Part VIII.

# Classification and Regression

## NLP using Supervised Classification and Regression

### Feature-based text classification (regression analogous)

- Supervised classification based on feature representations of texts
- **Input.** A set of texts or text spans  $O$ , represented in a feature space  $X$  (in training with class information  $C$ )
- **Output.** A class  $c \in C$  for each  $o \in O$ , and a model  $y : X \rightarrow C$



### Challenges of feature-based methods

- Develop features that help solve a given task.
- Choose a suitable learning algorithm that fits the data.
- Optimize important hyperparameters of the algorithm.

# Classification and Regression

## Evaluation and Application

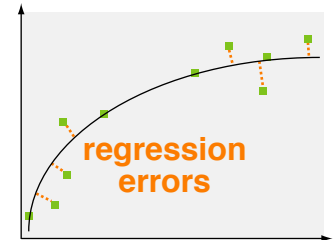
### Evaluation of supervised classification

- **Goal.** Classify as many test instances as possible correctly (from all or particular classes).
- **Measures.** Accuracy, precision, recall,  $F_1$ -score



### Evaluation of supervised regression

- **Goal.** Minimize the mean difference between predicted and correct test values.
- **Measures.** Mean absolute error, mean squared error



### Numerous applications in NLP

- **Classification.** Deciding about span boundaries, span types, text labels, relations between spans, relation types, ...
- **Regression.** Assigning scores and ratings, estimating probabilities, ...

# Data Preparation

# Dataset Preparation

## Why data preparation?

- Not always, the annotations in a corpus match the task instances required for supervised learning.



## Typical preparations

- **Instance creation.** Add missing “negative” instances, e.g., entity corpora only show what *is* an entity:

“**[Jaguar]**<sub>ORG</sub> is named after the animal **jaguar**.”

- **Variable mapping.** Map annotations to other target variables, especially for abstraction or unification:

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

- **Dataset balancing.** Make the distribution of the target variable uniform:

1000 negative, 600 neutral, 800 positive → 600 negative, 600 neutral, 600 positive

# Dataset Preparation

## Instance Creation

### Why creating instances?

- In many classification tasks, one (“positive”) 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 the negative class

- What to choose as negative instances is a design decision.
- This 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 name recognition

“**[tim]**<sub>PER</sub> works in **[cupertino]**<sub>LOC</sub>. **[san fran]**<sub>LOC</sub> is his home. as a cook, he cooks all day.”

- All other named entities? → Can distinguish only entity *types* then
- All other content words? → Verbs will never be person names
- All other noun phrases? → Typical choice (alternative: token-level tags, see Part VII)

# Dataset Preparation

## Dataset Balancing

### Dataset balancing

- The alteration of the distribution of a dataset regarding a target variable, such that the distribution is uniform afterwards
- Balancing works by either *undersampling* or *oversampling* instances.  
For regression, an alternative is *binning*, i.e., to make certain intervals uniform.

### When to balance?

- Balancing a training set prevents machine learning from being biased toward majority classes (or values).
- Validation and test sets should usually have *representative* distributions.  
In the corpus in Part III, all datasets are balanced (may be useful for analysis).

### Alternatives to balancing?

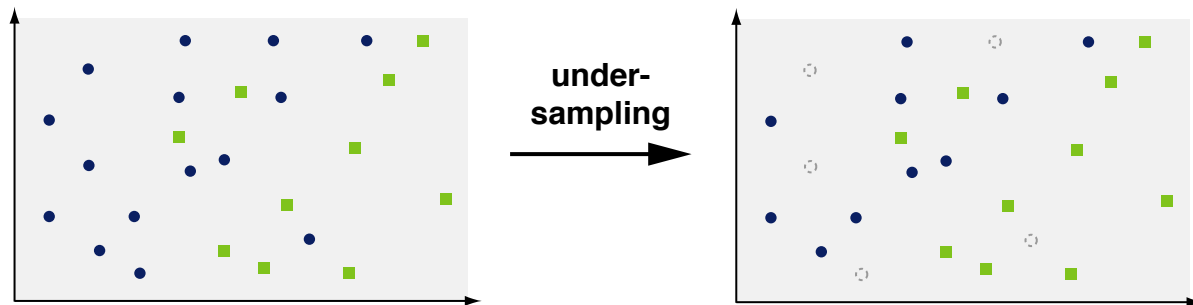
- Some optimization procedures can penalize wrong predictions more for minority instances than for majority instances.
- This is the more sound but also more complex way of preventing bias.

# Dataset Preparation

## Undersampling

### Balancing with undersampling

- Remove instances of all non-minority classes, until all classes have the size of the minority class.
- Instances to be removed are usually chosen (pseudo-) randomly.



### Pros and cons

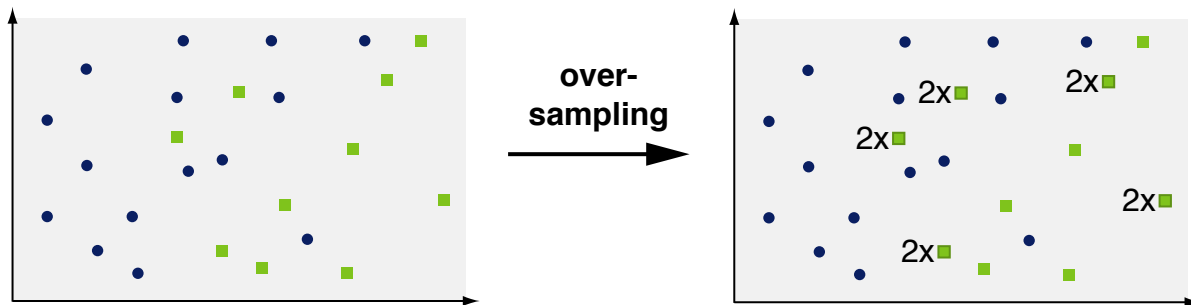
- **Pro.** All remaining data is real.
- **Pro.** Downsizing of a training set makes training faster.
- **Con.** Instances that may be helpful in learning are discarded, i.e., potentially relevant information is lost.

# Dataset Preparation

## Oversampling

### Balancing with oversampling

- Add instances of all minority classes, until all classes have the size of the majority class.
- Usually, the instances to be added are (pseudo-) random duplicates.  
In some cases, an alternative is to create artificial instances via interpolation.



### Pros and cons

- **Pro.** No instance is discarded, i.e., all information is preserved.
- **Con.** Upsizing of a training set makes training slower.
- **Con.** The impact of duplicated instances is boosted artificially, which may make features discriminative that are actually noise.

# Supervised Classification

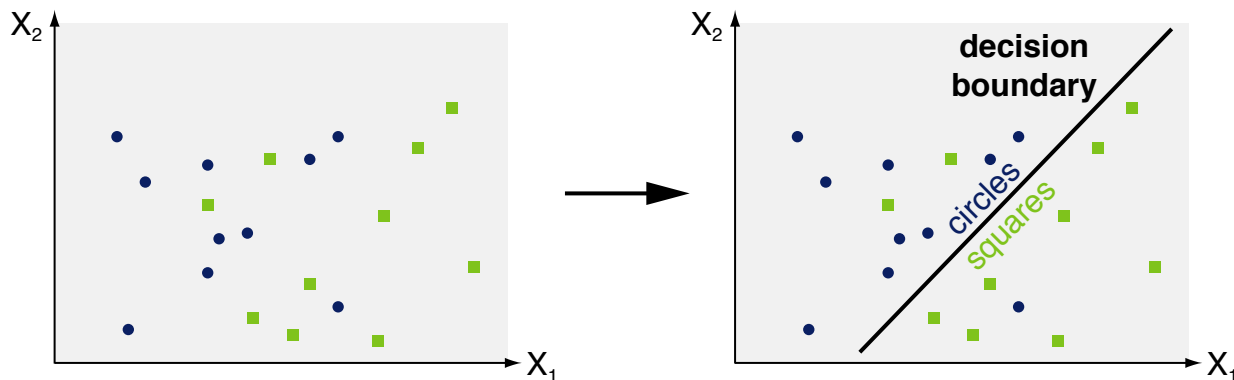
# Supervised Classification

## Supervised classification

- The learned prediction of the most likely of a set of  $k > 1$  predefined nominal classes for an instance

## 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



## Application phase (prediction)

- **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

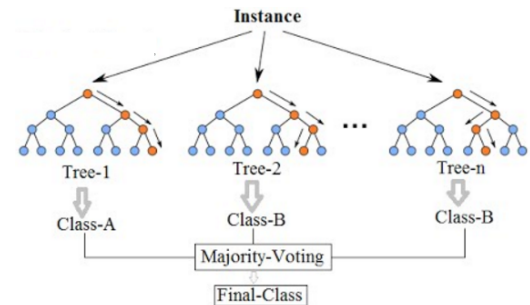
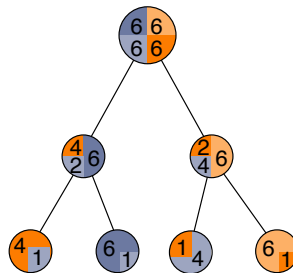
## Classification Algorithms

### Selected classification algorithms

- **Naïve Bayes.** Predict classes based on conditional probabilities.
- **Decision tree.** Stepwise compare instances on single features.
- **Random forest.** Take the majority vote of several decision trees.

