Identifying Melanoma using Computer Vision and an Artificial Neural Network: A Senior Thesis in Computer Science

Hannah Rivers

September 2011 – April 2012

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer luctus justo a nibh mollis pretium. Etiam rhoncus augue in velit venenatis et porttitor metus porta. Duis imperdiet posuere viverra. Sed non eros eu enim cursus iaculis. Donec laoreet, urna a bibendum tristique, lacus sem porttitor nisl, ut venenatis dolor quam gravida purus. Integer sed placerat neque. Quisque euismod pretium tortor tristique venenatis. Sed faucibus erat sed mi hendrerit euismod. Quisque commodo eleifend orci eu vestibulum.

Contents

1 Thesis statement **4**

2 Introduction to the problem **5**

2.1 Background information on human classification of melanoma . . 6

2.2 Past approaches in the field of machine learning 8

3 Theory of neural computation **15**

3.1 An introduction to neurocomputing 15

3.2 Biological neural networks 16

3.3 Artificial neural networks 18

3.4 Types of problems artificial neural nets are best suited for 19

3.5 Why an ANN is a good fit for this problem 21

4 Types of neural networks **22**

4.1 Necessary concepts 22

4.2 The perceptron 25

4.3 The multi-layer perceptron 28

4.4 Restricted Boltzmann machines 33

4.5 The deep belief network 33

5 Considered and selected approaches **35**

5.1 Questions of intent 35

5.2 Preprocessing techniques 36

5.3 Selection of classifiers 37

5.3.1 ABCD rule (Stolz method) 39

5.3.2 Menzies' method 40

5.3.3 Seven-point checklist 41

5.4 Software considered 41

5.5 Practical limitations 41

5.6 Selected approaches . . . 43

6 Final product **44**

6.1 Structure of ANN . . . 44

6.2 Marked-up code . . . 44

6.3 Results with benchmarks . . . 44

7 Future (or current) plans/applications **45**

7.1 Phone app . . . 45

7.2 Web app . . . 45

1 Thesis statement

With the proper image acquisition and preprocessing techniques and selection of classifiers, an artificial neural network can be created and trained to distinguish between malignant melanoma and benign melanocytic lesions.

2 Introduction to the problem

The most common form of cancer in the United States is skin cancer, affecting more than one million people nationally and between two and three million people globally each year. Melanoma is a type of skin cancer; specifically it is a cancer of melanocytes, the cells responsible for the pigmentation of skin. While it is among the least common types, it is the most serious; according to the National Library of Medicine roughly 160,000 cases worldwide are diagnosed yearly, with 48,000 melanoma-related deaths. According to the WHO, the incidence of both non-melanoma and melanoma skin cancers has been increasing over the past decade. At present, one in every three cancers diagnosed is a skin cancer and one in every five Americans will develop skin cancer in their lifetime.

Although serious, melanoma can be cured if it is diagnosed and treated early. If it is not removed in its initial stages however, the cancer cells can grow downward from the skin surface, invading healthy surrounding tissue and organs. Once this has occured, it is difficult to control. Subsequently, the most important aspect of melanoma treatment is to recognize it early.

Obviously, it is beneficial to be cautious when it comes to abnormal skin lesions. However, this caution comes at a price; healthcare is expensive. Many people cannot afford to consult with a doctor or a dermatologist at every suspicion. Mitigating this concern is the practicable eventuality (hopeful result?) of the application of this thesis. In general, the trend of modern medicine is creating machines to perform tasks previously performed by humans, ideally with greater accuracy in less time and for less capital. This thesis is the first step down that road.

On that note, it is critically important to issue the following disclaimer: The intent of any application directly based on this thesis is to serve as a guide to indicate the likelihood of melanoma. It is exclusively a primary recognition tool, and is in no way meant to be used as a diagnostic tool or as a replacement for a doctor. If you have serious concerns that you may have melanoma, it is imperative that you seek medical advice.

2.1 Background information on human classification of melanoma

To assist the public in recognizing melanoma, doctors developed the heuristic acronym "ABCDE." Each letter stands for a characteristic indicative of a melanoma, and meeting one or more of these criteria may indicate its presence. The first characteristic is asymmetry. Common nevi are generally symmetrical circles, while the shape of a melanoma is often irregular. The second characteristic regards the border. The borders of common nevi are usually very well defined and obvious, while a melanoma may have a ragged, blurred, or otherwise irregular border. The third stands for color. Typically, common nevi are a single, uniform shade of medium to dark brown. A melanoma may be a variety of colors including black, red, grey, or even slightly blue. Additionally, its color may not be consistent. The fourth distinguishing characteristic is the diameter. While common nevi are almost invariably one to three millimeters across, a melanoma may be larger than the eraser on a pencil. The last and most dangerous characteristic is evolution of the lesion. Any changes in size, shape, border, color, or structure are highly indicative of a melanoma, as well as the presence of pain, itchiness, bleeding, oozing, or general irritation. To recant, the most common features of melamona include asymmetry, border irregularity, color irregularity, a relatively large diameter, and evolution.

