Supervised Learning for NLP I

HSS 510: NLP for HSS

Taegyoon Kim

Mar 27, 2024

Agenda

Things to be covered

- Overview of supervised machine learning
- Step 1: Building a labeled data set
- Step 2: Extracting features
- Step 3: Selecting and training model(s)
- Step 4: Evaluating performance
- Guided labeling: text classification with movie review data in Python

Supervised Learning

We will focus on (text) classification with supervised learning

- Goal
  - To classify documents into pre-defined categories
  - E.g., sentiment of comments, stance on policy issues, topic of news articles, etc.
- We need
  - Human-labeled data set
  - Model (algorithm) that maps documents (i.e., their features) to labels
  - Evaluation approaches
    - Performance metrics, cross-validation, etc.

Supervised Learning

Supervised vs. Unsupervised

	Supervised	**Unsupervised**
Objective	Trained on a labeled data to learn a mapping from input to output	Find patterns or structures within data without labels
Outcome	Pre-defined categories	Not quite pre-defined
Model evaluation	Explicit metrics such as accuracy, precision, recall, or MSE	Can involve qualitative assessment
Examples	Classification/regression for texts	Topic models

Supervised Learning

Regression vs. Classification

- Regression
  - The outcome of interest is continuous or ordered (beyond binary)
  - E.g., OLS regression (+ non-linear regression algorithms such as random forest regression)
- Classification
  - The outcome is a value in an unordered set (i.e. categories)
- The two approaches share the broad principles of supervise leaning and can be adapted

Supervised Learning vs. Dictionary Methods

Limitations of dictionary methods

- Lack of learning (as in the the name machine “learning”)

- (Largely) ignores context
  - Polysemy, co-occurrences/interactions, etc.
  - Interactions are effectively modeled in random forest, deep neural networks, and large language models

→ Therefore, (generally) suboptimal performance

Overview of Process

Broad process

- Step 1: build a labeled data set
- Step 2: extract features
- Step 3: select and train model(s)
- Step 4: evaluate performance

Overview of Process

Step 1: build a labeled data set

- Documents with human-annotated labels (a.k.a. ground-truth) : C
- Randomly split into a training set and a test set: $C = C_{train} + C_{test}$
- E.g., identifying YouTube comments containing hate speech
  - C: 10,000 comments labeled for the presence of hate speech
  - $C_{training}$: 8,000 comments for training
  - C_{test} : 2,000 comments for test

Overview of Process

Step 1: build a labeled data set

Doc number	**Text**	**y**
1	This is great!	0
2	%@% ***k off!	1
…		
9999	This is sick	0
10000	Love BTS <3	0

Overview of Process

Step 2: extract features

- Generate X_{train} (feature matrix) from C_{train} (train set)
- E.g., count vectors, TF-IDF, or embeddings

Index	Token 1	Token 2	…	Token V-1	Token V
1	3	1.4	…	1.7	6
2	-0.8	6.4	…	5.7	-1.6
…					
7999	-2.8	0.9	…	3.3	-0.6
8000	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 3: select/train model(s)

Index	y	Token 1	Token 2	…	Token V-1	Token V
1	0	3	1.4	…	1.7	6
2	1	-0.8	6.4	…	5.7	-1.6
…						
7999	0	-2.8	0.9	…	3.3	-0.6
8000	0	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 3: select/train model(s)

- Choose a model F (e.g., logistic regression) and learn model parameters β (e.g., an array of coefficients)
  - The model provides a mapping between X_{train} and y_{train}
- Loss (cost) function: measures how much model predictions ($\hat{y}_{train}$) differ from the true labels (y_{train})
  - β is estimated in a way that minimizes the difference
  - $\hat{y}_{train} = F(\hat{\beta} * X_{train})$
- As a result, we get a classifier: $\hat{y} = F(\hat{\beta} * X)$

Overview of Process

Step 3: select/train model(s)

Index	y	$\hat{y}$	Token 1	Token 2	…	Token V-1	Token V
1	0	0	3	1.4	…	1.7	6
2	1	1	-0.8	6.4	…	5.7	-1.6
…							
7999	0	0	-2.8	0.9	…	3.3	-0.6
8000	0	0	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 4: evaluate performance

- We held out another labeled set C_{test} (n = 2,000) (**why?**)
- Use the classifier $F(\hat{\beta} * X)$ to generate predictions $\hat{y}_{test}$
- Compare the predictions $\hat{y}_{test}$ and the true labels y_{test}
- Performance metrics include accuracy, precision, recall, etc.
- *(Then use the classifier for unlabeled data)*

Overview of Process

Step 4: evaluate performance

Index	**y**	$\hat{y}$	**Token 1**	**Token 2**	**…**	**Token V-1**	**Token V**
1	1	1	3.12	1.99	…	5.77	0.36
2	1	0	-0.8	1.14	…	9.71	-1.66
…							
1999	0	0	-2.11	0.95	…	1.23	-0.62
2000	0	0	3.71	1.48	…	1.7	-5.84