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

$$P(c|\mathbf{x}) \propto P(x_1|c) \cdot \dots \cdot P(x_m|c) \cdot P(c)$$



- **Support vector machine.** Maximize the margin between classes.
- **Neural network.** Learn complex functions from features.

### 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

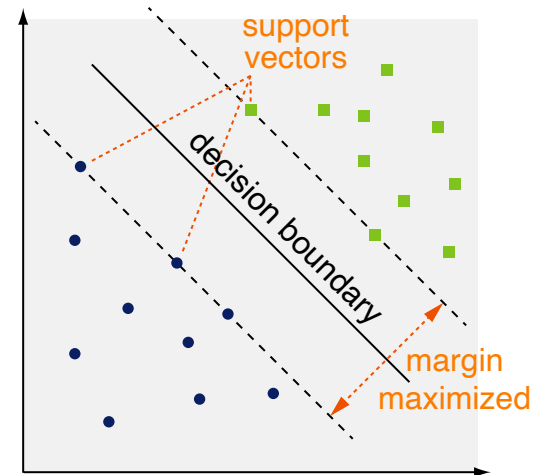
## Support vector machine (SVM)

- A learning algorithm that aims to find a *linear* decision boundary which maximizes the margin between two classes
- Non-linear classification is possible through the *kernel trick*. (more below)

## Large-margin classification

- The margin is the distance from the decision boundary to all closest instances.
- SVMs maximize the minimum margin of the boundary to the training instances.

Some instances may be discounted as outliers/noise.



## Support vectors

- A (usually small) subset of training instances that are used by an SVM to define the decision boundary
- Other instances are not memorized after training.

# Support Vector Machines

## Linear SVMs

### Decision boundary

- A hyperplane  $\mathbf{w}^T \mathbf{x} = b$ , that is, all points  $\mathbf{x}$  on it satisfy the equation
- The weight vector  $\mathbf{w}$  is the *normal* perpendicular to the hyperplane.
- The intercept term  $b$  denotes the distance to the origin, scaled by  $\|\mathbf{w}\|$ .

### Training

- **Input.** A set of  $n$  training instances  $\mathbf{x}^{(i)}$  with class  $y^{(i)} = c(\mathbf{x}^{(i)}) \in \{-1, 1\}$   
Nominal classes are mapped to  $-1$  and  $1$ .
- **Output.** A linear classifier  $y(\mathbf{x}) := \text{sign}(\mathbf{w}^T \mathbf{x} - b)$ , such that  $\mathbf{w}^T \mathbf{x} = b$  maximizes the minimum distance to instances  $\mathbf{x}^{(i)}$

### Optimization variants

- **Hard margin.** Find the best separating hyperplane. This works only for linearly separable training sets.
- **Soft margin.** Allow outliers, but penalize them. This always works.

# Support Vector Machines

## Linearly Separable Training Sets

### Maximum-margin decision boundary

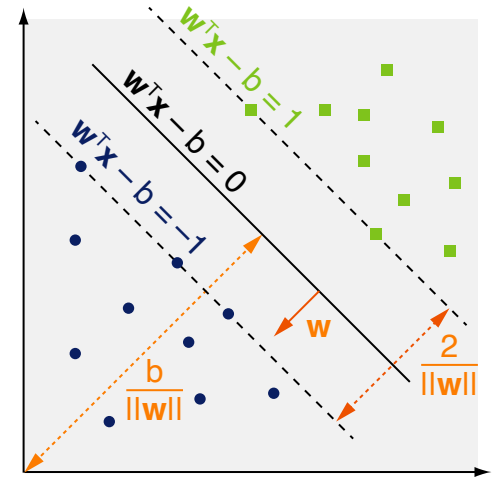
- If the training set is linearly separable, any  $\mathbf{x}^{(i)}$  must fulfill either of:

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

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

- From this, we can infer:

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



- Also, it follows that the size of the margin is  $\frac{2}{\|\mathbf{w}\|} = \frac{b+1}{\|\mathbf{w}\|} - \frac{b-1}{\|\mathbf{w}\|}$ .
- To maximize the margin,  $\|\mathbf{w}\|$  must be minimized.

### Minimization problem of hard-margin SVMs

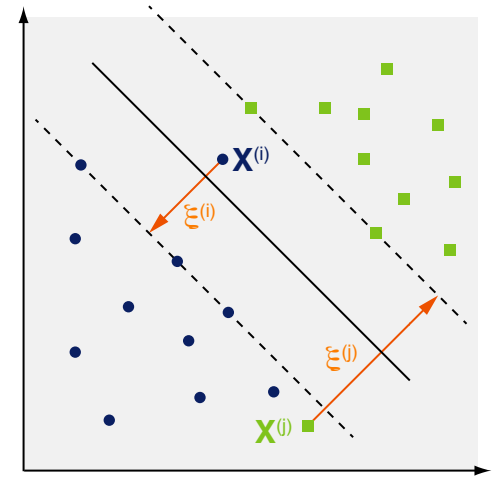
- Find  $\mathbf{w}$  and  $b$ , such that  $\|\mathbf{w}\|$  is minimized, and  $\forall i : y^{(i)} \cdot (\mathbf{w}^T \mathbf{x}^{(i)} - b) \geq 1$ .
- The support vectors that represent the hyperplane for these  $\mathbf{w}$  and  $b$  can be found on a training set.

# Support Vector Machines

## Linearly Inseparable Training Sets

### 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  $c \cdot \xi^{(i)}$  proportional to  $\xi^{(i)}$ .
- The SVM then trades the size of the margin against the number of correctly classified training instances.



### Minimization problem of soft-margin SVMs

- Find  $\mathbf{w}$ ,  $b$ , and  $\forall i : \xi^{(i)} \geq 0$ , such that  $\|\mathbf{w}\| + c \cdot \sum_i \xi^{(i)}$  is minimized, and  $\forall i : y^{(i)} \cdot (\mathbf{w}^T \mathbf{x}^{(i)} - b) \geq 1 - \xi^{(i)}$ .
- $c > 0$  is a regularization hyperparameter, 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, training instances may be misclassified at low cost.
- As  $c$  becomes larger, training misclassifications get more expensive.
- 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  $c \in \{10^{-5}, 10^{-4}, \dots, 10^4, 10^5\}$
- Then compare more fine-grained  $c$  values in this magnitude.
- The best found  $c$  is used on the test set (or in the application).

### Notice

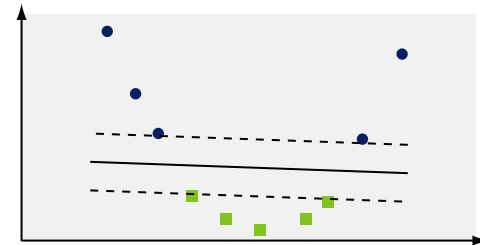
- Non-linear SVMs often have a second hyperparameter  $\gamma$  (see below).
- $c$  and  $\gamma$  then need to be optimized jointly. (see AutoML course for details :))

# Support Vector Machines

## Non-Linear Classification

### Non-linear SVMs

- Some data is only linearly “separable” with many misclassifications.
- As a solution, non-linear SVMs use the *kernel trick*.