Due to the time-sensitive nature of melanoma treatment, doctors recommend monthly self-examinations to check for skin lesions fitting the ABCDE rule. It is important to familiarize yourself with your moles so as to recognize changes. The use of a hand-held mirror is recommended to inspect hard-to-see regions. Melanoma can occur on the scalp, groin, fingernails, the soles of feet, and in the area between toes, so it is imperative to be thorough.

If a suspicious lesion is found, the next step is to contact a dermatologist or physician. If they think that it's possibly a melanoma, they will biopsy it and send the sample to a pathologist for classification. There are three types of biopsies: punch, incisional, and excisional. The first involves a tool with a circular blade, used to remove a round section of skin containing the lesion in question. In the second, only the most irregular portion of the growth is removed for analysis. In the third, the entire lesion is surgically removed. This is often done when the likelihood of melanoma is high and time is of the essence. Within a few days, the pathologist will inspect the sample and empirically determine whether or not it is cancerous.

On the case of a positive diagnosis, the subsequent action is to quantify the severity of the condition. Melanoma is staged between I and IV, where a stage I melanoma is small with a predominantly successful treatment rate and a stage IV melanoma has spread to other organs and has a very low likelihood of successful treatment. The pathologist assigns the stage through two main criteria. The first is thickness. The thickness of a lesion can be measured under a microscope; the higher the measure the more serious the disease. The second criteria is known as a sentinal node biopsy, conducted by the doctor. This test determines whether the cancer cells have spread to nearby lymph nodes by injecting dye into the area at which the melanoma was removed and charting its course through the lymphatic system. The first nodes to collect dye are removed and tested for cancer cells. If cancer cells are not present in the closest lymph nodes, it is unlikely that the melanoma has spread.

The differentiation of early melanoma from dysplastic nevi is not trivial even for experienced dermatologists. In fact, underestimation of the severity of melanoma in its early stages is sadly not uncommon. However, statistically the much larger issue is the patient failing to notice or to consult a doctor in time. In the majority of cases, if the patient brings the lesion to the doctor's attention early, melanoma can be properly identified and treated.

2.2 Past approaches in the field of machine learning

Medical image recognition is not a new problem in the field of machine learning. There have been numerous previous attempts of both skin lesion classification and similar problems with varying methods and varying levels of success, usually dependent on financial limitations.

Despite their differences in methods, all machine learning approaches to medical image classification necessarily share a set of fundamental characteristics, consisting of image acquisition, feature selection and extraction, image processing and analysis, and classification. The first step in any approach is the acquisition of a set of images on which to train the program. The next is the extraction of pertinent data from these images. Finally, said data must be processed in such a manner that analysis of the results yields a classification. In machine learning, the processing is done through a series of adaptive algorithms that allow the computer to evolve its behaviors based on relationships and patterns present in the data. This conduces more intelligent decision making than can be done with a static algorithm.

In each of the aforementioned areas, the details of operation vary. For instance, the primary question of image acquisition is what type of images the program is ultimately going to be used to classify. These are the images on which it is necessary to train the network. You must then determine where a set of these images can be attained, keeping in mind that supervised learning requires a data set with known outcomes. Next, you must make decisions regarding feature selection and extraction. The success of an image recognition program depends on the correct selection of distinguishing features for classification. They must be quantifiably measureable and have both high sensitivity (the proportion of correctly identified positive responses) and high specificity (the proportion of correctly identified negative responses). In the specific field of skin lesion classification, diagnostic methods such as the ABCDE rule, pattern analysis, Menzies' method, the seven-point checklist, and texture analysis have been developed with effective feature selection techniques. These methods are further explained in section 5.3 on page 37. Once you have chosen your features, the next step is to determine the method through which they will be extracted. Many computer vision libraries and software suites for this purpose already exist and are freely available online, occasionally with modifable source code. Pending feature extraction, the next step is to process the extracted data. Each of the aforementioned diagnostic methods has a corresponding set of rules through which to process feature data into information. Typically, these fall under the category of standard optimization problems that can be resolved with heuristic strategies, greedy or genetic algorithms, artificial neural networks, strategies derived from statistical pattern recognition, or other computational intelligence methods. The final step of the process is to take the information generated in the last step and decide for which thresholds it should be classified into different categories, as well as which categories to use.

For the problem of skin lesion classification, the most substantial issue is that of training image acquisition. This is the area in which budget has the most limiting effect. Ideally, you may use epiluminence microscopy (ELM), which is the examination of skin lesions with a dermatoscope. This process consists of a magnifier, typically x10, a non-polarized light source, a transparent plate, and a liquid medium between the instrument and the skin, which allows detailed inspection of skin lesions unobstructed by reflections on the skin's surface. It is used to render the epidermis translucent, making the dermal features clearly visible. Another option is transmission electron microscopy (TEM), a second microscopy technique which reveals the structure of elastic networks in the dermis. A beam of electrons is transmitted through an ultra-thin specimen, which the electrons interact with as they pass through. From this interaction an image is formed; that image is magnified and focused onto a fluorescent

