Hardware Implementation of Learning Feature Trees

(Patent pending)

🅭 2024 William Ward Armstrong, PhD

October 16, 2024

Abstract

A method is described for developing real-time machine learning systems based on Learning Feature Trees which grow according to training data. Pattern classification is performed by matching incoming images with a hierarchy of features that are automatically developed. The features define hyperplanes that partition the space of all possible images for a given retina into blocks. On each block, a linear function estimates the probability of membership in a target class. In comparison to convolutional neural nets used for the same purpose, this system is more easily understood and is expected to be faster in execution on simple hardware that uses little electrical power. The system is based on forward-propagation of credit assignment rather than back-propagation.

1 References

1. https://github.com/Bill-Armstrong/Real-Time-Machine-Learning

2. ”The MNIST Database of handwritten digits”. Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond.

3. ”Adaptive Logic Networks”. William W. Armstrong, Monroe M. Thomas, Handbook of Neural Computation, IOP Publishing and Oxford University Press, 1997, pp. C1.8:1 - C1.8:13.

4. ”Methods to Control Functional Electrical Stimulation in Walking”, R.B. Stein, A. Kostov, M. Belanger, W. W. Armstrong, D. B. Popovic, First International FES Symposium, Sendai, Japan, 1992 (Invited paper) pp. 135 - 140.

2 Background of the invention

Previous work done by the author using piecewise linear data fitting by Adaptive Logic Networks (ALNs, see ref.) used neural networks based on a layer of linear functions followed by layers of maximum and minimum operators. This was used successfully in many areas including forecasting of electrical power produced by wind-turbines, computing the health of gas turbines, and controlling a model of an off-road vehicle active suspension system. In these areas, power consumption, weight and the cost of devices were not important considerations. However in work done on prostheses for the disabled, e.g. to enable locomotion for spinal-cord injured persons (see ref.) these are very important for use outside the laboratory. In addition, for such applications the operation of the device must assure the safety of the user, and for that purpose, one needs to have a clear idea of how the system operates so there will not be any unexpected and undesirable behaviors.

There was a problem in using the ALN system, and that was the complete re-evealuation of the network necessary to find the linear piece responsible for computing the value of the output. It could be accelerated using alpha-beta pruning, but was still slower than necessary.

The advance described here uses linear pieces but is different in that the linear function to be given credit for an action is found by evaluation of a few features of a Learning Feature Tree and the path can be recorded in a few bits. ALNs produced a continuous function, but an LFT produces a discontinuous function on the space of all possible inputs. Discontinuity is easily overcome while at the same time increasing accuracy of a classification by employing a plurality of Learning Feature Trees (LFTs) whose outputs are used in a weighted average. We call this the method of ”Weighted Overlapping Linear Functions” (WOLF).

3 Description of the invention

Learning Feature Trees are described in the GitHub reference above in a pdf document and a C++ program. The program demonstrates using the method for classification of handwritten digits taken from postal codes on letters (MNIST data, reference above). The hardware implementation described below can accelerate the processing of the program in two ways: 1. by executing many LFTs at once and 2. by accelerating the processing of the LFTs.

In the following, we shall prefer the concrete terms image, sensor, pixel, intensity, retina etc. to mean more general objects which can be characterized by measurments obtained by sensors from the environment or which originate in the machine learning system itself (e.g. probabilities). The measurements may each have different ranges of values or scales. Probabilities would usually be floating point values between 0 and 1. A pixel will be one position for the values in a list of measurements characterizing an object called a retina. The measurements in a retina can be collected from sensors or the machine learning system's outputs at possibly different previous instants. In this description, a linear function can have a constant term, i.e. it is generally an affine function. The domain of a linear function will usually be the set of all possible measurements for a given retina. Often the domain will be the set of all possible images from a device like a webcam.

An LFT, mathematically speaking, is made up of nodes connected in the form of a binary tree (Figure 1). There are two kinds of nodes: 1. branching nodes where each node has two child nodes and 2. leaf nodes which have no child nodes. Both types are physically the same and contain the data defining a linear function acting on the values of pixels of an input image. The hardware is like a smaller LFT (Figure 1), a number of copies of which can represent and cover all the nodes of an abstract LFT, with maybe some unused hardware nodes like descendants of leaf nodes.