Bias, Variance, and Overfitting

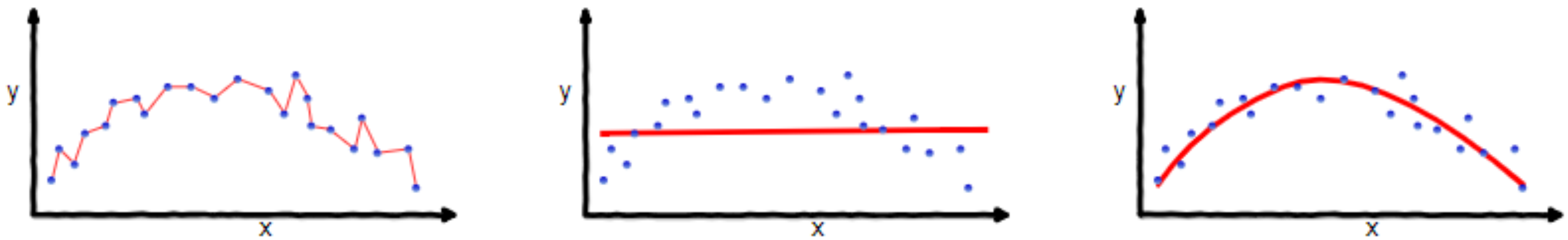
Bias

- The degree to which the model's predictions deviate from the true labels in a systematic manner
- A model with high bias make predictions that are consistently off-target

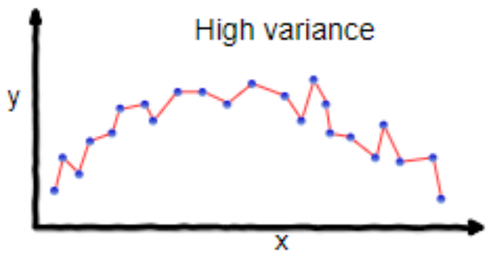
Variance

- The degree to which the model generalizes to different data
- High variance means low generalizability

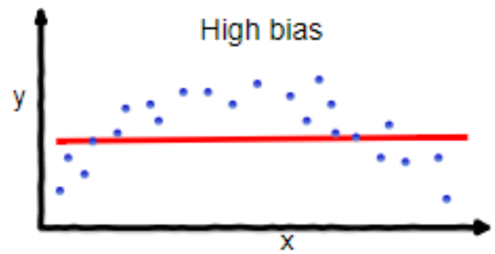
Bias, Variance, and Overfitting



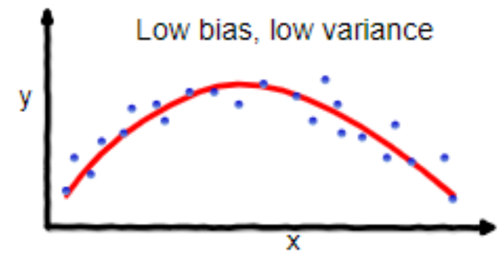
Bias, Variance, and Overfitting



overfitting



underfitting

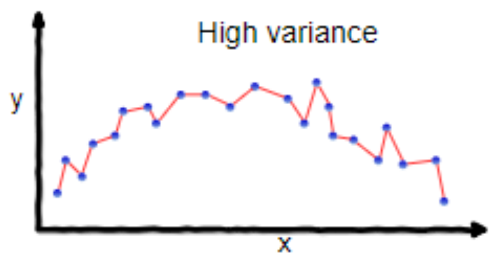


Good balance

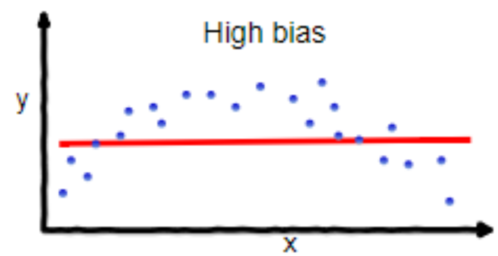
Bias, Variance, and Overfitting

Overfitting and underfitting

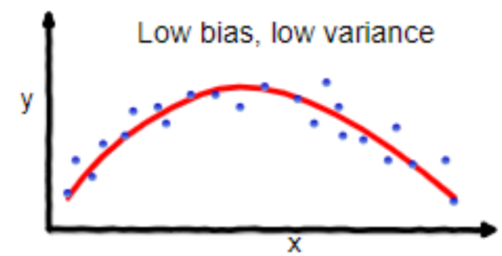
- If a model learns the training data “too well” (low bias), it can lead to overfit
- This happens when the model mistakes noise for signal
- The model would not generalize to the test set (high variance)



overfitting



underfitting



Good balance

Bias, Variance, and Overfitting

Training-test split

- A minimal measure to prevent overfitting
- The primary goal here is to make our model as generalizable as possible
- “Generalizable” means being able to perform well on unseen documents (other than the documents the model was trained on)
- When a model learns the noise or random fluctuations in the training set, this typically results in a model that performs poorly on new the test set

Prediction and Explanation

How are they different?

- Predictive modeling emphasizes predictive performance and generalizability (i.e., out-of-sample prediction)
- Explanatory modeling emphasizes hypothesis the statistical significance of an individual coefficient(s) associated with a (relational) hypothesis
- See Shmueli (2010) for a detailed treatment of the differences between prediction and explanation

Step 1: Building a Labeled Data Set

How do we obtain a labeled data set?