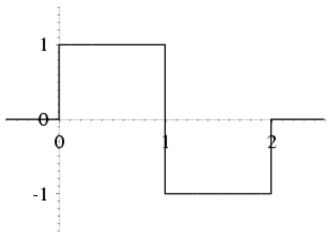


Figure 1: The Haar wavelet

screen on a layer of photographic film. Another option, computed tomography (CT) scans use digital geometry processing to generate a three-dimensional image of the inside of an object from a large series of two-dimensional X-ray "slices" taken around a single axis of rotation. Multi-frequency electrical impedance tomography can create an image of a lesion by recording and analyzing the resistance with which electric waves of different frequencies move through it. However, none of these methods are plausible without a professional budget. A common economical technique is to use video cameras that can be controlled and parameterized in real time for a more three-dimensional and detailed image with relatively high color constancy. Although this is a relatively inexpensive method, neural networks are only useful on the type of images on which they were trained, so for an application intended to be used by the general public the most readily availble imaging devices are high-resolution, low-distortion cameras. Unfortunately, they don't account for color constancy, and it's difficult to get a large, centered, focused, and detailed image of the lesion, however it is the most plausible option.

Among past machine learning approaches to medical image recognition, the use of boosted Haar features is relatively common. Haar-like features are digital image features used in the field of object recognition, named for their intuitive similarity to the Haar wavelet (see figure 1). Wavelets are sets of non-linear bases used to project a function. The Haar wavelet is the simplest possible wavelet, which can be transformed into any other wave. A Haar-like feature categorizes an image into subsections based on the difference between summed pixel intensities in adjacent rectangular regions at a specific location in a detection window. For instance, in the facial recognition problem, Haar wavelets are used to distinguish eyes, which are commonly shadowed, from cheeks, which are commonly the lightest part of the face. The key advantage of a Haar-like feature over most other features is its calculation speed. If a summed area table (also known as an integrated image) is used, a Haar-like feature of any size can be calculated in constant time. A summed area table is a quick and efficient algorithm to sum values in a rectangular subset of a grid. As explained in the paper "Retreival and Ranking of Biomedical Images using Boosted Haar Features" [?], a 2-dimensional decomposition of an image with n^2 pixels yields n^2 wavelet coefficients corresponding to a distinct Haar wavelet. It looks at 4 edge features, 8 line, and 2 center-surround features, as shown in figure 2. Several popular algorithms utilize Haar-like features, such as the Viola-Jones object detection framework. Proposed in 2001 as a solution to the facial recognition problem, it was the first framework to provide competitive object detection rates in real time. It is implemented in OpenCV as cvHaarDetectObjects(). In the detection phase, a target sized window is passed over the input image, and for each subsection of the image the pixel intensities are summed, thus calculating the Haar-like feature. The difference is then compared against a learned threshold that distinguishes objects from non-objects. Due to its simplicity, a Haar-like feature is only a weak learner or classifier; its detection quality is only slightly better than random guessing. Therefore either a large number of Haar-like features or a boosting algorithm to strengthen the most representative ones are necessary to describe an object with sufficient accuracy.

Boosting is a machine learning meta-algorithm implemented alongside supervised learning algorithms to transform a set of weak learners into a single strong learner. Meta-algorithms are iterative optimization algorithms that try to improve a candidate solution (a member of the set of all feasible solutions to the given problem), with

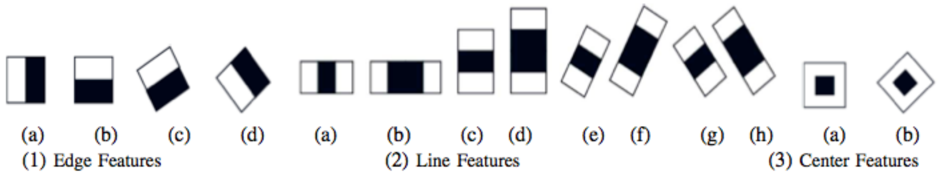


Figure 2: Haar features

regard to a given measure of quality. Weak learners are classifiers that perform slightly better than random guessing, while strong learners are strongly correlated with the true classification. In the aforementioned paper, Haar-like features are used in conjunction with the AdaBoost (adaptive boosting) algorithm to increase their detection power. The AdaBoost algorithm predisposes subsequent classifiers built in favor of instances misclassified by previous classifiers. In each round, it generates and calls a new weak classifier, which updates the distribution of weights to indicate the importance specific data for the classification by increasing the weights of incorrectly classified examples and decreasing that of correctly classified ones. This causes the new classifier to focus on the more heavily weighted former. In the paper, each image category is trained separately, then the weights and weak classifiers are stored. Next, the AdaBoost algorithm (Algorithm 1) takes several weak classifiers given by Haar-like features and develops them into stronger models after a number of iterations, then the highly selective features that minimize the classification error are extracted. The AdaBoost algorithm was the first boosting algorithm to gain prominence. Since then, LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost, and more have had success as well. Some of these have built-in implementations in OpenCV.

A complimentary image processing method to boosted Haar features is Gabor filters. A Gabor filter is a linear filter used for edge detection. The filter has a real and an imaginary component representing orthogonal directions. Like Haar wavelets, the Gabor filters are self-similar; all filters can be generated by dilation and rotation of one basic wavelet. They are similar to the human visual system in their representation

Algorithm 1 AdaBoost algorithm

Input: Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.

