Machine Learning in Spam Filtering

A Crash Course in ML

Konstantin Tretyakov

kt@ut.ee

Institute of Computer Science, University of Tartu

Overview

- Spam is Evil
- ML for Spam Filtering: General Idea, Problems.
- Some Algorithms
  - Naïve Bayesian Classifier
  - k-Nearest Neighbors Classifier
  - The Perceptron
  - Support Vector Machine
- Algorithms in Practice
  - Measures and Numbers
  - Improvement ideas: Striving for the ideal filter

Spam is Evil

- It is cheap to send, but expensive to receive:
  - Large amount of bulk traffic between servers
  - Dial-up users spend bandwidth to download it
  - People spend time sorting thru unwanted mail
- Important e-mails may get deleted by mistake
- Pornographic spam is not meant for everyone

Eliminating Spam

- Social and political solutions
  - Never send spam
  - Never respond to spam
  - Put all spammers to jail
- Technical solutions
  - Block spammer's IP address
  - Require authorization for sending e-mails (?)
- Mail filtering
  - Knowledge engineering (KE)
  - Machine learning (ML)

Knowledge Engineering

- Create a set of classification rules by hand:
  - "if the `Subject` of a message contains the text `BUY NOW`, then the message is spam"
  - `procmail`
  - "Message Rules" in Outlook, etc.
- The set of rules is difficult to maintain
- Possible solution: maintain it in a centralized manner
  - Then spammer has access to the rules

Machine Learning

- Classification rules are *derived* from a set of *training samples*
- For example:

  Training samples

```
Subject: "BUY NOW"     -> SPAM
Subject: "BUY IT"      -> SPAM
Subject: "A GOOD BUY"  -> SPAM
Subject: "A GOOD BOY"  -> LEGITIMATE
Subject: "A GOOD DAY"  -> LEGITIMATE
```

  Derived rule

```
Subject contains "BUY" -> SPAM
```

Machine Learning

- A training set is required. It is to be updated regularly.
- Hard to guarantee that no misclassifications occur.
- No need to manage and understand the rules.

Machine Learning

- Training set:

$$\{(\mathbf{m}_1, c_1), (\mathbf{m}_2, c_2), \ldots, (\mathbf{m}_n, c_n)\}$$

- $\mathbf{m}_i \in \mathbb{M}$ are training messages, a class $c_i \in \{S, L\}$ is assigned to each message.
- Using the training set we construct a classification function

$$f : \mathbb{M} \rightarrow \{S, L\}$$

- We use this function afterwards to classify (unseen) messages.

ML for Spam: Problem 1

- Problem: We classify text but most classification algorithms either
  - require numerical data ($\mathbb{R}^n$)
  - require a distance metric between objects
  - require a scalar product

ML for Spam: Problem 1

- Problem: We classify text but most classification algorithms either
  - require numerical data ($\mathbb{R}^n$)
  - require a distance metric between objects
  - require a scalar product
- Solution: use a feature extractor to convert messages to vectors:

$$\phi : \mathbb{M} \longrightarrow \mathbb{R}^n$$

ML for Spam: Problem 2

- Problem: A spam filter may not make mistakes
  - False positive: a legitimate mail classified as spam
  - False negative: spam classified as legitimate mail
  - False negatives are ok, false positives are *very bad*
- Solution: ?

Algorithms: Naive Bayes

- The Bayes' rule:

$$P(c\,|\,\mathbf{x}) = \frac{P(\mathbf{x}\,|\,c)P(c)}{P(\mathbf{x})} = \frac{P(\mathbf{x}\,|\,c)P(c)}{P(\mathbf{x}\,|\,S)P(S) + P(\mathbf{x}\,|\,L)P(L)}$$

Algorithms: Naive Bayes

- The Bayes' rule:

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c)P(c)}{P(\mathbf{x})} = \frac{P(\mathbf{x} \mid c)P(c)}{P(\mathbf{x} \mid S)P(S) + P(\mathbf{x} \mid L)P(L)}$$

- Classification rule:

$$P(S \mid \mathbf{x}) > P(L \mid \mathbf{x}) \Rightarrow \text{SPAM}$$

Algorithms: Naive Bayes

- Bayesian classifier is *optimal*, i.e. its average error rate is minimal over all possible classifiers.
- The problem is, we can never know the exact probabilities in practice.