- A form of manual content analysis
- “Ground truth” or “gold standard” fed to machines
- Our decisions/labels reflect latent features linked to the categories (some of which we are unconscious of)
- Manually labeled data are used for training (train set) and evaluation (test set)

Step 1: Building a Labeled Data Set

How do we obtain a labeled data set?

- Expert labeling
  - In many projects, a few domain experts work on a labeling (after training)
  - Annotators are trained to learn the concept and related guidelines
  - E.g., a researcher + two RAs from the department
- Crowd-sourced labeling
  - "Wisdom of crowd": aggregated judgments of (online) non-experts converge to judgments of experts at much lower cost (Benoit et al, 2016)
  - Difficult to educate annotators on sophisticated tasks
  - Inductive measurement based on loose conceptualization

Step 1: Building a Labeled Data Set

Expert labeling vs. Crowd Sourcing

- Deductive vs. inductive
- Degree of training
- Scalability (cost)

Step 1: Building a Labeled Data Set

Selected texts for manual labeling

- Should reflect the entire corpus
- Mismatch leading to low performance: shift/drift
- E.g., drift in anti-vaccine discourse throughout 2020

Step 1: Building a Labeled Data Set

Iterative process

- Definition/operationalization does not often take place at once but in an iterative process
- In many cases, it is difficult to specify an entire annotation guidelines ex ante
- Preliminary labeling rule are written and applied to an initial set of docs
  → Annotators identify ambiguities in the rule
  → Revision of the rule → …

Step 1: Building a Labeled Data Set

Dealing with subjectivity

- Many concepts in humanities and social sciences are not straightforward
- They can involve high levels of subjectivity
- This is, from the beginning, why 1) careful conceptualization and 2) writing an excellent labeling rule, and 3) training coders are extremely important
- **Evaluation metrics**: Krippendorf's α, Cohen's κ (alternatives include Pearson's **r**, Spearman's ρ) (recommended R package: **irr**)

Step 1: Building a Labeled Data Set

Who are the annotators?

- Expert coding
  - Academics/students (Javdani and Chang 2023)
- Crowdsourcing
  - Skewed distribution of worked hours (Difallah et al. 2018)
  - Inattentive workers (Peyton et al. 2022; Ternovski 2022)
  - LLM-based responses (Veselovsky et al. 2023)
  - Demographic characteristic (Al Kuwatly et al. 2020)

Step 2: Extract Features

$$C_{train} \rightarrow \quad X_{train}$$

- Options include count vectors, TF-IDF vectors, word/document embeddings, etc.

Index	Token 1	Token 2	…	Token V-1	Token V
1	3	1.4	…	1.7	6
2	-0.8	6.4	…	5.7	-1.6
…					
7999	-2.8	0.9	…	3.3	-0.6
8000	3.7	1.4	…	5.7	-5.8

Step 3: Select/train Model(s)

So far we have:

- Built a labeled data set (Step 1)
- Generated a feature matrix (Step 2)
- This means that we have the outcome (y) and features (X_{train})

Now we will:

- Select a model F
- Learn model parameters β to build a classifier ($\hat{y} = F(\hat{\beta} * X)$)

Step 3: Select/train Model(s)

Numerous algorithms

- Logistic regression
- Naive Bayes
- Support vector machine
- Tree-based models (decision tree, random forest, XGBoost, etc.)
- Neural networks
- Etc.

Step 3: Select/train Model(s)

Logistic regression

- Used to classify a document into binary categories
  - Multinomial logistic regression for more than two
- One of the most useful analytics tools in science (not just NLP/ML)
- The baseline supervised learning algorithm for classification
- Forms the basis of neural networks

Step 3: Select/train Model(s)

Components of logistic regression (j documents n features)

- Features (e.g., tokens)
  - A document is represented as a vector of features $\vec{x}$ (= $[x_1, \ldots, x_n]$)
- A classification function (F)
  - $p(y = 1 \; x)$ is computed for each document given the feature vector and β
- Loss function and algorithm for optimizing it (gradient descent)

Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

- $p(y = 1 \; x)$
  - We want to know the probability of y = 1 given a feature vector $\vec{x}$ (= $[x_1 , \ldots, x_n]$)
  - For a simple count vector, it would be the number times each token appears in the document
- Logistic regression learns β, a vector of coefficients
  - A bias term b: a single number (a.k.a. intercept)
  - Weights $\vec{w}$ (= $[w_1 , \ldots, w_n]$)
    - E.g., features signaling hateful intention would get high weights
  - With b, $\vec{w}$, and $\vec{x}$, we compute z (= ($\sum_{i=1}^{n} w_i x_i$) + b)