Output: Trained model $H(x)$

Algorithm:

```
INITIALIZE weights
    set $D_1(i) = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$
TRAIN the network
    for each t in T:
        $D_t(i) = \frac{D_{t(i)}}{\sum_{j=1}^{N} D_t(j)}$
        for each feature j:
            $h_t = argmin_{h_j} \epsilon_j = \sum_i D_t(i) \cdot I[y_i \neq h_j(x_i)]$
        set $\alpha_t = \frac{1}{2} log \frac{1-\epsilon_t}{\epsilon_t}$
        update $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
    The final strong classifier is:
        $H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$
```

- m is the number of negatives and l is the number of positives
- D_t is a probability distribution
- $h_j(x) = \begin{cases} 1 & p_j f_j(x) \leq p_j \theta_j \\ 0 & otherwise \end{cases}$

of frequency and orientation. In fact, simple cells in mammalian visual cortexes can be modeled by Gabor functions. Their strength lies in texture representation and discrimination rather than object recognition. This is useful for the problem of skin lesion classification, as abnormal texture can be an indication of melanoma. Open-source libraries for Gabor wavelet feature extractions exist, however they are predominantly targeted at facial recognition rather than medical image classification.

3 Theory of neural computation

3.1 An introduction to neurocomputing

Before discussing artificial neural networks, there are a few central concepts one must be familiar with to understand the field of neurocomputing. First, it is necessary to understand enough neuroscience to comprehend why certain neuroscientific models work in a way to make certain types of approximations, and in which fields these approximations are more and less accurate. Second, it is necessary to be acquainted with enough fundamental math and coding to understand the implementation of these models. Next, it is necessary to understand enough cognitive science to have some idea about what the brain is supposed to do, and how it does it. Our brains are not all-purpose computers; they are powerful at some tasks and weak at others. Their power resides in effective biological preprocessing, the use of memory in place of computing power, and effeciency with a small number of operations. Finally, with this knowledge, it is necessary to assume that it's possible to make meaningful simplifications of some aspects of the human nervous system. Some would argue that neurobiology is intrinsically too complicated to simplify, or that we don't know enough about its mechanism to make correct generalizations, which may actually be true, but neuroscientific models are otherwise baseless.

With this inherent assumption, neurocomputing is most basically defined as brain-like computation; while there are many ways to organize a computing system, neurocomputing is an attempt to build computers that are designed like the brain, in hopes of emulating it. Brains have both strengths and weaknesses, like any other computational mechanism. They're adept at pattern recognition, motor control, perception, flexible inference, intuition, and educated guessing, however they are slow, imprecise, make erroneous generalizations, are prejudiced, and are usually incapable of explaining their actions. All of these properties, desirable or not, may be typical of neurocomputing. Biology is the practical science of what can be done with the set of resources available, not the theoretical ideal. Because of this, biological neural nets are a wealth of information in the practical engineering and economic sense as well as the computational sense, providing a naturally selected example of optimization over available resources.

3.2 Biological neural networks

Biological neural networks are populations of interconnected neurons whose inputs or signalling targets define a recognizable circuit. They are the structure through which our brains process informtion.

Our brains have roughly 100 billion neurons. Neurons are made up on the input end of dendrites, which branch out in a tree-like manner from the cell body, also known as the soma. Extending from the soma is a long, thin projection known as the axon, the transmission line of the neuron. The axon can give rise to collateral branches, forming a vast, interconnected network. When axons reach their final destination, they branch again into a structure known as the terminal arborization. At the ends of the terminal arborization are synapses, comprising the output end of the neuron. Dendrites receive inputs from other cells, the soma processes the inputs then transmits information along the axon to the synapses, which give output that is received by other neurons via neurotransmitters diffused across the synaptic cleft, or empty region between the synapses of one neuron and the dendrites of another.

Each neuron is connected to thousands of other neurons which communicate via electrochemical signals. It continuously receives signals from these other neurons and then sums up the inputs according to some process. If the end result is greater than some threshold value, the neuron fires by generating a voltage, known as an action potentional, that transmits down the axon to the terminal arborization. This response is an all-or-nothing binary; there either is action potential or there is not. After firing, a neuron has both an absolute and a relative refractory period during which it cannot fire again. For the former, there is no action threshold nor subsequent possibility of firing, but for the latter, the threshold is merely elevated. This relative refractory period, also known as synaptic resistance, is adaptable, which can cause modified or “learned” behavior of the neuron in which firing frequency will drop if a stimulus is maintained. This is known as Hebb’s learning rule: if the input of a neuron is repeatedly and persistently causing the neuron to fire, a metabolic change occurs in the synapse of that particular input to reduce its resistance. These varying, modifiable resistances give rise to detailed interactions among the web of neurons that are paramount to the nature of computation that neural networks perform.

Biological pressues on neural networks enforce three basic ground rules: each human begins with roughly 100 billion neurons, no new neurons can be created, and neurons must earn their existence or die. They are metabolically expensive, and thus biological pressure dictates that as few as possible must be used.

There are three different types of neurons: sensory, motor, and central. Sensory neurons are activated by physical stimuli and send signals that inform the central nervous system of the state of the body and the external environment. Motor neurons connect muscles and effector organs to the nervous system so that they can be controlled. Central neurons make all of their input and output connections with other neurons.

