Incorporating Invariances in Support Vector Machines using Max-Pooling Kernels

Tuhin Das
Delft University of Technology
t.das@student.tudelft.nl

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

1. Introduction

For certain tasks like digit recognition, image classifiers often benefit from invariances to image translations and rotations, as object appearance is usually more important than object location. Some classifiers such as the Convolutional Neural Network (CNN) are invariant to small image transformations to some extent [1]. Many classifiers are however not invariant, an example of which is the Support Vector Machine (SVM). The classification capabilities of Support Vector Machine can be improved by incorporating invariances to image transformation.

Various methods exist to incorporate invariances in SVMs [8], but many methods either rely on data augmentation or manually defining which transformations a SVM should become invariant to. In this paper we present a new method to make SVMs more invariant, where data augmentation and manually defining transformations are not necessary. Inspired by CNNs, which are partially invariant due to their max-pooling filters [3, 6], we attempt to incorporate max-pooling filters in SVMs. We modify Radial Basis Function (RBF) kernels of SVMs to use max-pooling filters, such that the kernels become invariant to image transformations. The kernels should denote more similarities between different samples of a similar class, even though the images are translated or rotated compared to each other.

We make the following contributions. First we show that max-pooling filters can be incorporated in Gaussian RBF kernels to incorporate invariances. Second, we remove the need to manually define which image transformations the SVM should be invariant against, which is needed in certain methods [2, 9]. Third, we show that the use of max-pooling kernels can make SVMs more robust against noise in the data samples.

2. Related Work

Invariance of the class to a transformation in the input is a type of prior knowledge that can be incorporated in a classifier [8]. An effective method to incorporate invariances in a classifier is to augment the data with transformed versions of training samples. Data augmentation has been shown to be effective in CNNs using data sets with geometrically transformed training samples [7] and with photometrically transformed samples [10].

The Virtual Support Vector (VSV) method [9] is a method based on data augmentation, specifically for SVMs. Instead of augmenting the original data set, a SVM is first trained using the original data set. Afterwards a new training set is generated by applying transformations on the support vectors of the trained SVM. A new SVM is trained on the transformed support vector set, which makes the second SVM more invariant against the transformations. The downside of data augmentation techniques is that the training duration becomes larger. Further, it is necessary to manually define which transformations are used to augment the data set.

Instead of augmenting data sets, some methods modify SVM kernels to induce invariances while learning. Examples of such kernel methods are jittering kernels [2], tangent distance kernels [4] and Haar-integration kernels [5]. Jittering kernels for example consider the transformed variations of its input samples given a set of manually defined transformations. Jittering kernels define the distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ as the shortest distance between $\mathbf{x}_i$ and each transformed variation of $\mathbf{x}_j$. A benefit of such kernel methods is that the data set does not need to be enlarged. To keep this benefit we also show a method for incorporating invariances by modifying kernels only.

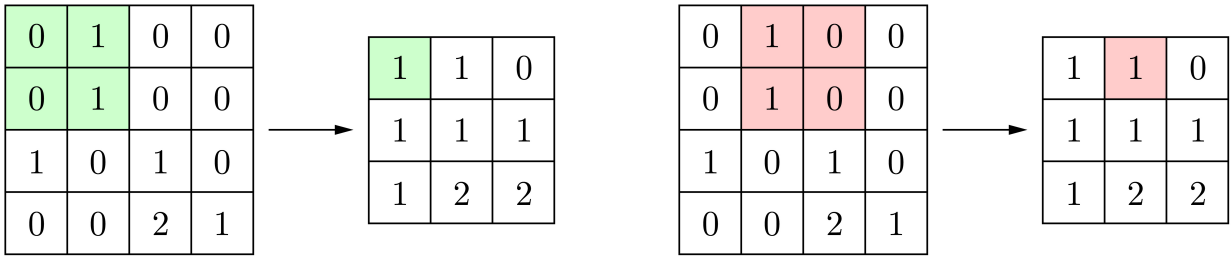

Figure 1: Two steps are shown of a 4×4 input layer that is filtered with a max-pooling filter of size 2×2 with a stride of 1. Each square of 2×2 in the original layer is merged into a single value in the output layer by taking the maximum value in the 2×2 filter. The resulting output layer has a reduced size of 3×3.