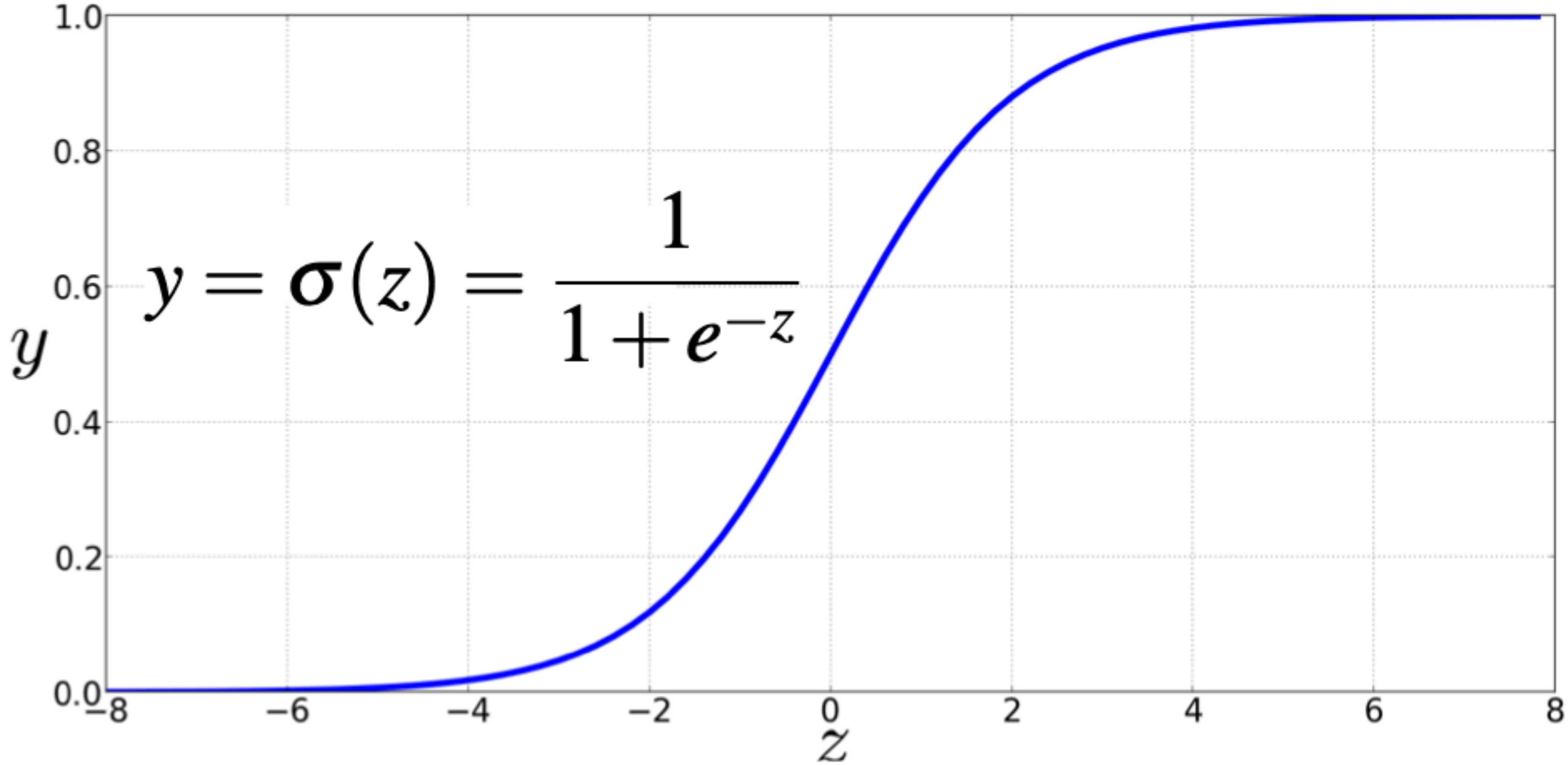
Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

- z = ($\sum_{i=1}^{n} w_i x_i$) + b = $\vec{w} \cdot \vec{v}$ + b

- σ (z) = $\frac{1}{1+e^{-z}}$ = $\frac{1}{1+exp(-z)}$

Step 3: Select/train Model(s)



Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

$$p(y = 1 \;\; x) = \sigma(\vec{w} \cdot \vec{x} + b)$$

$$p(y = 0 \;\; x) = 1 - \sigma(\vec{w} \cdot \vec{x} + b)$$

Step 3: Select/train Model(s)

How do predicted probabilities turn into binary labels ($\hat{y}$)?

$$\begin{cases} 1 & \text{if } P(y = 1 \ x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Step 3: Select/train Model(s)

Sentiment classification from movie reviews

"It's hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great, Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing? It sucked me in, and it'll do the same to you."