The reasons for not implementing a large tree in hardware that could do evaluation of all features on an image in parallel are that

1. this would require a lot of hardware and half of it would be made useless by the first result of applying a feature to an image;
2. large hardware would use a lot of electrical energy even if parts of the device could be turned off as computation advances;
3. prefetching some feature and image data and starting several computations without waiting for the first result can be done in a smaller tree and thereby accelerate operations without using a lot of hardware and electrical power.

During execution, after values of the input to be classified and the node's feature's or linear function's coefficients are loaded into registers of the LFT, both types compute the inner product of the image and the coefficients and add a constant term. We refer to this as evaluating a feature on an image.

Branching nodes use the sign bit of the feature value to decide which of the node's two children gets to further process the input image. Leaf nodes use the value as a membership function giving an extimated probability of the image belonging to the target class, one of the two classes involved in a classification task. The non-target class may be defined simply as the images that are not in the target class. Leaf nodes must provide probability values between 0 and 1 including these two limits, so these upper and lower bounds must be imposed on the linear least squares fit to the reinforcement values whose images belong to the leaf. Bounds can be enforced either by hardware in the LFT or by the computer.

The nodes on a path from the root to a leaf also compute the minimum of absolute values of features of branching nodes as a weight that goes to zero at leaf block boundaries. It is also bounded by 0 and 1. The feature value is returned to the computer at nodes along the path if the weight is not computed by the LFT. The weighted average of class-probability values of several LFTs tends to be more accurate than using a single tree and the result is a continuous membership function unlike the discontinuous function produced by a single LFT. To simplify the diagrams, we omit hardware for calculating the weight, since it is easily designed and is not critical for speed.

An input pattern to be classified enters from the host computer at the root node of the LFT. After evaluation of the feature on an input image, the sign bit of the value is used to select either the left or right child to continue processing the image. (If power is not switched off to one child, then both continue processing but one will be ignored). The sign bits along the path are concatenated into a string that functions (together with an LFT identifier, not described) as an engram identifying the selected leaf. In a leaf node, the linear function is evaluated on the input image and the value is returned to the computer. The path of selected nodes leads from the root node to a single leaf node.

When the registers would be too large for storing a feature or an image all at once, or too many wires are required for parallel operation, the solution is to use fast memory to store the whole feature and image data in the device and do the loading and computation in several steps sequentially. We omit the details, since doing this is well within the state of the art. Only this memory size then limits the size of retinas that could be accomodated.

Mathematically, a feature determines a hyperplane that, when evaluated on inputs, partitions the set into two subsets. The bounding values of the sensors (e.g. 0 and 255) define hyperplanes too. Together, these result in blocks which are convex sets of images (as points in the space of all possible image measurements which is a 784-dimensional hypercube in the case of the MNIST data). It would be ideal if all the blocks in an LFT contained only images of the target class, or no images of the target class, or no images at all, but in general a block will contain a mixture of the two classes. In that case, the linear membership function computes (after bounding) an estimated probability that the input is a member of the target class. That function can be constant zero or conastant 1 to cover the previous cases.

The present application sketches digital hardware that could be used for executing the pattern classification task of a subtree of an LFT (or all of the LFT if it has enough nodes). The hardware described optimizes just the classification operation as carried out after training the LFT, but there are other operations in training of LFTs that could benefit from using similar hardware. The machine learning algorithm can use the hardware described here to accelerate training.

The hardware interfaces with a computer. The operation of the computer unique to the hardware will be sketched here, but generally follows what is described in the C++ program referred to above. One difference is that the program eliminates some inputs when their variance is judged too low to discriminate between classes. This is possibly not very useful if hardware is available.

Learning consists in defining LFTs so that at the end of training, the trees can produce an estimate of the probability that an input pattern belongs to the target class. Each LFT also outputs a weight that is generally around 1 in the centre of a leaf node block, but goes to zero at the edges of the block. By using a pluality of LFTs which are shifted in a regular way in the image space, so the boundaries of leaf blocks tend not to coincide, and taking the weighted average of the probabilities coming from several trees, there will be non-zero weights. Hence the weighted average can be computed. Should all trees give a zero weight for a certain image, the system could extrapolate a value by selecting two images near that image but with non-zero weights so that all three images are in a line.

Sometimes the term forest is used for a collection of trees, but we suggest the term *orchard* for a complete collection of trees that are systematically shifted to perform the same classification task. Shifting changes the placement of hyperplanes of different LFTs, hence it creates leaf blocks with different training sample data, thereby increasing statistical accuracy of the weighted average probability. The program currently creates orchards of trees which are all trained. It is easier and faster to simply shift the first tree of an orchard to form the other trees. A possible problem with doing that is that some leaf nodes may be shifted to where there is no data.