### Kernel trick

- Map the original feature space to a higher-dimensional space where it is (better) linearly separable.
- A linear classifier is then learned for the higher-dimensional space.
- **Conceptually.** The mapping is a non-linear transformation  $\Phi : \mathbf{x} \mapsto \phi(\mathbf{x})$ .
- **Practically.** A *kernel function*  $\mathcal{K}$  is used that computes the dot product of two transformed instances in the original feature space.

# Support Vector Machines

## Kernel Functions

### Linear SVMs with kernel functions

- A linear SVM can be specified using dot products  $\mathcal{K}_{lin}(\mathbf{x}, \mathbf{x}^{(j)}) = \mathbf{x}^T \mathbf{x}^{(j)}$  for an instance  $\mathbf{x}$  and each support vector  $\mathbf{x}^{(j)}$ :

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

### Non-linear SVMs with kernel functions

- Replace  $\mathcal{K}_{lin}$  above by another kernel function that computes the dot product of two transformed instances,  $\phi(\mathbf{x})$  and  $\phi(\mathbf{x}^{(j)})$ .

For optimization, kernel functions must fulfill certain properties (omitted here).

### Common kernel functions

- **Polynomial kernels.**  $\mathcal{K}_{poly}(\mathbf{x}, \mathbf{x}^{(j)}) := (1 + \mathbf{x}^T \mathbf{x}^{(j)})^d$

A quadratic kernel ( $d = 2$ ) is often used.  $d = 1$  results in a linear kernel.

- **Radial basis function.**  $\mathcal{K}_{rbf}(\mathbf{x}, \mathbf{x}^{(j)}) := e^{-\gamma \cdot (\mathbf{x} - \mathbf{x}^{(j)})^2}$

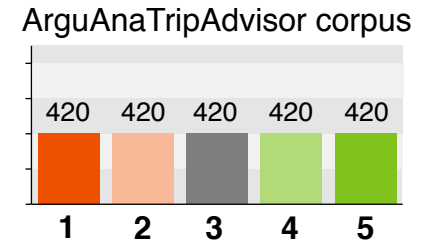
A more flexible Gaussian distribution, with additional hyperparameter  $\gamma$ .

- Kernel functions can also be designed for the specific data at hand.

# Review Sentiment Analysis

## Data (Wachsmuth et al., 2014a)

- 2100 English hotel reviews from TripAdvisor  
900 training, 600 validation, and 600 test reviews
- Each review has a sentiment score from 1–5.



## Case Study: Sentiment Polarity classification (Wachsmuth, 2015)

- Classification of the “global” sentiment label of hotel reviews
- **5-class sentiment.** Each score interpreted as one (nominal) class.
- **3-class sentiment.** 1–2 mapped to negative, 3 to neutral, 4–5 to positive  
Training set balanced with random undersampling

## Approach

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

# Review Sentiment Analysis

## Case Study: Selected Standard Feature Types

### Content features

- **Token unigrams (bag-of-words)**. The distribution of all token 1-grams that occur in at least 5% of all training texts
- **Token bigrams/trigrams**. Analogous for 2-grams and 3-grams

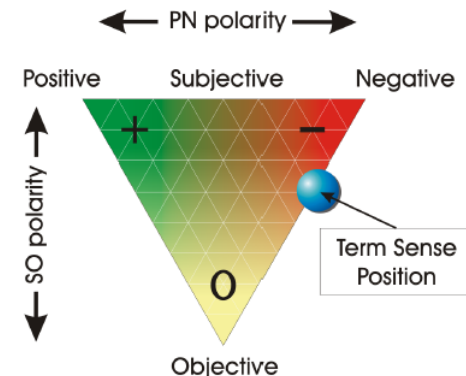
### Style features

- **POS  $n$ -grams**. Analogous for part-of-speech  $\{1, 2, 3\}$ -grams
- **Character trigrams**. Analogous for character 3-grams
- **Lengths**. Average numbers of tokens, sentences, tokens/sentence, ...

### Target class features

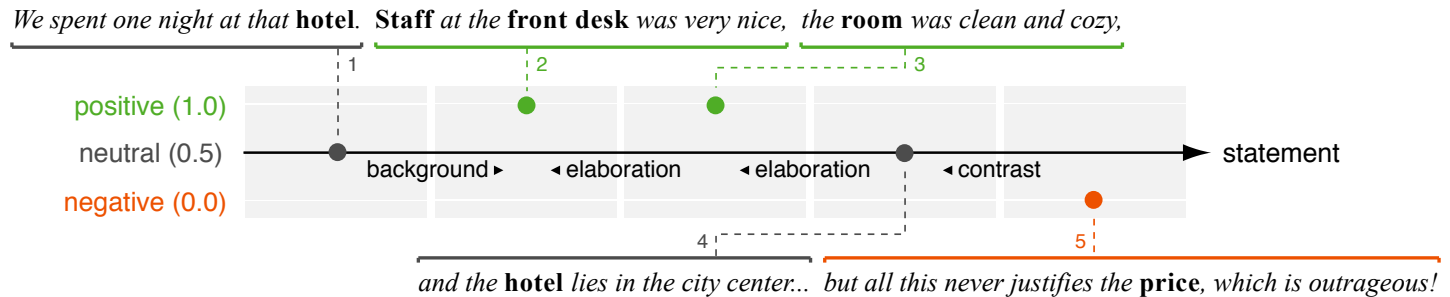
- **SentiWordNet scores**. Mean positivity, negativity, and objectivity of all first and average word senses in *SentiWordNet*

A lexicon with subjectivity and polarity values for words (Baccianella et al., 2010)



# Review Sentiment Analysis

## Case Study: Selected Specific Feature Types (1 of 2)



## Local sentiment distribution

- Local sentiment mean, frequencies, changes, and normalized positions

mean 0.6    positive 0.4    neutral 0.4    negative 0.2    (neutral, positive) 0.25 ...

9 normalized positions: (0.5, 0.75, 1.0, 1.0, 1.0, 0.75, 0.5, 0.25, 0.0)

## Discourse relation distribution

- Frequency of discourse relations and their combination with sentiment

background 0.25    elaboration 0.5    contrast 0.25    (all others 0.0)

background(neutral, positive) 0.25    elaboration(positive, positive) 0.25 ...

# Review Sentiment Analysis

## Case Study: Selected Specific Feature Types (2 of 2)

### Sentiment flow patterns (see previous lecture part)

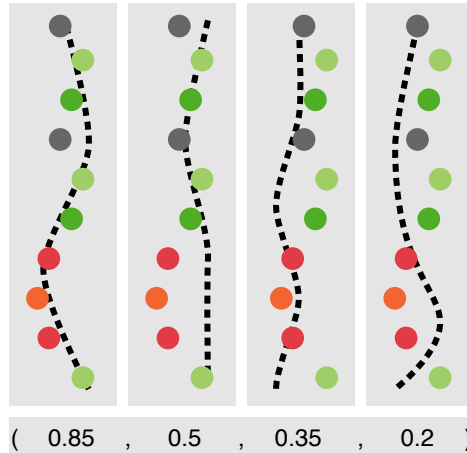
- Compute similarity of sentiment flow to each learned flow pattern.
- Each similarity becomes one global-structure feature for prediction.