3. Max-Pooling

Max-pooling is a process where an input representation is downsampled by taking the maximum of groups of input values. Max-pooling can be used to reduce the dimensionality of layer vectors to create a downsampled vector that retains the most important information. For example, CNNs use max-pooling to retain the presence of signals in input representations rather than the exact location of the signals.

Max-pooling is applied by using a filter of a predefined size $m \times n$ and by sliding this filter across a rectangular input representation such as an image. The output of the filter is the maximum of the mn values covered by the filter. The filter can be shifted one position at a time or can jump across the input layer with a predefined stride size. By applying this filter on a complete input layer, the input is effectively downsampled into a new layer with less features. Fig. 1 shows the working of a max-pooling filter of size 2×2 with a stride of 1 on a 4×4 input layer.

4. Max-Pooling Kernels

Support Vector Machines [11] make use of various kernels to train on the data set and to predict labels using its support vectors. Several kernel functions are available such as the linear kernel, polynomial kernel and the Gaussian RBF kernel, which are respectively defined as

$$k_{\text{linear}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{1}$$

$$k_{\text{poly}}(\mathbf{x}_i, \mathbf{x}_j) = \langle (\mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d, \quad d \in \mathbb{N} \tag{2}$$

$$k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right), \quad \sigma > 0, \tag{3}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two input samples, d is the polynomial degree and σ is a free parameter. Often $\gamma = 2\sigma^{-2}$ is used as the free parameter. Because the kernels define the notion of similarity for a SVM, the kernel choice will influence the performance of a SVM. It is necessary to choose a fitting kernel for any given problem to achieve optimal SVM performance. In order to incorporate a max-pooling filter in a kernel, this must be possible regardless of the type of kernel.

To create a max-pooling kernel, we incorporate a max-pooling filter in any given kernel k as follows. First, we define a function p that takes a feature vector, applies a pooling filter on the square representation of the vector, and returns the vector format of the resulting layer. A kernel k will be modified as follows, such that it applies this pooling filter on its input vectors:

$$k'(\mathbf{x}_i, \mathbf{x}_j) = k(p(\mathbf{x}_i), p(\mathbf{x}_j)). \tag{4}$$

Kernel k' first applies a pooling filter on both input vectors, before calculating their similarity. The max-pooling filters remove slight transformations on the input vectors, such that input vectors that were originally not similar are considered more similar by the kernel.

5. Experiments

To determine whether max-pooling kernels actually make SVMs invariant against image transformations, we built an artificial data set that is easier to learn by invariant classifiers. The data set has three classes and contains images of 30 by 30 pixels with randomly placed vertical lines of arbitrary lengths. The data set has 2400 samples with 800 samples per class. The classes are determined by the horizontal location of the lines. Class 0 contains images with lines in the left third of the image. Class 1 with lines in the middle third, and class 2 with lines in the right third. Classes that are next to each other have some overlap in the data set, to avoid perfect learning scores. Two data samples of this data set are shown in Fig. 5.

Figure 2: Two samples of the artificial data set with left a sample from class 0 and right a sample from class 2.

A classifier learning this data set needs to determine whether a line is in the first, second or last vertical segment of the image. As the classification task is location based, it is beneficial for a classifier to be invariant against small transformations. An image from any class can be translated by some small values and the resulting image will probably still be part of the same class. An invariant SVM can detect this and will be able to classify the correct class more often than a standard SVM, given the same data samples.

For our experiments we use a C value of 2 and RBF kernels for each SVM. We use one standard SVM, and two SVMs with max-pooling kernels. The first max-pooling SVM uses a filter size of 2×2 with a stride of 1 and the second max-pooling SVM uses a filter size of 3×3 with stride 1. Finally we also apply the VSV method on a fourth SVM. For the VSV SVM we generate the training set with 8 translated versions of the support vectors. The vectors are translated up, down, left and right with 1 pixel and are also translated in each diagonal direction with $\sqrt{2}$ pixels.