The hardware evaluation of a feature or leaf linear function is the same. The only difference is whether the value is used for branching or producing a probability estimate. If an input has N values, say from an image with N pixels, then the feature will have N values in one-to-one correspondence. The corresponding values have to be multiplied, all products added together and the constant term added(Figure 2). To accelerate this operation, it is done in parallel by grouping two products together, adding them and doing this for N/2 groups (Figure 3). Then the groups are grouped by two's, the results added, and so on until a final sum is reached. The constant term can be added at the end, but it should be added earlier where a layer has an opening.

If N = 1000, say, and if an addition takes 50 ns and a multiplication takes 100 ns, then arithmetic that would take 150000 ns if done in a sequence, would be done in 600 ns in parallel, a speedup of 250 times. This neglects other operations like loading.

The sign bit of the evaluation is used to pick one of the child nodes. It doesn't matter whether the sign bit of the value zero is one or zero as long as it is fixed. The value zero defines a hyperplane that partitions the current block in two.

A possible function of the sign bit is to cut electrical power to a non-selected child and its descendants once the selection is determined. If power is delivered by an input to the root (see Figure 1), then both children (and maybe other descendants) of the root will get power for prefetching, but only the selected child and its descendants will get power after the root evaluation is finished. Only the powered node in a layer has its sign bit transmitted to the computer. The values of these bits from the root to the node before the leaf determine for the computer where to get the output of the single selected leaf. This is part of the engram that allows forward propagation of credit assignment.

The weights are determined by the absolute values of the evaluated features on the input. On the boundary hyperplanes of leaf blocks, the weights are zero. How fast it varies between 0 and 1 determines the smoothness of the membership function when a plurality of LFT are used, and this is determined by the norm on the direction vector of the hyperplane. To get a continuously differentiable membership function, a cubic function of the absolute values of the evaluated features on the input may be used. The weight for the LFT could be computed by the computer or by the hardware since it is not on the critical path.

If a single piece of hardware is too small to contain the whole path to a leaf, the values from the selected branching node furthest from the root are passed to the root of another hardware LFT and the path is continued, the sign bits being concatenated.

The operation of the hardware can be thought of this way: for highest speed, all features in an LFT would be loaded and evaluated on the input simultaneously and that would tell which leaf node is selected. But this would waste a lot of electrical power because most of the tree is not useful for that path. Only the selected path to a single leaf node is useful.

By using a small subtree which pre-computes several features, the trade off between speed and power is made. For example, the power can be passed from the root to both child nodes at the start and only when the selected child is determined the power to the other child and its descendants is cut off. This power gating could be done with a signal from the feature evaluation or derived from a clock circuit since the evaluation is constant time. The sequence of hardware operations is that nodes of the tree are loaded with features and functions starting at the root and evaluations are started. The evaluations are stopped at any node when power is cut off, providing power only to the selected nodes on the path to a leaf.

4 The many uses of features

The use of a convolution to smooth a feaature is important in the following way: moving the image or some of its pixels a bit in any direction results in a slightly changed value when the feature is applied to the image. Classifications are not likely to change. This effect disappears when the motion is the size of the convolution kernel's radius. For images located beyond that range, we could translate pixels of the feature to recognize translated images that are beyond the range of the first filter. There are edge effects when pixels leave the feature and some new ones arrive. We have to find a way of dealing with that but once we have that, the possibilites for creating new recognizers for distorted images is vast. We now examine a few of the vast array of possibilities for using modifications (distortions) of features in our pattern recognition system, beginning with data augmentation, the technique of creating artificial images that belong to a given class that can be added to the training set so the generalization is improved.

Pick any two pixels of an image and the corresponding two pixels of a feature. Now interchange the values of the two pixels in the image and do the same with the feature's values. The resulting value of applying the feature to the image is unchanged. This can apply to changed features in an entire LFT. Any transformation of an image done by interchange of pixel values can be mirrored in features so that the transformed image is classified the way it was before by the LFT. Any distortion of an image done by interchanges in this way can be undone by doing interchanges of pixel values in reverse order. We call this the inverse distortion.

Examples of image changes done by interchanges are: cyclic shifts left and right or up and down of the whole image. The same shifts applied by varying distances to individual rows or columns or both. In round retinas, rotations and magnifiactions by any amount can be performed. This could possibly be done in a way that resembles having a macula in the eye of the vision system.