Normalized/abstracted flow



①  
Computing similarities

One similarity value for each flow pattern



②  
Learning prediction

Output class or value



## Hypotheses

- Similar flows indicate similar global sentiment. (evaluated below)
- Similar flow patterns occur across review domains. (evaluated later)

# Review Sentiment Analysis

## Case Study: Experimental Setup and Hyperparameter Optimization

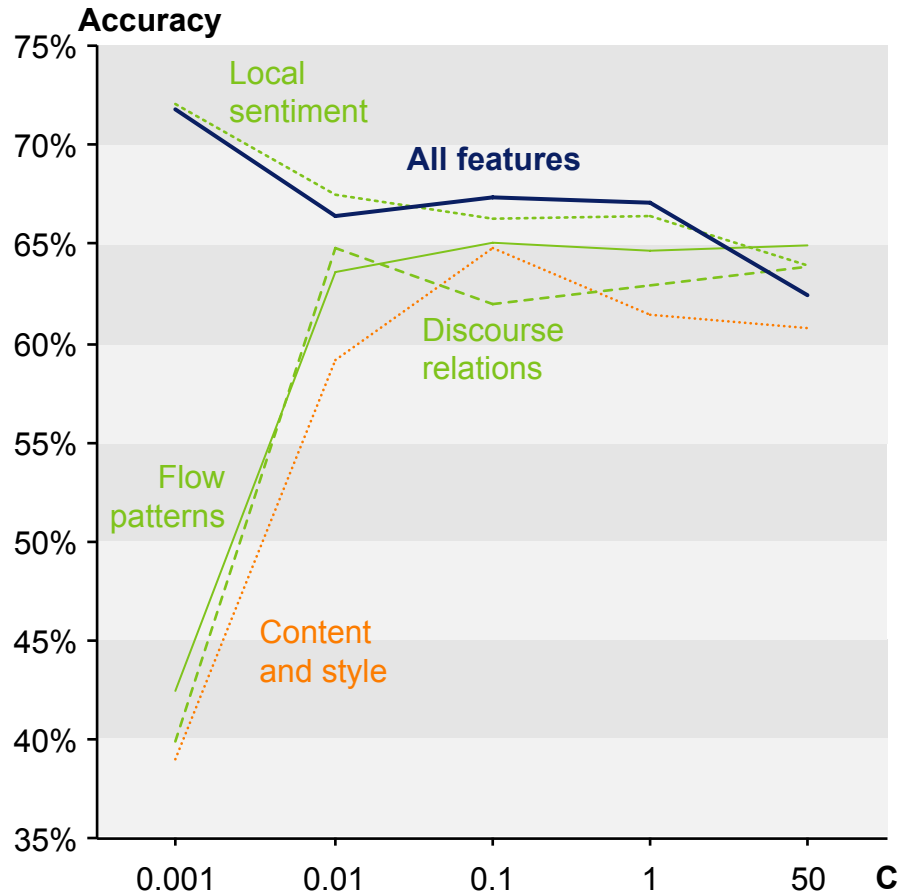
### Experimental setup

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

### Cost hyperparameter tuning

- Tested  $c$  values. 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

## Case Study: Results and Discussion

### Effectiveness results on test set (accuracy)

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

### Discussion

- **Standard features.** Worse than specific features in the 3-class task
- **Combination of features.** Works best, indicating complimentary features
- **Sentiment flow patterns.** Impact more visible across domains

This will be demonstrated in Lecture Part IX.

- Particularly the 5-class accuracy seems insufficient.
- Classification misses to model the ordinal relation between classes.

# Supervised Regression

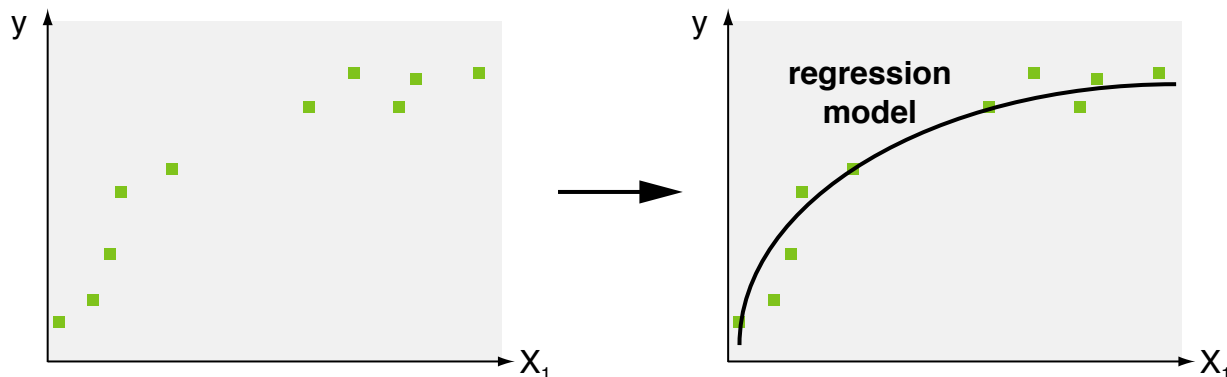
# Supervised Regression

## Supervised regression

- The learned prediction of the most likely value from a real-valued scale for an instance

## 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



## Application phase (prediction)

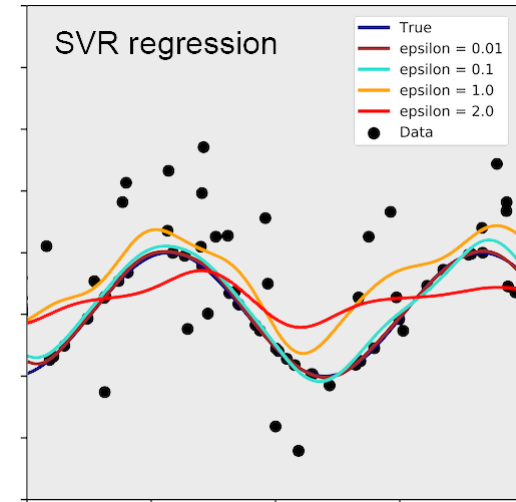
- **Input.** A set of unknown instances  $\mathbf{x}^{(i)}$  without output value
- **Output.** The predicted output value  $y(\mathbf{x}^{(i)})$  for each instance

# Supervised Regression

## Regression Algorithms

### Selected supervised regression algorithms

- **Linear regression.** Predict output values using a learned linear function.
- **Neural network.** Learn complex functions on input representation.
- **Support vector regression.** Maximize the flatness of a regression model.



Adapted from <https://commons.wikimedia.org>

### Focus on linear regression here

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

# Linear Regression

## Linear regression

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

$$y(\mathbf{x}) := \mathbf{w}^T \mathbf{x} = w_0 + \sum_{j=1}^m w_j \cdot x_j = w_0 + w_1 \cdot x_1 + \dots + w_m \cdot x_m$$

- The weight vector  $\mathbf{w} = (w_0, w_1, \dots, w_m)$  is learned on training instances.

## Training of a linear regression model

- **Input.** 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
- **Output.** The model  $y(\mathbf{x})$  found to minimize the *regression error*, i.e., the deviation of the predictions  $y(\mathbf{x}^{(i)})$  from the correct values  $y^{(i)}$

## Loss function in linear regression

- Usually, a 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  $\mathbf{x}^{(i)}$ :

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

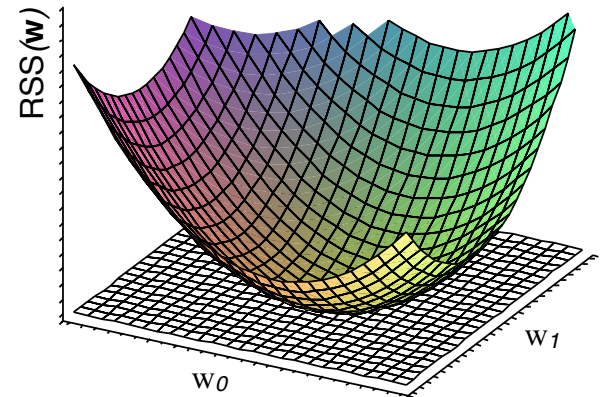
- RSS works as a loss function for any regression function  $y$  (linear or not) and for any dimensionality of  $\mathbf{x}^{(i)}$ .

### 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 Optimization

### RSS regularization

- To avoid overfitting, a regularization term is added to the loss function.
- It prevents  $y$  from becoming too complex, by penalizing high (absolute) weights.

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

- The hyperparameter  $\lambda$  trades the training error against model simplicity.

### RSS optimization

- The most common optimization method is *gradient descent*.
- It stepwise adapts a model  $y$  based on the gradient of the loss function.
- We here look at the variant *stochastic gradient descent* (SGD), which adapts  $y$  to a single training instance in each step.

Actually, an *incremental* variant of SGD is presented below (no stochastic sampling).

