Incorporating Invariances in Support Vector Machines using Max-Pooling Kernels

Tuhin Das
Delft University of Technology
Institution1 address
t.das@student.tudelft.nl

Second Author
Institution2
First line of institution2 address
secondauthor@i2.org

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. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

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 [1]. Many classifiers are however not invariant, an example of which is the Support Vector Machine (SVM). As SVMs generate hyperplanes that separate different classes as much as possible, SVMs remain adequate learners for image classification. It is thus desired to improve the classification capabilities of SVMs by incorporating invariances to image transformations.

In this paper we present a new method to make SVMs more invariant. Inspired by CNNs, which are partially invariant due to their max-pooling filters [3, 5], 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. Training SVMs with such max-pooling kernels enables the SVMs to become invariant to small image translations and rotations.

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, 7]. Third, we show that the use of max-pooling kernels makes SVMs more robust against noise in the data samples.

2. Related Work

Virtual Support Vectors [7]
Data Augmentation [6]

Jittering Kernels [2]
Tangent Distance [4]

3. Support Vector Machines

Support Vector Machines [8] are binary classifiers that find the optimal separating hyperplane between two classes to maximize the margin between the classes. If the original problem is not linearly separable, a mapping function $\Phi : \mathbb{R}^p \to \mathbb{R}^q$ is used to map each data sample in a new space where the problem is linearly separable. The SVM is trained by finding the unique solution of the following convex quadratic problem:

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad (1)$$

$$\text{subject to } y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \ \forall i \quad (2)$$

where $\mathbf{w}$ is a vector of weights, C is the regularization parameter, ξ_i is a slack variable and b is the bias term of the separating hyperplane. The solution of this optimization problem is found by solving the dual problem, which is formulated with the Lagrangian:

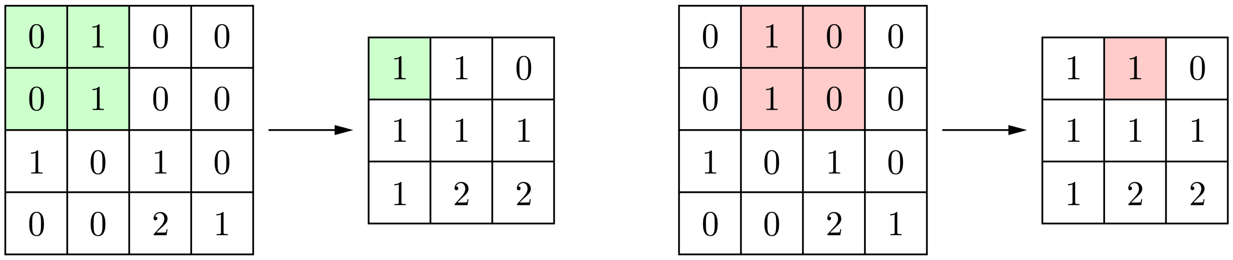

Figure 1. Two steps are shown of a 4×4 input layer that is filtered with a max-pooling layer of size 2×2 with a stride of 1. Each square of 2×2 in the original layer is merged into a single value in the output layer by taking the maximum value in the 2×2 filter. The resulting output layer has a reduced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

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4}$$

$$\text{and} \quad \alpha_i > 0, \ \ \forall i \tag{5}$$

where α_i is a Lagrangian multiplier, and $k : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ is a kernel function such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$.

Several kernel functions are available such as the linear kernel, polynomial kernel and the Gaussian RBF kernel, which are respectively defined as

$$k_{\text{linear}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{6}$$

$$k_{\text{poly}}(\mathbf{x}_i, \mathbf{x}_j) = \langle (\mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d, \ \ q \in \mathbb{N} \tag{7}$$

$$k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right), \ \ \sigma > 0. \tag{8}$$

Because the kernels define the notion of similarity for a SVM, the kernel choice can influence the performance of a SVM. Hence, it is necessary to choose a fitting kernel for any given problem.

4. Max-Pooling Kernels

Max-pooling filters are often used in CNNs to cluster information in smaller representations. In CNNs the presence of signals can be more important than their exact locations. Max-pooling layers can be used to reduce the dimensionality of layer vectors to create a downsampled vector that retains the most important information.

Max-pooling layers use a filter of a predefined size such as 2×2 and slide this filter across an input layer. The output of the filter is the maximum of the values covered by the filter. The filter can be moved one place at a time or can stride across the layer with a predefined stride size. 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.

We incorporate a max-pooling filter in a 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. It is possible to redefine a kernel such that it applies this pooling filter on its input vectors. A kernel k will be modified to give a pooling kernel k':

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

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

For our

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. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

6. Discussion

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. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

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] 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.

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

[7] 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.

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