If only a fixed part of an image is of interest, then a feature can be made which has zero values outside that part. Say the part is a small circle. Suppose the system has been trained to recognize images of a certain target class that are shown inside a larger circle with a certain size, position and rotation angle. Then the features of that system can be changed in size, position and rotation to recognize the same target in the small circular part of the image.

The inverse image can be used to do data augmentation. If the distorted image of numeral six still looks like a six but is not correctly classified as such, then by adding it to the training set, generalization can possibly be improved. By applying to features of an LFT the distortion of how we want the images augmented, we get the same classifications as if the images were distorted and the LFT features were not. We can do it the other way around, perhaps automatically. We take an image and distort it by interchanges so it becomes a model of the target class. Say the numeral six. Then applying the inverse distortion to the filter, we get a recognizer for the distorted six. So we generate new LFTs, one new LFT per original LFT per distortion and achieve the effect of data augmentation.

In general if something is not recognized by an LFT, but should be in the target class, we can add to the original orchard modified LFTs. Since the probabilities of the original orchard are small and their weighted average is small, the existing LFTs will not impact the decisions of the augmented system.

Why should this method of data augmentation be able to raise the classification accuracy on a test set of images? Because if you look at the accuracy of testing on the training set as done in the program, you see that the accuracy is close to 100 percent. The LFT system appears capable of absorbing the effect of almost all training images.

One promising idea is to develop LFTs with distorted features that correct the badly classified images of the original training set. (If a classification label is really questionable by human judgement, then maybe it is better just to remove the said image from the training set.) For the sake of efficiency, one would apply distorted features only to doubtful classifications obtained using the undistorted features first. Such doubtful cases could be identified by a low probability of membership in the class obtained by using the original features. It turns out that badly classified training images using the program for MNIST data have probabilities very close to 0.

The likely distortions in the MNIST data could be slight changes of size, translations, tilts and slight curves that could come from right- or left-handed writers. The curves could be done by small cyclic shifts of individual layers in a retina. If the image is generally zero near the edges, it is not necessary to use cyclic shifts, but rather zero-filled ones.

This is the general idea, but to realize it fully, we have to be able to move pixels by smaller increments to easily reshape an image for augmentation. This can be done by dividing each pixel into sub-pixels, say, 100 smaller squares in a grid. Originally they all have the same value as the complete pixel. Then we apply a distortion to features the LFT for a class. The number of pixels in the new features would be 100 times larger. It is easy to change an original image to use such a feature by simply copying values from larger pixels to sub-pixels.

5 Efficient use of the hardware

Since, at the time of writing this document, no LFT hardware exists, this section is speculative. A single LFT produces a rough estimate of the probability that the input image is in a target class. That probability estimate can become more accurate when more trees in an orchard are added to the weighted average. So far, experiments on MNIST data suggest that the estimate from just one LFT is fairly good. If that is generally true, that fact can be exploited.

Suppose LFTs have been trained to recognize faces, with seven LFTs per face, and suppose there is a collection of 100,000 facial images that have been classified, each by an orchard of seven LFTs. Then all orchards might have to be examined to find the unknown person in an image.

But instead of applying seven LFTs to the unknown to check person number one in the collection, and continuing with seven LFTs for the next and so on, only one tree from each orchard is applied at a time. Each time one applies an LFT of someone in the collection to the image, a probability estimate is returned. If the weighted probability estimate is below a certain level, then the search continues. Otherwise, the remaining LFTs are applied to the image and the weighted average is updated with each one. If a resulting probability estimate is above another threshold, then we consider the unknown to be identified by the high enough probability estimate. We note that distorted features used for data augmentation can be added to an existing orchard and only when the probability estimate is high would that allow the "augmented" data samples to have an effect.

When there is hardware with enogh LFTs on it so that 100, 000 hardware LFTs can be executed in parallel, then the search would be done in the time required to evaluate one orchard, or less if doing just one LFT in each orchard at a time finds an individual with high enough weighted average probability. Some hardware would be freed up after the first LFT of all orchards had been applied, and one could continue looking for the maximum probability estimate of the promising orchards left.

One question relating to the "competition": how could you implement transformations (distortions) with a convolutional neural net? We assume that the convolution operation has be absorbed into the first layer of any trained net, otherwise, convolution would be a drag on speed. It could be done by distortion of the images and using the original net to recognize the class, but that would be an extra step before using the net and doing one distortion at a time. Using LFTs and hardware appropriately, it might be much faster. If the image could be in any of 100, 000 classes, using all available hardware to apply one LFT from different orchards (including appropriate distortions) and continuing with LFTs from higher-probability orchards could be a good strategy. Note that the weighted average using one LFT is the same as not using weights.