Interactions between these types of neurons are what cause our brains to function, distinguishing animals from plants. They constantly and tirelessly perform not only our cognitive functions but also automatic and unconscious ones, such as response to pain stimulus and keeping us breathing. Every action and thought is a result of learned interactions between neurons in an incessantly flowing network spanning our whole body, generating our perception of the world and determining our behavior.

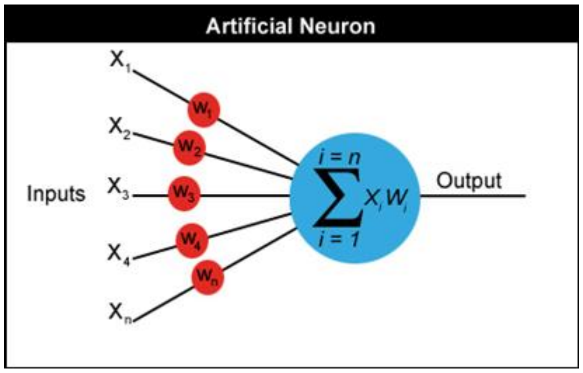


Figure 3: Artificial neuron

3.3 Artificial neural networks

An artificial neural network (ANN) is a computational model that either abstracts or is inspired by the structure and/or functional aspects of biological neural networks. It is a network of simple processing elements that exhibits complex global behavior determined by adaptive weighted connections between processing elements and element parameters. In it, neurons are represented as nodes, and synapses as weighted connections between nodes. Nodes recieve input, apply some sort of summation function, and then output along adjacent weighted connections according to a comparison of the input against some threshold, as illustrated in figure 3. These nodes and connections form a network that learns by means of a modifying algorithm that changes the structure of the network based on external or internal information to produce a desired signal flow so that eventually a certain input yields a certain known output; meaning the network performs some function.

While ANNs typically abstract key characteristics of biological neural networks, there are some ways in which the biological model can be improved by technology, since computers are not constrained to biological pressures. For instance, we are capable of creating and maintaining as many neurons as we need (although most successful ANNs to date use on the order of hundreds or thousands of nodes rather than 100 billion). Also, computers are also never innacurate or fatigued (this does not mean that an artificial neural network can't be wrong or slow, but that each individual step comprising the process won't be), and can perform operations very fast. Some purists reject the omission of modeled biological pressures, thinking them to be a strong factor in the computing strength of neural networks, however much more often they are ignored. On the whole, the trend of modern software implementations of artificial neural networks has been largely to abandon the approach inspired by biology for a more practical approach based on statistics and signal processing. Instead, components of neural networks are used as part of a larger system with both adaptive and non-adaptive elements.

3.4 Types of problems artificial neural nets are best suited for

Wolpert and Macready's "No Free Lunch" theorem (1997) states that all optimization algorithms are equivalent when their performance is averaged across the entire possible problem set. While the explanation for this theorem is mathematically technical and non-intuitive, it is an easily derived consequence of theorems they actually proved, and thus is generally accepted as mathematical fact.

Since we know that not all optimization algorithms perform equally well on every problem, we can infer that each optimization algorithm must perform better on some problems than on others. There can be no globally optimal algorithm for solving optimization problems because on average, performance comes out to be the same. Therefore, each algorithm has strengths and weaknesses, and individual problems have algorithms that will work better to solve them than others.

Obviously, this indicates that problems exist for which artificial neural networks are the most efficient approach. However, practical applications for ANNs have been slow in presenting themselves. In computers, hardware largely determines the software that runs efficiently on it. While every computer is inherently a programmable electronic device that can perform binary arithmetic and logic functions, much more goes into our modern devices. Computers suited to adaption and learning have only been in existence for the last decade or so, able to handle behavioral change, changes in internal functioning, and potential instability. This problem can be perceived as a parallel to the way in which we and chimpanzees differ. While 99% of our DNA is the same, the 1% difference is predominantly in genes which modify sizes and shapes of brain structures. While we contain no radical new hardware over chimps, rearrangements, inflations, contractions, and other modifications of the same physical structure has made our brains capable of considerably more computational power and flexibility.

It is important not to lose sight of this evolutionary perspective when considering ANNs. Neural networks, like brains, cannot compute everything. Their performance is highly detail dependent, arriving at reasonable solutions only for a limited and specific set of problems. The practical applications of neural nets are in the same problems selective pressure has caused animals to biologically adapt to solve: association, classification, categorization, generalization, rapid response, and quick inference from inadequate data.

In the brain, functions are performed collectively and in parallel rather than with a clear delineation of subtasks to which various neurons are assigned. This is a fundamental distinguishing property of artificial neural nets, and the reason why they are so useful for time-sensitive tasks. Aside from speed, the utility of artificial neural network models lies in the fact that they can be used to infer and employ a function from a given set of observations, particularly in applications where the complexity of the data or task makes the design of such functions by hand impractical.

ANNs are primarily used to model complex relationships between inputs and outputs or to find patterns in data. The most successful applications of neural nets to date are: function approximation or regression analysis including time series prediction and modeling, classification including pattern recognition, novelty detection, and sequential decision making, and data processing including filtering, clustering, blind signal separation, and compression.