Algorithms: Naive Bayes

- How to calculate $P(\mathbf{x} \mid c)$?

Algorithms: Naive Bayes

- How to calculate $P(\mathbf{x} \,|\, c)$?
- It is simple if the feature vector is simple:
  Let the feature vector consist of a single binary attribute x_w. Let $x_w = 1$ if a certain word w is present in the message and $x_w = 0$ otherwise.

Algorithms: Naive Bayes

- How to calculate $P(\mathbf{x} \,|\, c)$?
- It is simple if the feature vector is simple:
  Let the feature vector consist of a single binary attribute x_w. Let $x_w = 1$ if a certain word w is present in the message and $x_w = 0$ otherwise.
- We may use more complex feature vectors if we assume that presence of one word does not influence the probability of presence of other words, i.e.

$$P(x_w, x_v \,|\, c) = P(x_w \,|\, c)P(x_v \,|\, c)$$

Algorithms: k-NN

- Suppose we have a distance metric d defined for messages.
- To determine the class of a certain message $\mathbf{m}$ we find its k nearest neighbors in the training set.
- If there are more spam messages among the neighbors, classify $\mathbf{m}$ as spam, otherwise as legitimate mail.

Algorithms: k-NN

- k-NN is one of the few *universally consistent* classification rules.
- Theorem (Stone): as the size of the training set n goes to infinity, if $k \to \infty$, $\frac{k}{n} \to 0$, then the average error of the k-NN classifier approaches its minimal possible value.

Algorithms: The Perceptron

- The idea is to find a linear function of the feature vector $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ such that $f(\mathbf{x}) > 0$ for vectors of one class, and $f(\mathbf{x}) < 0$ for vectors of other class.
- $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ is the vector of coefficients *(weights)* of the function, and b is the so-called *bias*.
- If we denote the classes by numbers $+1$ and -1, we can state that we search for a decision function

$$d(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Algorithms: The Perceptron

- Start with arbitrarily chosen parameters $(\mathbf{w}_0, b_0)$ and update them iteratively.
- On the n-th iteration of the algorithm choose a training sample $(\mathbf{x}, c)$ such that the current decision function does not classify it correctly (i.e. $\mathrm{sign}(\mathbf{w}_n^T \mathbf{x} + b_n) \neq c$).
- Update the parameters $(\mathbf{w}_n, b_n)$ using the rule:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + c\mathbf{x} \qquad b_{n+1} = b_n + c$$

Algorithms: The Perceptron

- Start with arbitrarily chosen parameters $(\mathbf{w}_0, b_0)$ and update them iteratively.
- On the n-th iteration of the algorithm choose a training sample $(\mathbf{x}, c)$ such that the current decision function does not classify it correctly (i.e. $\operatorname{sign}(\mathbf{w}_n^T \mathbf{x} + b_n) \neq c$).
- Update the parameters $(\mathbf{w}_n, b_n)$ using the rule:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + c\mathbf{x} \qquad b_{n+1} = b_n + c$$

- The procedure stops someday if the training samples were *linearly separable*

Algorithms: The Perceptron

- Fast and simple.
- Easy to implement.
- Requires linearly separable data.

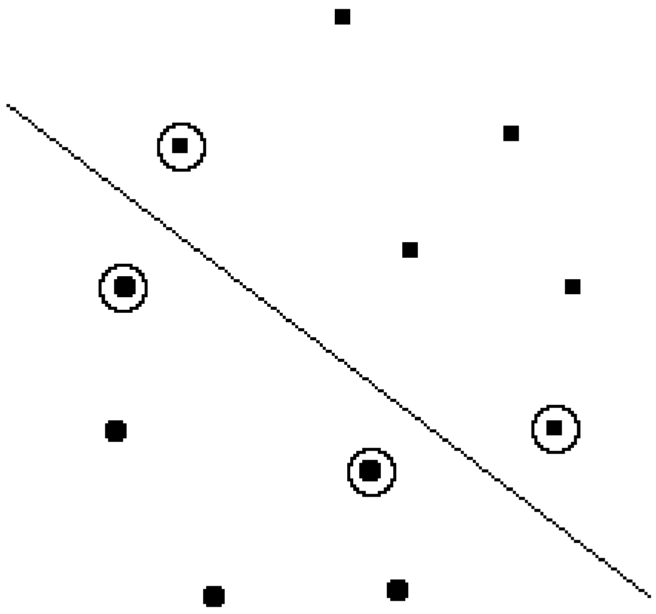
Algorithms: SVM

- The same idea as in the case of the Perceptron: find a *separating hyperplane*

$$\mathbf{w}^T\mathbf{x} + b = 0$$

- This time we are not interested in *any* separating hyperplane, but the *maximal margin* separating hyperplane.