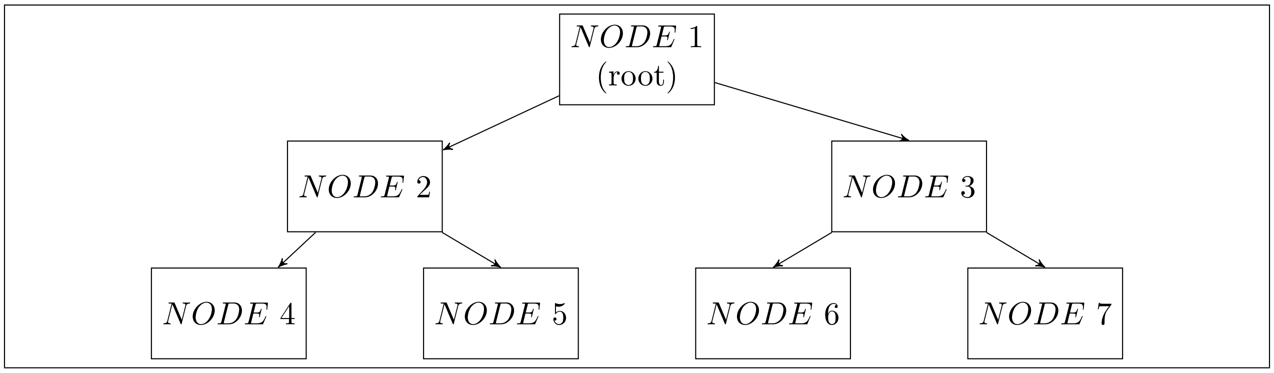

Figure 1: Seven nodes of a small hardware device. Arrows show transmission of electrical power and the image to be processed. Connections from/to the computer are not shown.