5.1. Exp. 1: Are Pooling SVMs Invariant?

If pooling SVMs are indeed invariant to translations, the performance of pooling SVMs should be similar to or better than the performance of the VSV SVM on the artificial data set. Further, the pooling SVMs as well as the VSV SVM should both perform better than the standard RBF SVM. To investigate whether these conditions hold, we ran each of the four SVMs on the artificial data set with a 0.9 training set to test set ratio. As the RBF kernel contains an optimizable parameter γ, the SVMs are run with multiple values of γ. Fig. 3 shows the generalization error of the four SVMs on the artificial data set, averaged over 5 runs. Table 1 shows the minimum average errors of each SVM with the corresponding γ value.

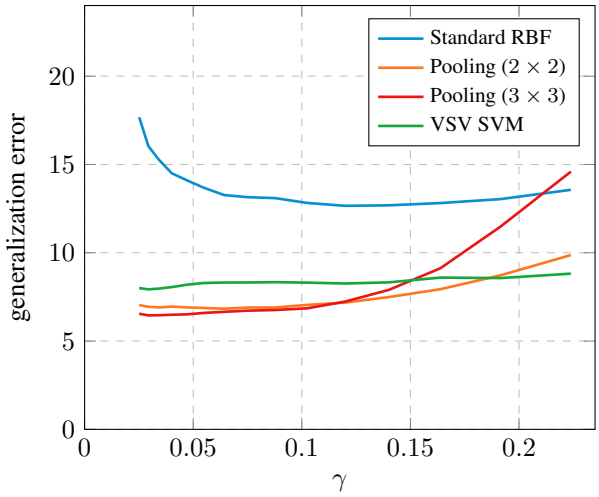

Figure 3: The 5-run averaged errors of a standard RBF SVM, two pooling SVMs and a VSV SVM on the artificial data set for various values of γ.

Table 1: The minimum 5-run averaged errors of a standard RBF SVM, two pooling SVMs and a VSV SVM on the artificial data set with the corresponding γ value.

SVM	min. average error	γ
Standard SVM	12.66	0.120
Pooling SVM (2×2)	6.83	0.064
Pooling SVM (3×3)	6.45	0.029
VSV SVM	7.93	0.029

It is visible that the SVMs with pooling kernels as well as the VSV SVM have a lower generalization error than the classical RBF SVM for most values of γ. Clearly these three SVMs are more invariant against translations of the vertical lines, and can more often deduce the correct class than the standard SVM. Further we see that the pooling SVMs perform slightly better than the VSV SVM, with an error of 6.83 and 6.45 for 2×2 and 3×3 pooling respectively, versus 7.93 for the VSV method. Thus, without enlarging the data set with manually defined transformations such as necessary for the VSV method, the pooling SVMs still manage to be more invariant than the VSV SVM and the standard RBF SVM.

5.2. Exp. 2: Pooling Kernels and Variable Training Set Sizes

In the previous experiment the training set to test set ratio was 0.9, but what if we increased the number of training samples for each SVM? Does each model perform equally well for a large number of training samples, or do the invariant SVMs have an edge over the standard SVM for this data set? We performed a second experiment to determine

the learning curves of the four SVMs for the same data set. Fig. 4 shows the learning curves of the four SVM models for training set sizes ranging from 5 to 1000, averaged over 5 runs. For each SVM we used the corresponding γ value in Table 1 in each run.