- **Pro.** Scales well, allows for online learning and stochastic sampling
- **Con.** Does not guarantee to find the minimum

# Linear Regression

## Pseudocode of Linear Regression with (Incremental) SGD

### Signature

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

### **linearRegressionWithSGD (List<Instance> $X$ , double $\eta$ , int $k$ )**

```
1.   List<double> w ← getMRandomValues(-1, 1)
2.   for int l ← 1 to k do
3.       for each Instance  $(\mathbf{x}^{(i)}, y^{(i)})$  in  $X$  do
4.           double  $y(\mathbf{x}^{(i)}) \leftarrow \mathbf{w}^T \mathbf{x}^{(i)} = w_0 + w_1 \cdot x_1^{(i)} + \dots + w_m \cdot x_m^{(i)}$ 
5.           List<double> gradient  $\leftarrow \frac{\partial}{\partial w_j} ((y^{(i)} - y(\mathbf{x}^{(i)}))^2 + \frac{\lambda}{n} \cdot \sum_{j=0}^m w_j^2)$ 
6.           w  $\leftarrow \mathbf{w} - \eta \cdot \text{gradient}$ 
7.   return w
```

### Impact of hyperparameters

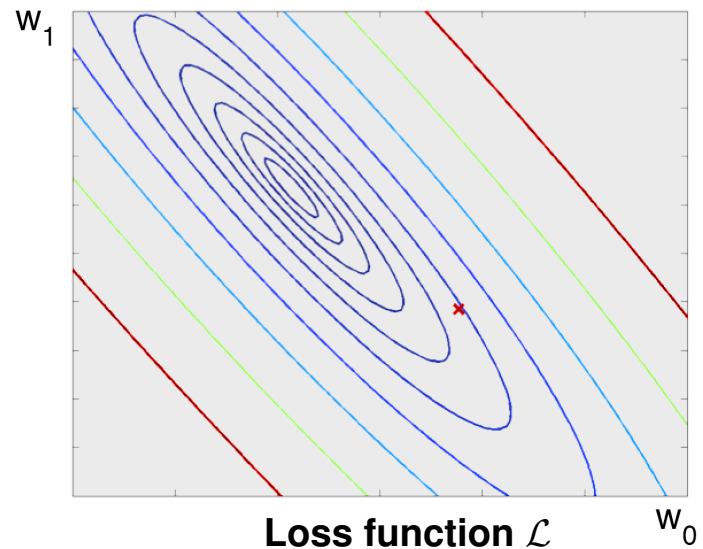
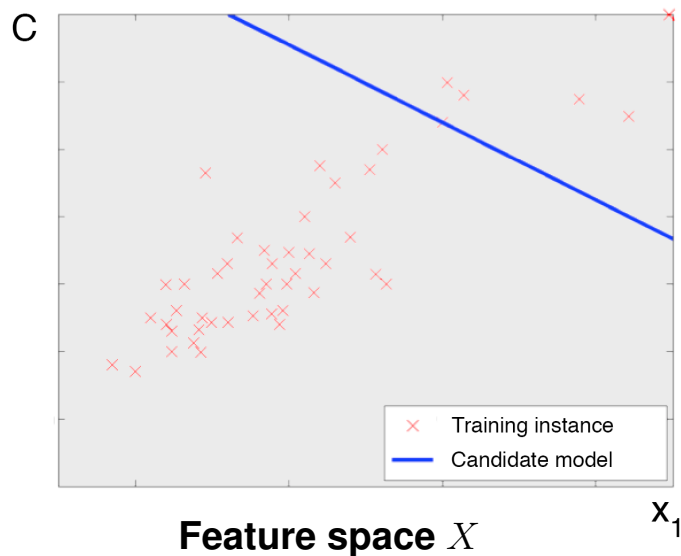
- **Number of epochs.** The higher  $k$ , the more  $y$  will fit the data.
- **Learning rate.** Higher  $\eta$  may speed up learning, but may fail to optimize.

# Linear Regression

## Gradient Descent: Example (1 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

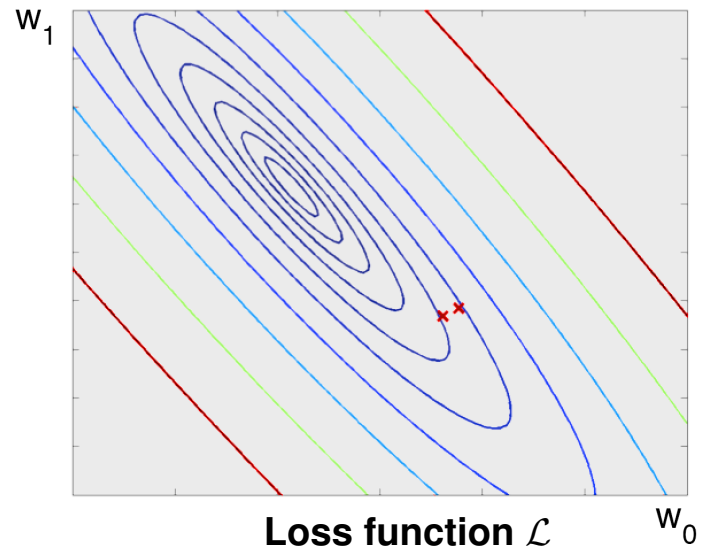
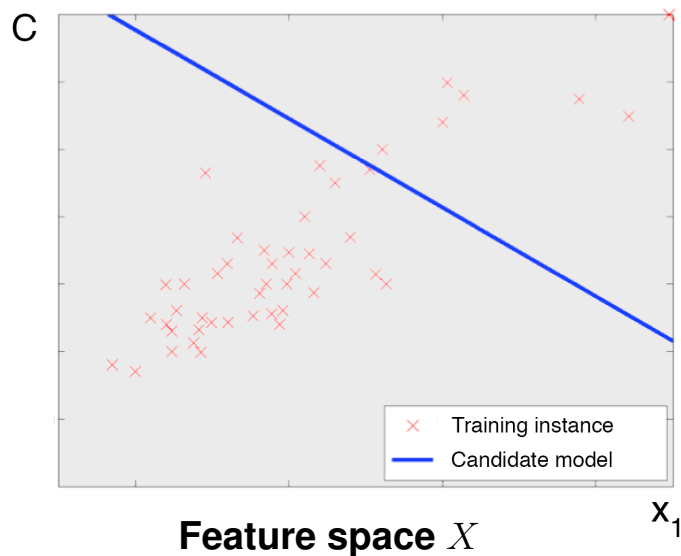
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (2 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

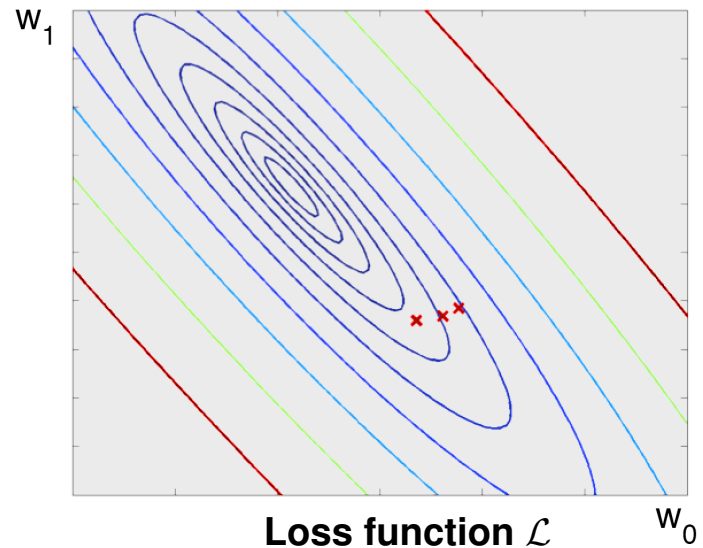
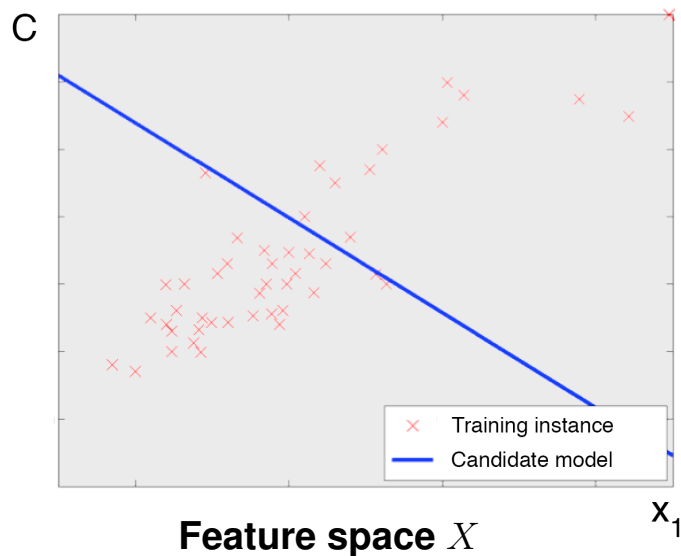
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (3 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