3.5 Why an ANN is a good fit for this problem

Medical diagnosis of melanoma requires recognition of cancerous lesions through images, which falls under the theme of classification through pattern recognition.

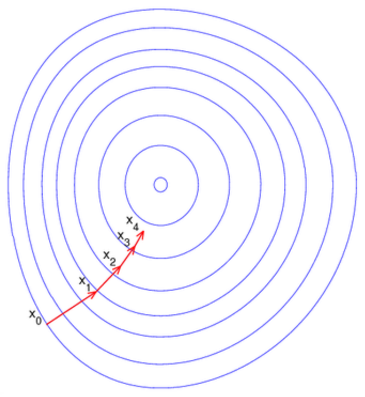


Figure 4: Gradient descent

4 Types of neural networks

4.1 Necessary concepts

To understand the difference between various types of neural nets, you first must be familiar with several concepts.

The first is supervised and unsupervised learning. For all neural nets, a training set of data is necessary to teach the network the connection weights that generate the desired flow. Usually, we know the appropriate output for the given data set, and can use that information to calculate then minimize the discrepancy between the actual and ideal output. Using that knowledge to appropriately modify and train the network is called supervised learning. Sometimes, however, we don't know the appropriate output, and instead require the system to take the input data and somehow organize it in an appropriate and unknown way. Methods to suit these needs are much more difficult to construct and use, however can be very rewarding. These methods utilize unsupervised learning.

The second concept is gradient descent. Gradient descent is a first-order optimization algorithm that finds a local minimum of a function. It takes steps proportional to the negative of the gradient of the function at the current point to descend towards the minimum more quickly (see figure 4). Sometimes an approximate instead of exact gradient is used to further speed the process. It is important to notice that gradient descent does not necessarily converge to the absolute minimum. The minimum it converges on is based on the starting values. Gradient descent is based on the observation that if the multivariable function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point a, then $F(\mathbf{x})$ decreases fastest if one moves from a in the direction of the negative gradient of F at a, $-\nabla F(\mathbf{a})$. It follows that, if $\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$ for step size $\gamma > 0$ a small enough number, then $F(\mathbf{a}) \geq F(\mathbf{b})$. With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of F, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ such that $\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0$ (note that the value of the step size γ is allowed to change at every iteration). Thus, we have $F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots$, which shows that the sequence $(\mathbf{x}_n)$ converges to a local minimum (nearest to x_0).

The last and possibly most important concept fundamental to neural nets is Bayesian probability. In general, probability is viewed in the *classical* or *frequentist* interpretation, where it is defined in terms of the frequencies of random, repeatable events. In contrast, *Bayesian* probability provides a quantification of uncertainty, or a measure of a state of knowledge. It's an extension of logic that enables reasoning with uncertain statements whose outcomes we cannot repeatedly measure. To evaluate the probability of a hypothesis, some prior probability must first be specified which is updated in light of new relevant data.

For instance, we speak of the probability of events such as the earth being destroyed or a person dying. These are not repeatable events from which we can determine a probability through measures of frequency. However, they still have a likelihood of occurrence that we can educatedly guess through consideration of relevant evidence, such as the frequency of asteroids that have come close or the person's consumption of carcinogens and participation in hazardous activities. When we receive new evidence, we can process it and appropriately reassess our hypothesis. Bayesian probability allows us to quantify such expressions of uncertainty.

In 1946, phycisist Richard Cox showed that if we numerically quantify degrees of belief (such as those used in Bayesian probability), then a simple set of axioms which encode common-sense properties of these beliefs can be used to intuitively construct a set of rules for manipulating degrees of belief that are equivalent to the sum and product rules of classical probability. In 2003, statistical theorist Edwin Thompson Jaynes used Cox's axioms to prove that probability theory could be conceived as an extension of Boolean logic to situations involving uncertainty. Over the years, many mathematicians have proposed various sets of axioms similar to Cox's that all have behaved precisely according to the rules of probability.

The central idea of Bayesian probability is defined in Bayes' theorem:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

Where D is the set of observed data and w is the event whose uncertainty we wish to numerically express. This equation expresses the uncertainty of the event w depending on the set of observed data D as the product of the uncertainty of the evidence as a probability distribution over w times the prior probability distribution of w, scaled to make sure it integrates to one and is therefore a valid probability density. In other words, the posterior probability is proportional to the likelihood times the prior probability distribution, where all of these quantities are viewed as functions of w.

An advantage Bayesian probability holds over classical probability is that the inclusion of prior knowledge arises naturally. For instance, classical probability would assign a fair coin a 100% probability of landing on tails if it does so three times of three, whereas a Bayesian likelihood would include the knowledge that it's a fair coin. However, the likelihood the Bayes' theorem assigns is highly dependent on the prior

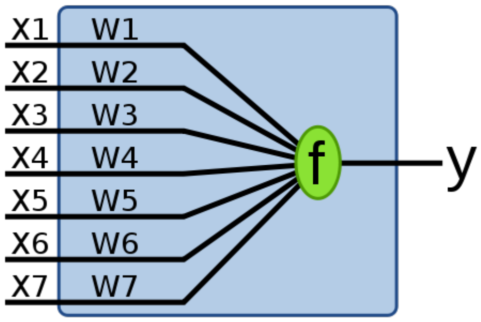


Figure 5: The perceptron function

distribution used, which is a sizeable weakness. Bayesian methods based on poor choices of prior distributions can yield poor results with high confidence.