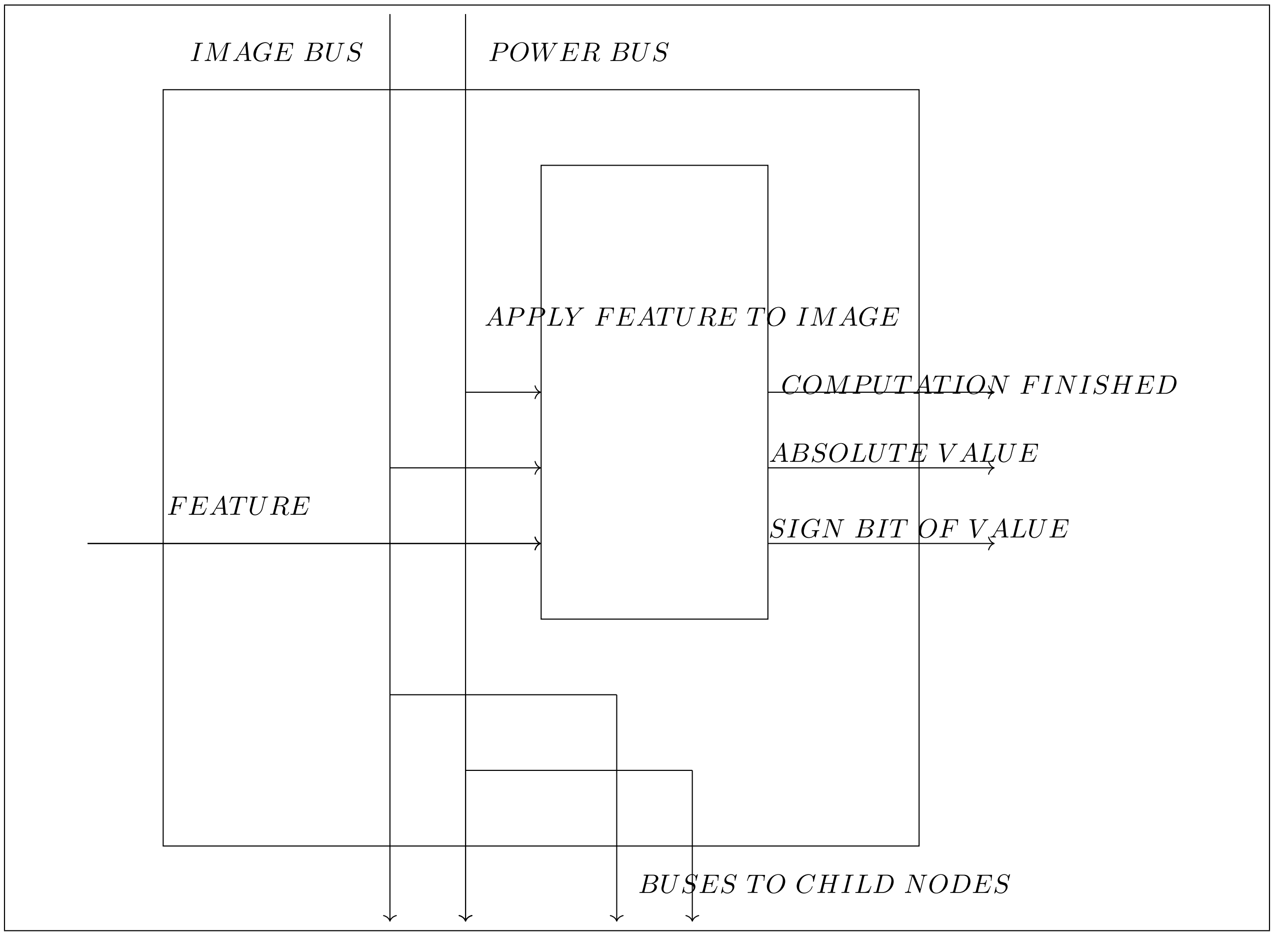

Figure 2: Node connections

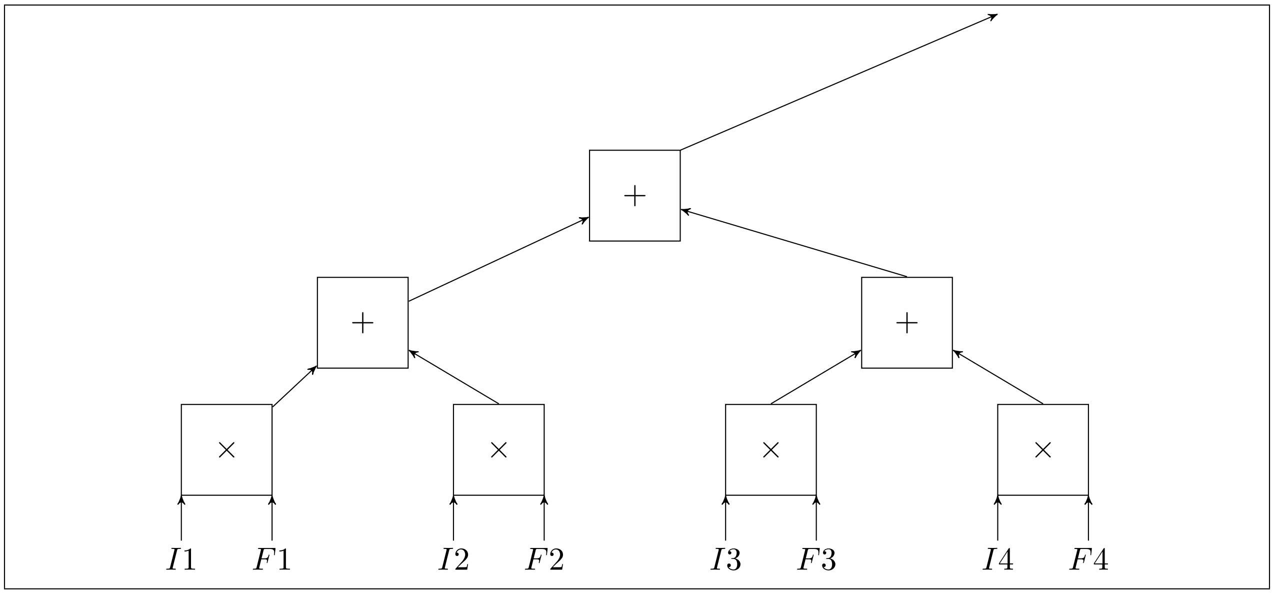

Figure 3: Operation of a feature on an image. Fn = pixel n of feature, In = pixel n of image. If F0 is the constant term, it could replace an unused product entering the sum. The value is output as the sign and its absolute value. This is bounded above by 1.0 for every purpose: defining a hyperplane, determining a probability value or assigning a weight.