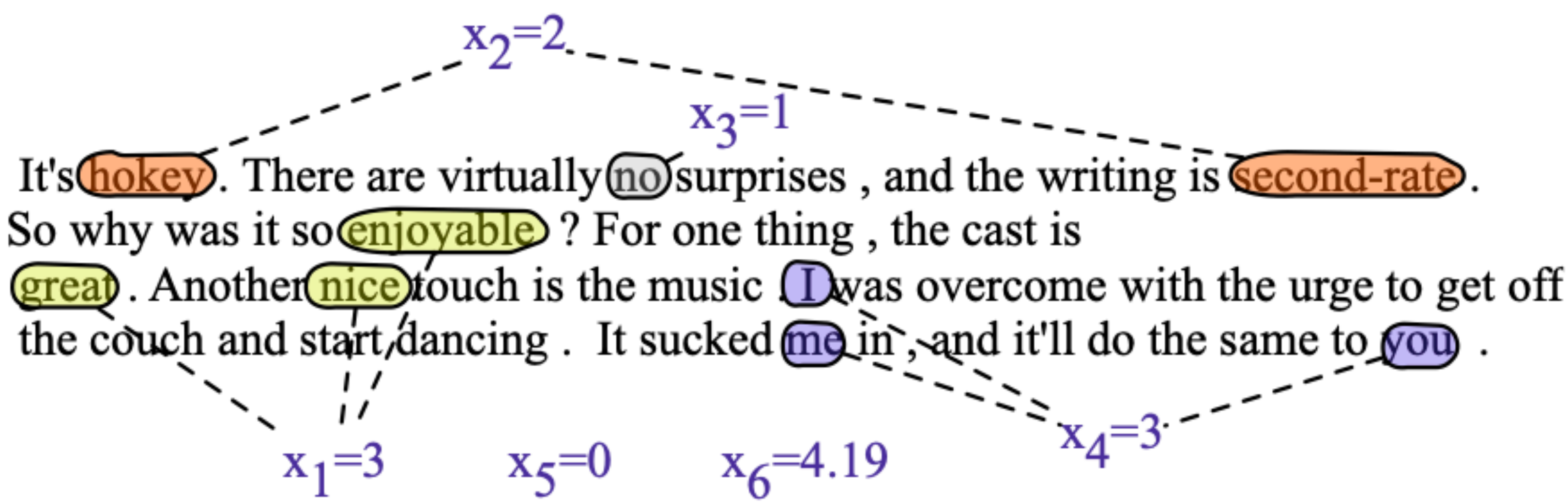
Step 3: Select/train Model(s)

Sentiment classification from movie reviews

Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Step 3: Select/train Model(s)

Sentiment classification from movie reviews



Step 3: Select/train Model(s)

Positive

- $p(+\ x) = p(y = 1\ x) = \sigma(\vec{w} \cdot \vec{x} + b)$
  $= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3,2,1,3,0,4.19] + 0.1)$
  $= \sigma(.833)$
  $= 0.70$

Negative

- $p(-\ x) = p(Y = 0\ x) = 1 - \sigma(\vec{w} \cdot \vec{x} + \mathrm{b})$
  $= 0.30$

Step 3: Select/train Model(s)

Learning coefficients

- MLE (Maximum Likelihood Estimation)
- Loss function / cross entropy
- Gradient descent

Step 4: Evaluate Performance

We have

- Manually labeled documents
- Split them into C_{train} (training set) and C_{test} (test set)
- Trained a classifier on C_{train} (with y_{train} and X_{train}) $\rightarrow F(\hat{\beta}^* X)$

Now we need to evaluate its performance on C_{test}

- We compare $\hat{y}_{test}$ (predicted labels) against y_{test} (true labels)

Step 4: Evaluate Performance

Performance metrics

- Accuracy: the proportion of all predictions (both positive and negative) that the model got right
- Precision: the proportion of positive predictions that were actually correct
- Recall: the proportion of actual positives that were correctly predicted
- F-1: the harmonic (as opposed to arithmetic) mean of precision and recall

Step 4: Evaluate Performance

Confusion matrix: predictions against true labels

		True condition	
		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Step 4: Evaluate Performance

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$

		True condition	
		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Step 4: Evaluate Performance

Precision: $\frac{TP}{TP+FP}$

		True condition	
		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Step 4: Evaluate Performance

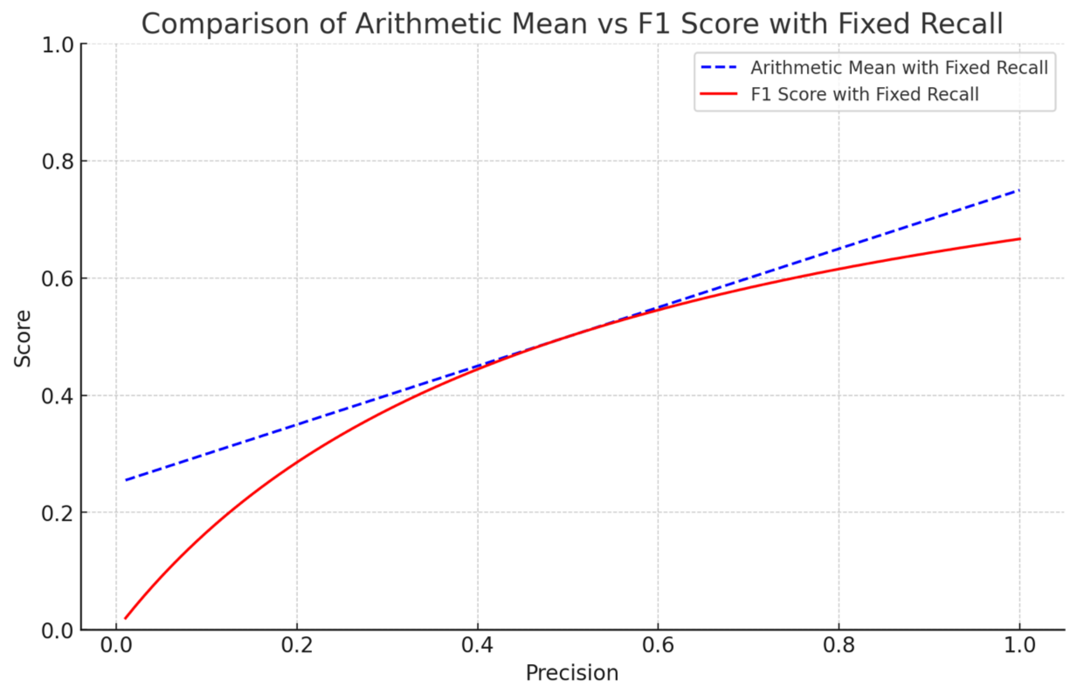
Recall: $\frac{TP}{TP+FN}$

		True condition	
		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Step 4: Evaluate Performance