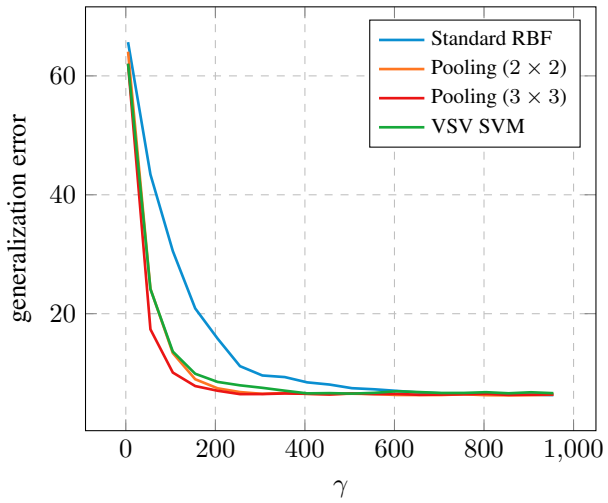

Figure 4: Learning curves of a standard RBF SVM, two pooling SVMs and a VSV SVM on the artificial data set for fixed values of γ.

Based on Fig. 4 we see that each classifier approaches the same performance as the number of training samples grows. Clearly classifier invariance is not necessary for this data set, as long as there are enough training samples. However, it is interesting to see that the invariant classifiers perform better when there are fewer training samples. We see that the pooling SVMs and the VSV SVM perform better than the standard SVM with training sets of fewer than 600 training samples. It is thus attractive to use invariant classifiers, if the number of training samples is limited. Further we see that the 2×2 pooling SVM has a similar curve to the VSV SVM, but the 3×3 pooling SVM performs even better than the other two in a small region. The lower errors of the 3×3 pooling SVM can be explained by the larger range of transformations that are covered by the 3×3 pooling filter.

5.3. Exp. 3: Pooling SVMs and Noisy Data

Max-pooling filters only extract the maximum values of the filter covered values, which means that max-pooling is mainly sensitive to the highest values in the input. If we consider a noisy input image, a max-pooling filter should be able to extract the original data from the image, as long as the noise values are small compared to the actual input values. Would a pooling SVM be able to perform better on a noisy data set than a standard SVM?

We investigated this by using the same artificial data set, to which positive Gaussian noise was added with a 0.2 standard deviation. The values of the vertical lines are still higher than the noise values, so the max-pooling filter should be able to extract the vertical lines. Examples of the artificial data set with Gaussian noise are shown below.

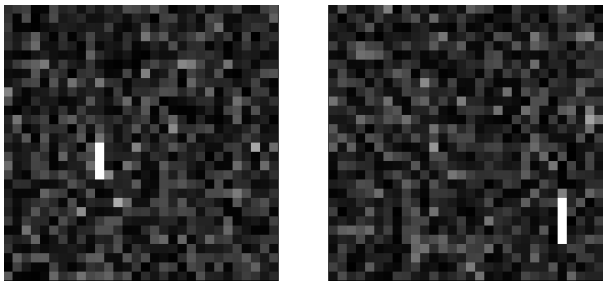

Figure 5: Two samples of the noisy artificial data set with left a sample from class 1 and right a sample from class 2.

We recreated experiment 1 and experiment 2 using the noisy data set. Fig. 6 presents the generalization error of the four SVMs on the noisy data set, averaged over 5 runs. Table 2 shows the minimum error values of each SVM with the corresponding γ value. Finally, we show the learning curves of the SVMs in Fig. 7 using the γ values in Table 2 corresponding to each SVM.

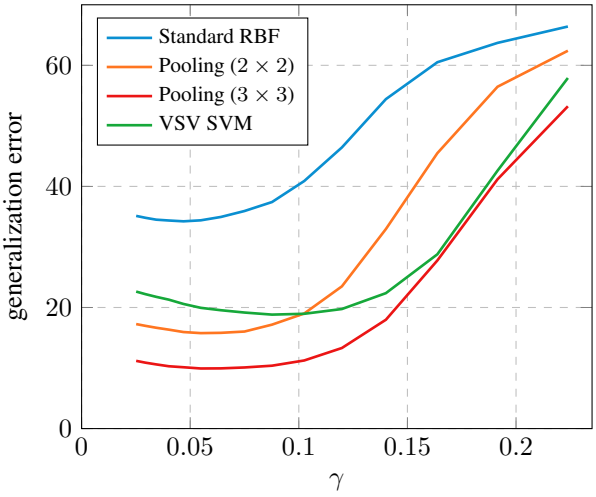

Figure 6: The 5-run averaged errors of a standard RBF SVM, two pooling SVMs and a VSV SVM on the noisy artificial data set for various values of γ.