Despite this, Bayesian principles are still very popular in pattern recognition and other machine learning methods, as the ability to incorporate evidence during progression is hugely advantageous.

4.2 The perceptron

The perceptron was the first influential model of a simple artificial neural network, proposed by psychologist Frank Rosenblatt in 1958. It is the simplest kind of feedforward neural network, functioning as a linear classifier (feed-forward means that there are no cycles or loops, the connections only move forward). Because it is solely a linear discriminant model, it works only when two patterns are linearly separable, meaning a hyperplane can be drawn in the problem space in such a way to completely separate the two classes.

The perceptron function is defined as:

$$f(x) = \begin{cases} 1 & w * x + b > 0 \\ 0 & otherwise \end{cases}$$

Where x is the input vector, $f(x)$ is the output vector, w is the weight vector, and b is the bias. The bias is a constant value that serves to alter the position (not orientation) of the decision boundary so that the threshold value is 0. This function is a binary classifier where 1 indicates a positive case and 0 represents a negative one.

The perceptron's method of learning is described by the perceptron learning algorithm. It uses knowledge of past results to modify the weights of the connections, thereby improving the performance of the network. This is conceptually similar to Bayesian probabilities, in which likelihoods can be updated to reflect new data. The perceptron learning algorithm is shown in Algorithm 2.

Note that the weights are adapted immediately for each pair instead of at once for the entire vector, and also that weights are only updated if there is error, i.e. only when learning will occur. The perceptron convergence theorem states that if there exists an exact solution, the perceptron learning algorithm is guaranteed to learn the classification in a finite number of steps. If there is not an exact solution and the training data is not linearly separable, the algorithm will run indefinitely without converging. Unfortunately, many interesting classifications are not linearly separable and require a more complex decision surface to separate the classes.

The perceptron has an input sensory layer called a retina, which is partially connected to an association layer. It is important theoretically that these layers are not fully connected; this establishes that each unit in the association layer computes a different function of the image on the retina. The association layer is then reciprocally connected to a response layer by way of unidirectionally modifiable weighted "synapses" (in the direction of the response layer). The goal of the perceptron is the activation of a single appropriate unit in the response layer for a certain set of inputs.

Algorithm 2 Perceptron Learning Algorithm

Input: Given a training set $D = \{(x_1, d_1), \ldots, (x_n, d_n)\}$

Output: Trained model $y = f(x)$

Algorithm:

```
INITIALIZE the weights and the threshold
    set $w_0 = -b$
    set $x_{j,0} = 1$
    set $w_i(0) =$ small random values
TRAIN the network
    for each sample j in D:
        calculate the actual output
          $y_j(t) = f(w(t) \cdot x_j) = f(w_0(t)x_{j,0} + \ldots + w_n(t)x_{j,n})$
        adapt the weights
          for each node i in n:
            $w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i}$
    repeat until iteration error $d_j - y_j(t) < \gamma$
```

- $y = f(x)$ is the perceptron's output for an input vector x
- b is the bias
- $D = \{(x_1, d_1), \ldots, (x_n, d_n)\}$ is the training set where
  - x_j is the input vector, and $x_{j,i}$ is the value of the i^{th} node of the j^{th} input vector
  - d_j is the desired output value of x_j
- w is the weight vector
  - w_i is the i^{th} value of the weight vector, to be multiplied by $x_{j,i}$
  - $w_i(t)$ is the i^{th} weight at time t
- α is the learning rate, where $0 < \alpha \leq 1$ (a learning rate set too high will periodically oscillate around the value of the solution)
- γ is a user-specified error threshold

The activation of any appropriate unit is known as a candidate solution, or a member of the set of possible solutions for a given problem.

The basic computing element of the perceptron is an artificial neuron known as the *threshold logic unit*, or TLU, comprised of n input weights with strengths $w[i]$. It sums the product of the input weights and their respective strengths and nonlinearly transforms them into binary values indicating activation if the scaled summed input is over a certain threshold (see figure 3 on page 18). The activation function is binary; either a correct unit is activated or it is not. There is no graded activation.

Although its learning capabilities are extremely limited, the perceptron proved to be a valuable tool in prediction and explanation of human cognition. Interestingly, the limitations that make it less effective as a computing device are very similar to those of humans, and as such it was widely used among psychologists and cognitive scientists to help form many theories about the workings of the brain. Along with the ADALINE (an early gradient descent algorithm using a non-binary error function), the perceptron framed the network learning problem in a way that was fundamentally accepted and unconsciously integrated into scientific thought, shaping the evolution of computational science and future network models for years.

4.3 The multi-layer perceptron

The multi-layer perceptron (MLP) is a feed-forward artificial neural network model consisting of multiple fully connected layers of nodes in a directed graph that maps sets of input data onto a set of appropriate output. It is proven to be a universal function approximator by the *universal approximation theorem*, otherwise known as the *Cybenko theorem*. A trademark of the multi-layer perceptron is that it does not have a connection to biological neural networks like other types of ANNs.