F-1: $(2 \times precision \times recall)$ / $(precision + recall)$

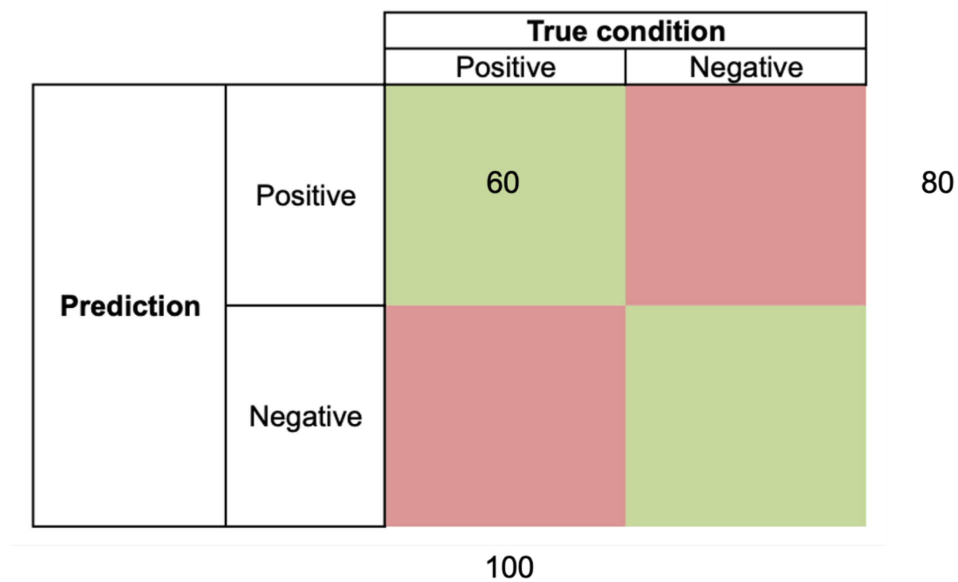
- Why not arithmetic mean ($(precision + recall)/2$)?



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

- 100 positives
- 80 predicted positives
- 60 true positives



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

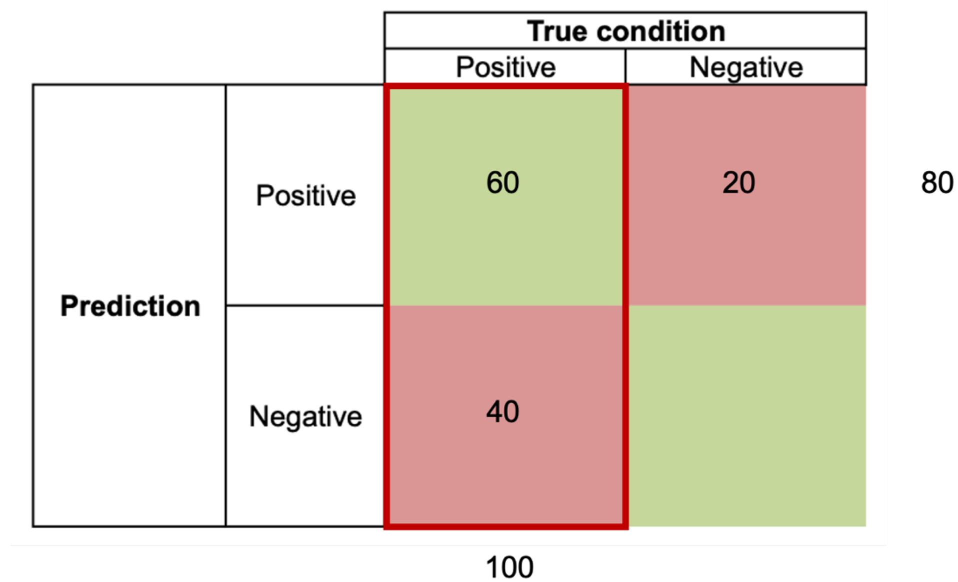
- Precision: $\frac{60}{60+20} = 0.75$

		True condition		
		Positive	Negative	
Prediction	Positive	60	20	80
	Negative			
		100		

Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

- Recall: $\frac{60}{60+40} = 0.6$



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

- Precision: $\frac{60}{60+20} = 0.75$
- Recall: $\frac{60}{60+40} = 0.6$

- Accuracy: $\frac{60+50}{60+20+40+50} = 0.65$

		True condition		
		Positive	Negative	
Prediction	Positive	60	20	80
	Negative	40	*50*	
		100		

Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

- Precision: $\frac{60}{60+20} = 0.75$
- Recall: $\frac{60}{60+40} = 0.6$

- Accuracy: $\frac{60+150}{60+20+40+150} = 0.78$

		True condition		
		Positive	Negative	
Prediction	Positive	60	20	80
	Negative	40	***150***	
		100		

Step 4: Evaluate Performance

An extremely imbalanced case

- Accuracy: ??
- Precision: ??
- Recall: ??
- F-1: ??