Algorithms: SVM



Maximal margin separating hyperplane

Algorithms: SVM

- Finding the optimal hyperplane requires minimizing a quadratic function on a convex domain — a task known as *a quadratic programme*.

Algorithms: SVM

- Finding the optimal hyperplane requires minimizing a quadratic function on a convex domain — a task known as *a quadratic programme*.
- *Statistical Learning Theory* by V. Vapnik guarantees good generalization for SVM-s.

Algorithms: SVM

- Finding the optimal hyperplane requires minimizing a quadratic function on a convex domain — a task known as *a quadratic programme*.
- *Statistical Learning Theory* by V. Vapnik guarantees good generalization for SVM-s.
- There are lots of further options for SVM-s (soft margin classification, nonlinear kernels, regression).

Algorithms: SVM

- Finding the optimal hyperplane requires minimizing a quadratic function on a convex domain — a task known as *a quadratic programme*.
- *Statistical Learning Theory* by V. Vapnik guarantees good generalization for SVM-s.
- There are lots of further options for SVM-s (soft margin classification, nonlinear kernels, regression).
- SVM-s are one of the most widely used ML classification techniques currently.

Practice: Measures

- Denote by $N_{S\to L}$ the number of false negatives, and by $N_{L\to S}$ number of false positives. The quantities of interest are then the *error rate* and *precision*

$$E = \frac{N_{S\to L} + N_{L\to S}}{N}, \qquad P = 1 - E$$

legitimate mail fallout and *spam fallout*

$$F_L = \frac{N_{L\to S}}{N_L}, \qquad F_S = \frac{N_{S\to L}}{N_S}$$

- Note that the error rate and precision must be considered *relatively to the case of no classifier*.

Practice: Numbers

Algorithm	$N_{L \to S}$	$N_{S \to L}$	P	F_L	F_S
Naïve Bayes	0	138	87.4%	0.0%	28.7%
k-NN	68	33	90.8%	11.0%	6.9%
Perceptron	8	8	98.5%	1.3%	1.7%
SVM	10	11	98.1%	1.6%	2.3%

Results of 10-fold cross-validation on PU1 spam corpus

Eliminating False Positives

Algorithm	$N_{L \to S}$	$N_{S \to L}$	P	F_L	F_S
Naïve Bayes	0	140	87.3%	0.0%	29.1%
l/k-NN	0	337	69.3%	0.0%	70.0%
SVM soft margin	0	101	90.8%	0.0%	21.0%

Results after tuning the parameters to eliminate false positives

Combining Classifiers

- If we have two different classifiers f and g that have low probability of false positives, we may combine them to get a classifier with higher precision:

  Classify message $\mathbf{m}$ as spam, if f or g classifies it as spam.

- Denote the resulting classifier as $f \cup g$

Algorithm	$N_{L \to S}$	$N_{S \to L}$	P	F_L	F_S
N.B. $\cup$ SVM s. m.	0	61	94.4%	0.0%	12.7%

Combining Classifiers

- If we add to f and g another classifier h with high precision, we may use it to make $f \cup g$ even safer:

  If $f(\mathbf{m}) = g(\mathbf{m})$, classify message $\mathbf{m}$ as $f(\mathbf{m})$, otherwise (if f and g give different answers) consult h (instead of blindly setting $\mathbf{m}$ as spam).

  In other words: classify $\mathbf{m}$ to the class, which is proposed by at least 2 of the three classifiers.

- Denote the classifier as $(f \cap g) \cup (g \cap h) \cup (f \cap h)$.

Algorithm	$N_{L \to S}$	$N_{S \to L}$	P	F_L	F_S
2-of-3	0	62	94.4%	0.0%	12.9%

Questions

?