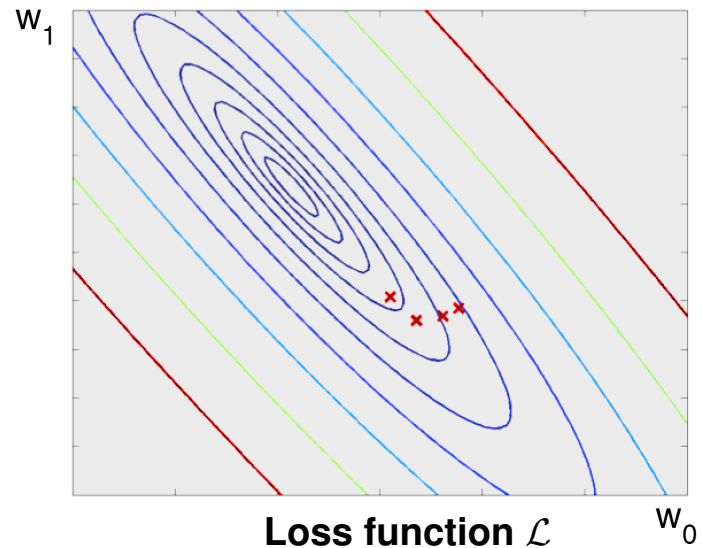
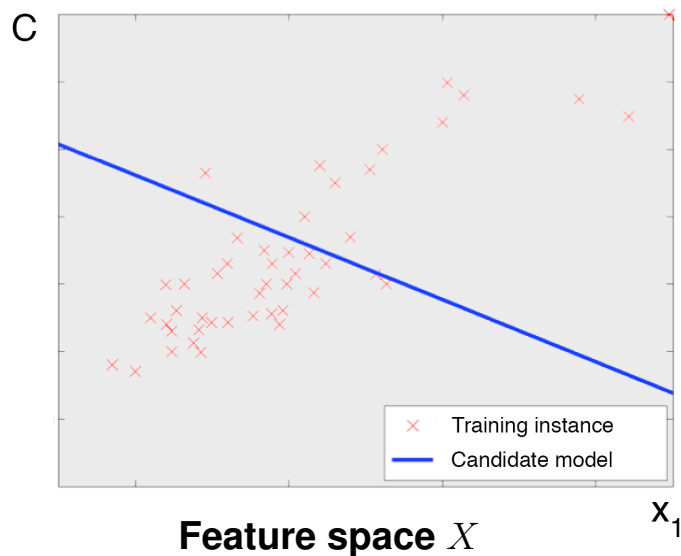
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (4 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

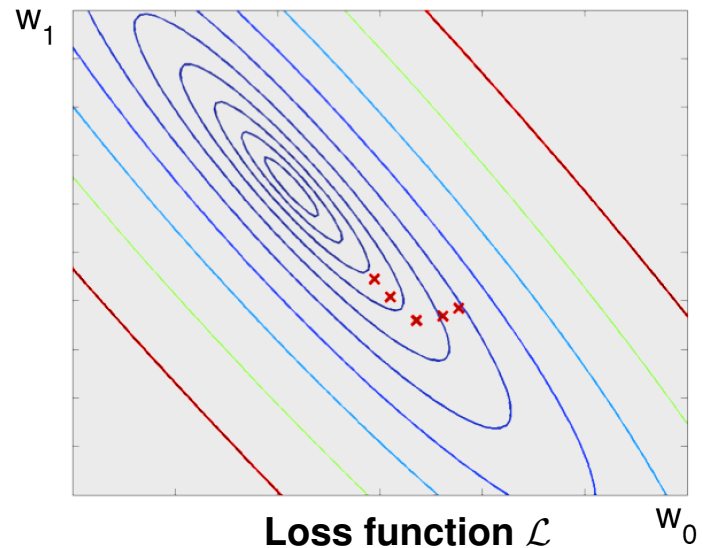
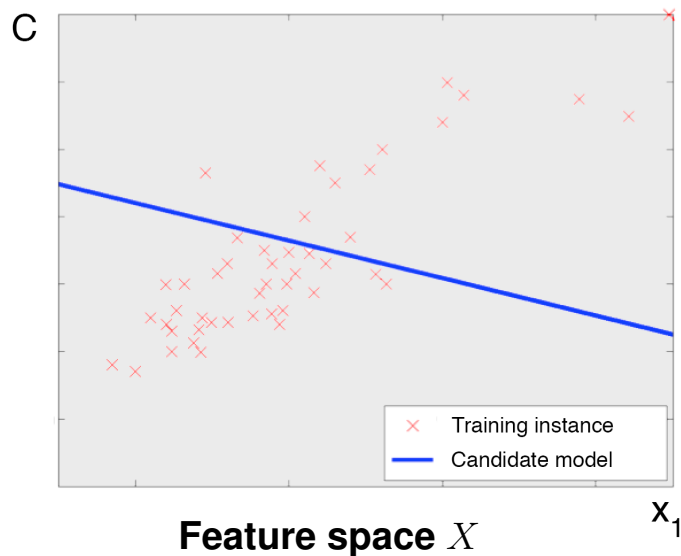
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (5 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

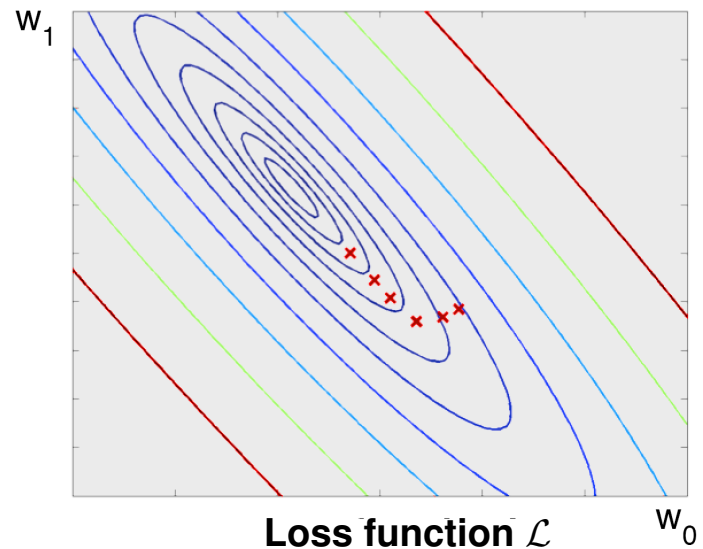
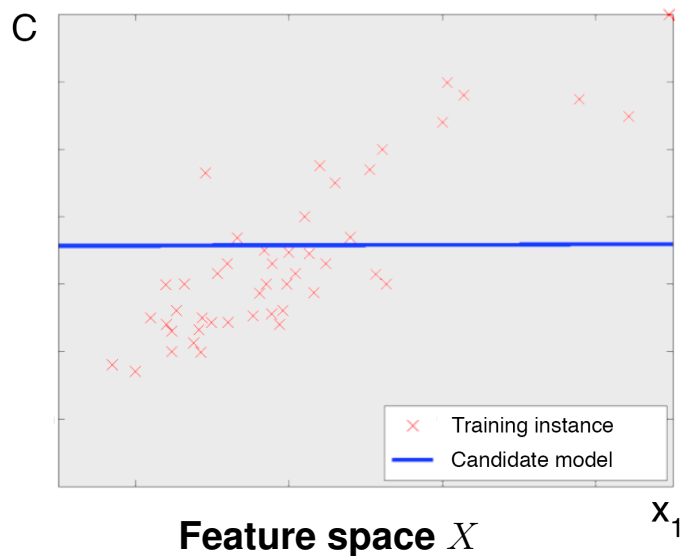
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (6 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