		True condition		
		Positive	Negative	
Prediction	Positive	2	1	3
	Negative	8	989	997
		10	990	100

Step 4: Evaluate Performance

An extremely imbalanced case

- Accuracy: 0.991
- Precision: 0.66
- Recall: 0.2
- F-1: 0.31

		True condition		
		Positive	Negative	
Prediction	Positive	2	1	3
	Negative	8	989	997
		10	990	100

Step 4: Evaluate Performance

Reminders

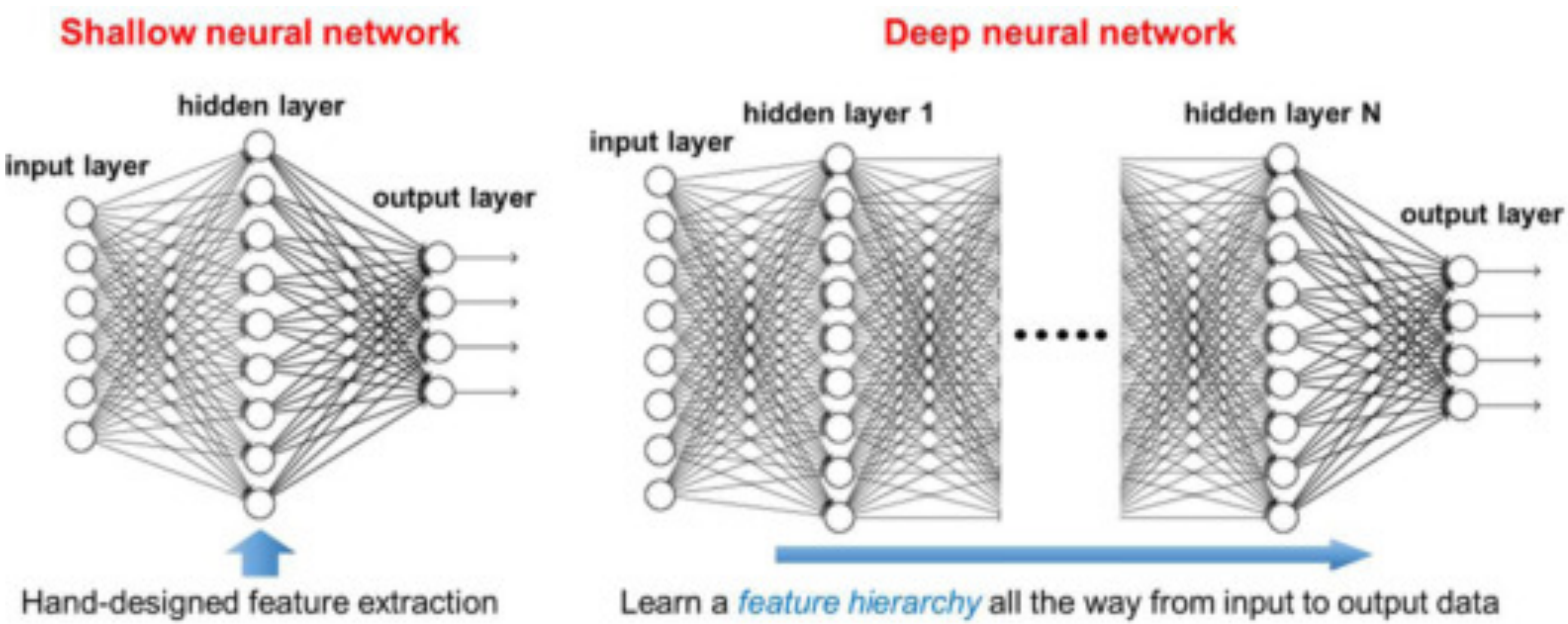
- Random train-test split: C_{train} , C_{test}
  - E.g., 10,000 comments labeled for hate speech into 8,000 and 1,000
- Our classifier learned **parameters**, maximizing performance on C_{train} and evaluating it on C_{test}

Step 4: Evaluate Performance

Parameters vs. hyper-parameters

- Parameter
  - Learned (estimated) from data (internal to the model)
  - E.g., logistic regression weights/coefficients (= β)
- Hyper-parameters
  - Defines the model structure itself (not internal to the model)
  - E.g., the size of a regularization term in logistic regression, the number of layers or learning rate in neural networks, etc.

Step 4: Evaluate Performance



Step 4: Evaluate Performance

Hyper-parameters influence model performance, and we want to “tune” them

- With different hyper-parameter values, we could fit a model configured with each value on C_{train} and evaluate performance on C_{test}
- E.g., regularization strength λ in logistic regression
  - Train different models with different λ values on C_{train}
  - Evaluate on C_{test}
  - Pick the best performing model

Step 4: Evaluate Performance

What could go wrong?