Table 2: The minimum 5-run averaged errors of a standard RBF SVM, two pooling SVMs and a VSV SVM on the noisy artificial data set with the corresponding γ value.

SVM	min. average error	γ
Standard RBF	34.23	0.047
Pooling SVM (2×2)	15.77	0.055
Pooling SVM (3×3)	9.93	0.055
VSV SVM	18.82	0.088

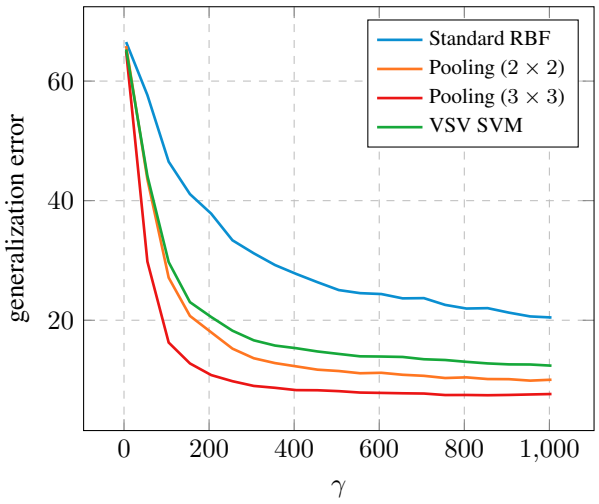

Figure 7: Learning curves of a standard RBF SVM, two pooling SVMs and a VSV SVM on the noisy artificial data set for fixed values of γ.

Fig. 6 and Table 2 clearly show that the pooling SVM has a better performance than the other SVMs. Not only does this hold for a small number of training samples, but also for larger training sets as suggested by Fig. 7. For this particular data set we can say that the max-pooling filters make the SVMs more robust against noisy data.

6. Discussion

Pooling worked in our case for simple input

Might not work for low resolution images as details are lost

Pooling helped against noise, but that means that pooling kernels are oblivious to certain textures, while they may be important

It is possible to combine multiple kernels with various filters

References

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, Aug. 2017.

[2] Dennis Decoste and Bernhard Schölkopf. Training Invariant Support Vector Machines. *Machine Learning*, 46(1):161–190, Jan. 2002.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *Object Recognition Supported by User Interaction for Service Robots*, volume 2, pages 864–868 vol.2, Aug. 2002.

[5] B. Haasdonk, A. Vossen, and H. Burkhardt. Invariance in Kernel Methods by Haar-Integration Kernels. In Heikki Kalviainen, Jussi Parkkinen, and Arto Kaarna, editors, *Image Analysis*, Lecture Notes in Computer Science, pages 841–851, Berlin, Heidelberg, 2005. Springer.

[6] David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, volume 6354, pages 92–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Eric Kauderer-Abrams. Quantifying Translation-Invariance in Convolutional Neural Networks. *arXiv:1801.01450 [cs]*, Dec. 2017.

[8] Fabien Lauer and Gérard Bloch. Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing*, 71(7):1578–1594, Mar. 2008.

[9] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines. In Christoph von der Malsburg, Werner von Seelen, Jan C. Vorbrüggen, and Bernhard Sendhoff, editors, *Artificial Neural Networks — ICANN 96*, Lecture Notes in Computer Science, pages 47–52, Berlin, Heidelberg, 1996. Springer.

[10] Luke Taylor and Geoff Nitschke. Improving Deep Learning using Generic Data Augmentation. *arXiv:1708.06020 [cs, stat]*, Aug. 2017.

[11] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.