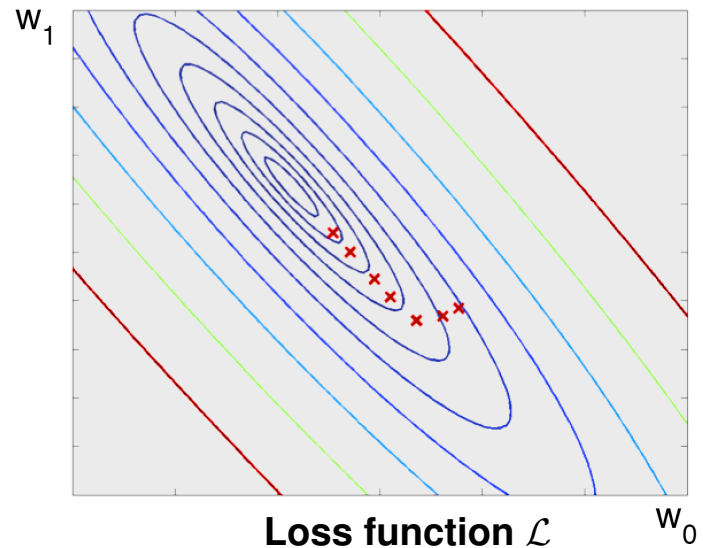
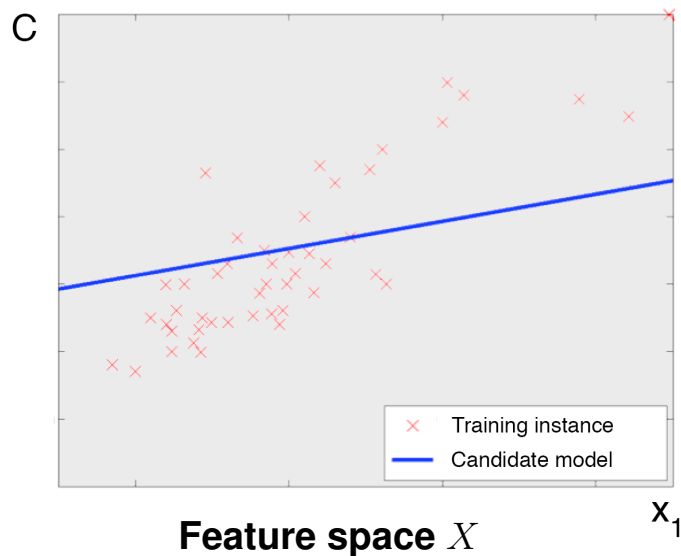
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (7 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

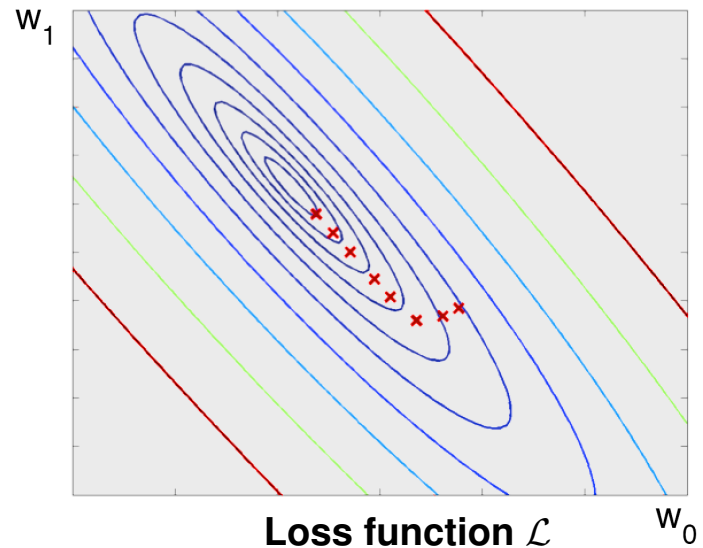
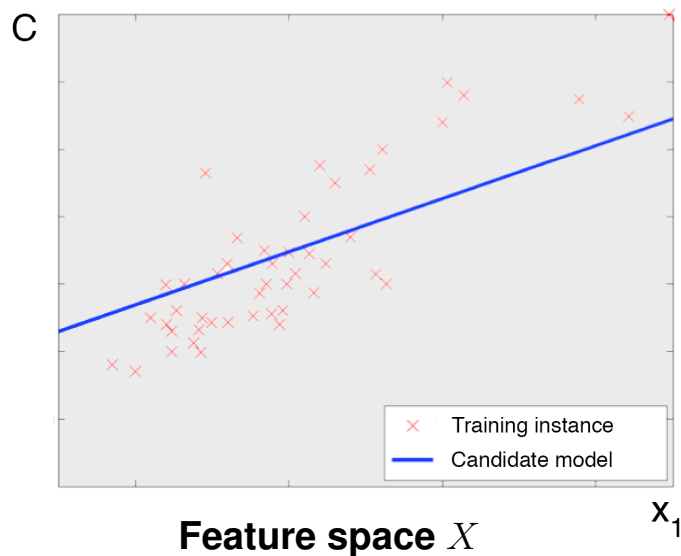
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (8 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

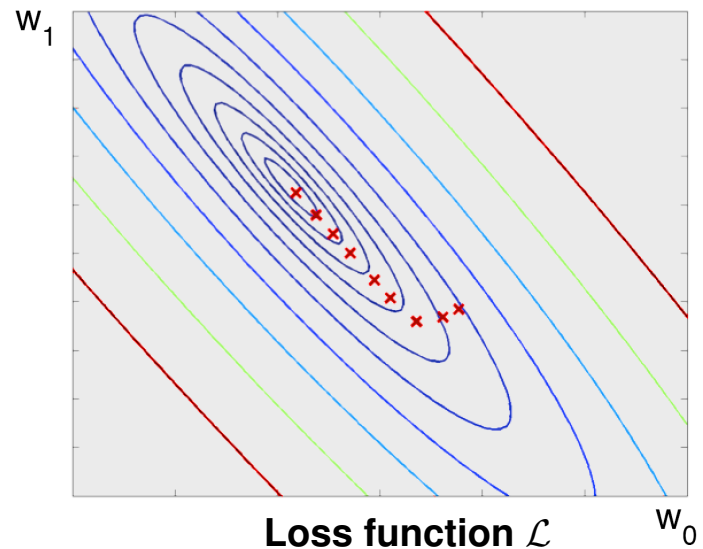
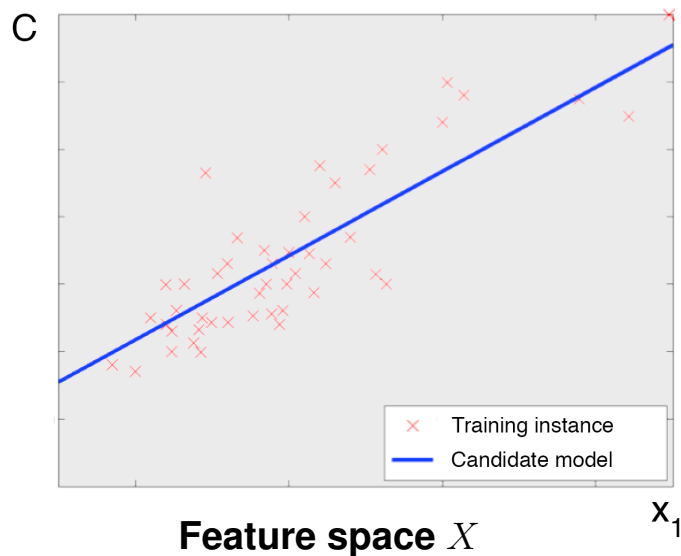
- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Linear Regression

## Gradient Descent: Example (9 out of 9)

### Learning of a linear regression model

- Feature space  $X$  (left) and loss  $\mathcal{L} = RSS_{reg}(\mathbf{w})$  (right) in training



Adapted from Ng (2018)

### Note on example

- A regression model  $y(\mathbf{x}) = w_0 + w_1 \cdot x_1$  is learned for a single feature  $x_1$ .
- Each pair  $w_0, w_1$  defines one candidate model.