- In comparing different models (different hyperparaemter values), we might overfit on C_{train}
- By repeatedly using the trainig set, our comparison can be affected by the specific characteristics of the test set

Step 4: Evaluate Performance

Validation set

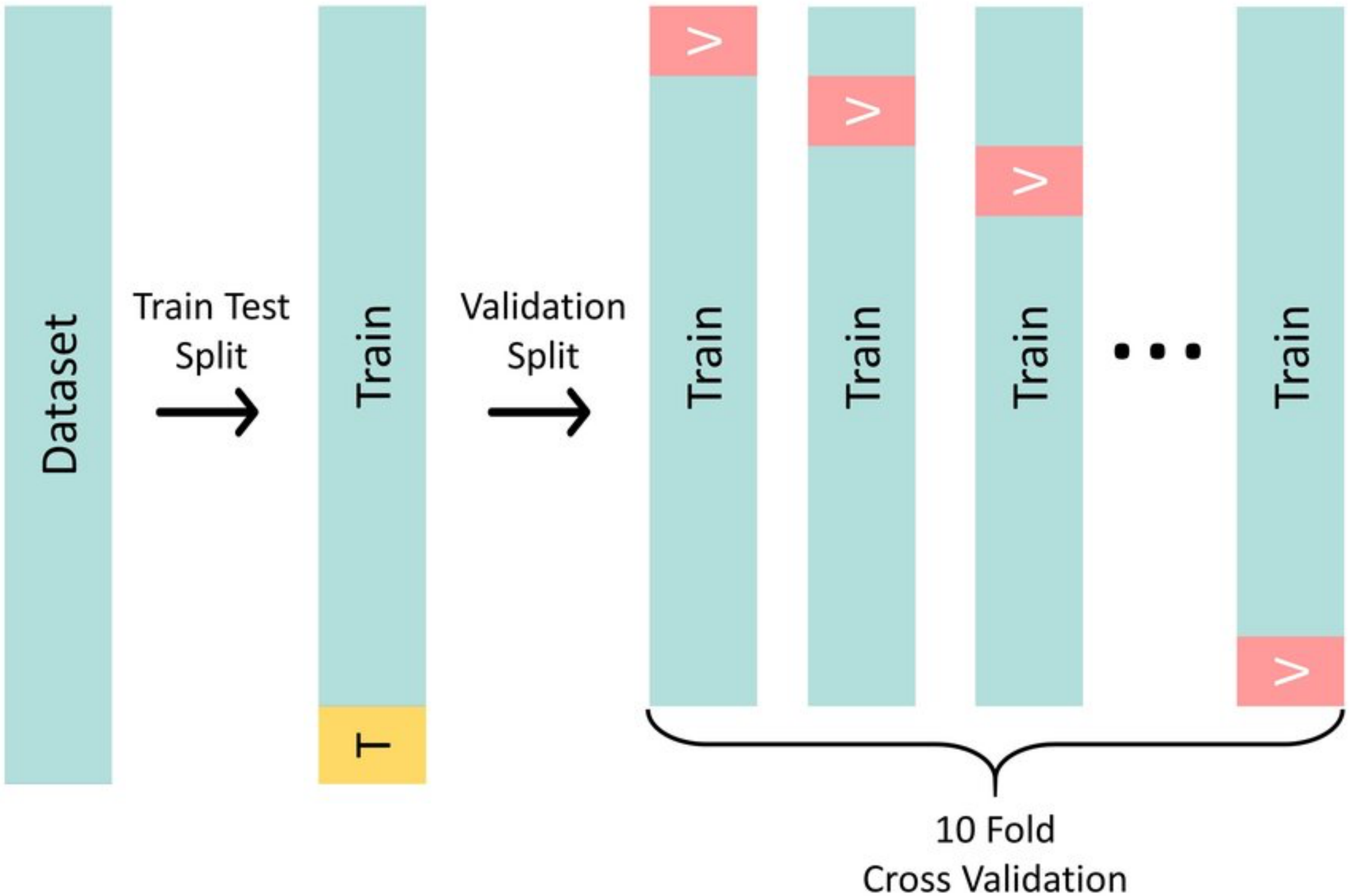
- We split the labeled data set into a training set, a “validation set”, and a test set: C_{train} , $C_{validation}$, and C_{test}
- Train a model on C_{train} and see how it performs on $C_{validation}$
- Repeat this step for multiple configurations (e.g., hyper-parameters)
- Pick the best-performing configuration
- Train a final model based on the configuration on C_{train} + $C_{validation}$
- Evaluate on C_{test}

Step 4: Evaluate Performance

K-fold cross-validation

- Randomly split C_{train} into K equal parts or “folds” (commony 5 or 10)
- For each iteration
  - Treat one fold as the “validation set”
  - Train your model on the remaining $K - 1$ folds
  - Evaluate performance on the validation set kept aside
- After cyclig through all iterations
  - Aggregate the performance metrics obtained from each iteration
  - Choose the classifier with the highest cross-validated performance
  - This step may invovle not just hyperparameter tuning but also things like feature repesentation, etc.
- (Re)train the chosen best classifier on C_{train} (all K folds combined) and evalute on C_{test}

Step 4: Evaluate Performance



Summary

Supervised text classification provides a highly useful tool to assign labels to documents

- Be aware of the principles of building a labeled data set
  - Conceptualization, intercoder reliability, annotator bias, etc.
- Validate, validate, and validate!
- Choose appropriate evaluation metrics

Guided Coding

Text classification with movie reviews data (link)