# Review Sentiment Analysis

Case Study: Sentiment Scoring of Hotel Reviews (Wachsmuth et al., 2014b)

## Task

- Regression of the numeric global sentiment score of a review
- **5-class sentiment.** Each score used as numeric value

Data as for classification above

## Approach

- **Algorithm.** Linear regression with stochastic gradient descent (SGD)  
Epoch hyperparameter tuned on validation sets, other parameters fixed
- **Features.** Exactly as for classification above

## Experimental setup

- Linear regressor for each feature type alone and for their combination
- Training on training set, tuning on validation set, test on test set
- **Metrics.** Mean absolute error (MAE), root mean squared error (RMSE)

# Review Sentiment Analysis

## Case Study: Results and Discussion

### Effectiveness results on test set

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

### Discussion

- **Standard features.** Worst here (unlike in classification); this suggests that they sometimes fail strongly.
- **Local sentiment distribution.** Consistently best feature in this case study
- **Combination of features.** Improves over single feature types here, too
- The best MAE and RMSE values do not seem fully convincing.
- More advanced techniques may be needed (neural methods?).

# Review Sentiment Analysis

## Case Study: Main Features used in Regression

### Interpretability of linear regression

- Linear regression simply assigns one weight to each given feature.

### Top features of the best model

- +0.6457 First local sentiment in text
- +0.2768 Proportion of neutral clauses
- +0.2186 Discourse relation: elaboration(positive, positive)
- +0.2001 Last local sentiment in text
- +0.1682 # Clauses per sentence
- +0.1681 SentiWordnet objectivity score
- +0.1477 Flow pattern: 10x positive, 10x negative, 10x positive
- -0.0691 Token bigram “. i” (origin: next sentence starts with “I ...”)
- -0.0714 Local sentiment bigram (negative, neutral)
- -0.0714 Character trigram “t o” (“bit of”, “just okay”, “not one”, “without our”, ...)
- -0.0800 Local sentiment bigram (neutral, neutral)
- -0.0858 Character trigram “d s” (“had some”, “had stayed”, “and saw”, ...)
- -0.1246 Local sentiment change (negative, neutral)
- -0.2438 Discourse relation: elaboration(negative, negative)

# Review Sentiment Analysis

## Case Study: Error Analysis (Ground-truth Scores vs. Predictions)

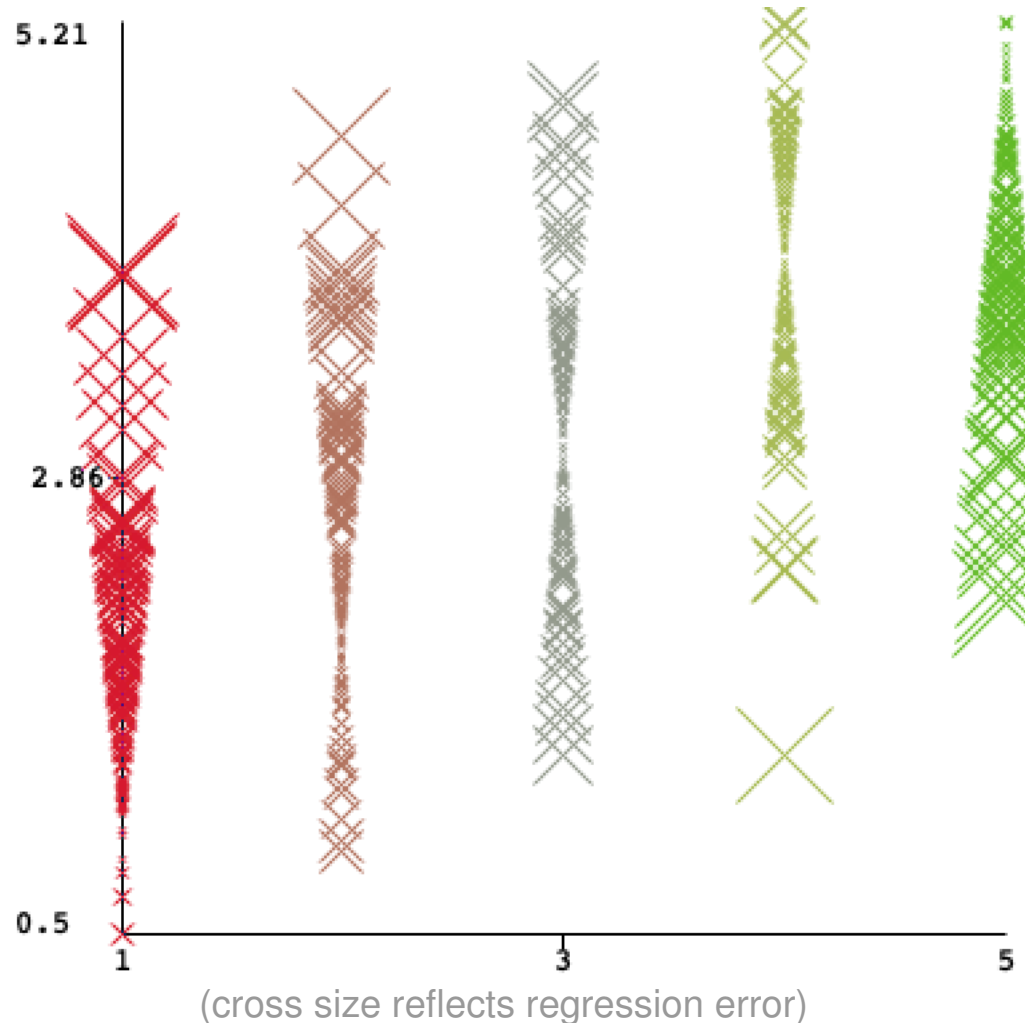
### Visualization

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

### Observations

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

Typical for regression

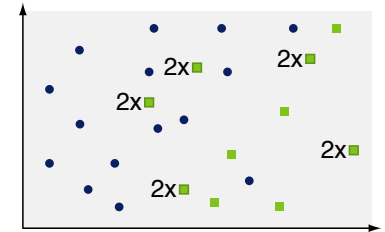


# Conclusion

# Conclusion

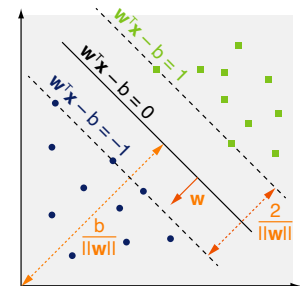
## NLP using classification and regression

- Assessment and quantification of text properties
- Usually done with supervised learning
- Datasets may have to be prepared for learning.



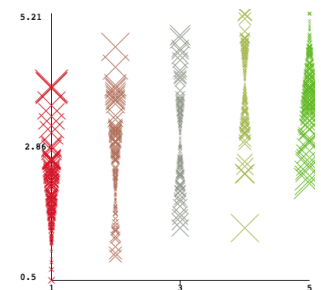
## Supervised classification

- 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

- **Manning et al. (2008)**. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). Introduction to Information Retrieval. Cambridge University Press.
- **Ng (2018)**. Andrew Ng. Machine Learning. Lecture slides from the Stanford Coursera course, 2018. <https://www.coursera.org/learn/machine-learning>
- **Wachsmuth (2015)**. Henning Wachsmuth (2015): Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining. LNCS 9383, Springer.
- **Witten and Frank (2005)**. Ian H. Witten and Eibe Frank (2005): Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition.

## Other references

- **Wachsmuth et al. (2014a)**. Henning Wachsmuth, Martin Trenkman, Benno Stein, Gregor Engels, and Tsvetomira Palarkarska. A Review Corpus for Argumentation Analysis. In Proceedings of the of the 15th International Conference on Intelligent Text Processing and Computational Linguistics, pages 115–127, 2014.
- **Wachsmuth et al. (2014b)**. Henning Wachsmuth, Martin Trenkman, Benno Stein, and Gregor Engels. Modeling Review Argumentation for Robust Sentiment Analysis. In Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers, pages 553–564, 2014.