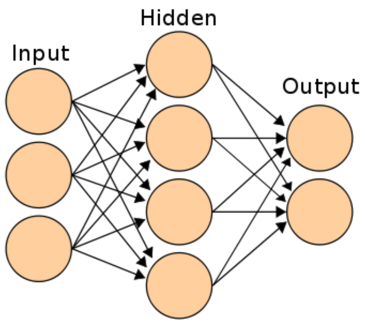


Figure 6: Multi-layer perceptron

MLPs are comprised of multiple stages of processing; an input layer, an output later, and at least one hidden layer (referred to as such because their function is obscured from sight). Each layer individually resembles the perceptron model. The difference lies in the full connectedness and the intermediate units' use of continuous sigmoidal nonlinearities instead of step functions. This type of processing means that the neural network function is differentiable with respect to the network parameters, a fact which plays a central role in network training. If the hidden units used linear activation functions, the network could be simplified to an input, association, and output layer, i.e. a perceptron (because the composition of successive linear transformations is itself a linear transformation).

Each node of an MLP has a non-linear (sigmoidal) activation function meant to model the frequency of action potentials of biological neurons instead of the simplified binary function. The two most common activation functions are the hyperbolic tangent function $\Phi(y_i) = \tanh(v_i)$ which ranges from $(-1, 1)$, and the logistic function $\Phi(y_i) = (1 + e^{-v_i})^{-1}$ which ranges from $(0, 1)$, where y_i is the output of the i^{th} node and v_i is the weighted sum of the input synapses.

MLPs use backpropogation, a supervised learning technique, to train the network. Backpropogation is an efficient technique for evaluating the gradient of an error function for a feed-forward neural net, which is the first step in most training algorithms. It can be perceived as a local "message passing" scheme in which information is sent both forwards and backwards through the network. Backpropogation is dependent on the activation function being differentiable, which is a key feature of multi-layer perceptrons. Learning in the network occurs by modifying the connection weights by a factor calculated through gradient descent after a piece of data is processed, so that the change is based on the amount of error generated compared to the expected result.

```
Initialize the weights in the network (often randomly)
Do

    For each example e in the training set


        O = neural-net-output(network, e) ; forward pass
        T = teacher output for e
        Calculate error (T - O) at the output units
        Compute delta_wh for all weights from hidden layer to output layer ; b
        Compute delta_wi for all weights from input layer to hidden layer ; ba
        Update the weights in the network

Until all examples classified correctly or stopping criterion satisfied
Return the network
```

The first phase of network training is propagation. The training data is propagated forward through the network, generating output activations by applying the selected sigmoidal function to the weighted sum of the inputs. The input vector x_n is applied

Algorithm 3 Multi-layer perceptron learning

Input: Given a training vector x

Output: a trained model

Algorithm:

```
Apply an input vector $x_n$ to the network and forward propagate through the netw
    $a_j = \sum_i w_{ji} z_i$
    $z_j = h(a_j)$
Evaluate the errors for all the output units
    $d_k = y_k - t_k$
Backpropagate the errors to obtain errors for each hidden unit in the network
    $d_j = h'(a_j) \sum_k w_{kj} d_k$
Evaluate the required derivatives
    $E_n = \frac{1}{2} \sum_k d_k^2$
    $\frac{dE_n}{dw_{ji}} = d_j z_i$
```

- a_j is the weighted sum of the inputs of unit j
- z_i is the activation of a unit that sends a connection to unit j
- w_{ji} is the weight associated with that connection
- h is the nonlinear activation function
- d_k is the error associated with the output units
- y_k is the network output value, a linear combination of input variables
- t_k is a value specified by the error function
- E_n is the summed error function

to the network and propagated forward using $a_j = \sum_i w_{ji} z_i$ and $z_j = h(a_j)$ where z is the activation, w is the weight, h is the nonlinear activation function, and a is the weighted sum of the inputs. The error d_k is then calculated for all the output units using $d_k = y_k - t_k$ where y is the output (a linear combination of the input variables), and t is a value specified by the error function. After this, backward propagation of the output activations through the ANN occurs using the target output vector as a starting point. The algorithm backpropogates using $d_j = h'(a_j) \sum_k w_{kj} d_k$ to obtain the error for each hidden unit in the network. The value of d for a particular hidden unit can be obtained by propagating the previous values of d backwards from units higher up in the network. Because we already know the error values for the output units, we can evaluate them for all of the hidden units in a feed-forward network by recursively applying the equation above. The equation $\frac{dE_n}{dw_{ij}} = d_j z_i$ is used to evaluate the required derivatives where E_n is the summed error function, w_{ij} is the weight, z_i is the activation of unit i and d_j is the error.

The second phase of training is weight update. This is done using the error values d_k calculated in phase one. For each weighted synapse you must first multiply d_k and the input activation to get the gradient of weight, then change the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight.

Phases one and two are repeated until the desired accuracy is achieved.

There are practical limitations to the performance of MLPs such as requiring inputs to be scaled and normalized, and slow convergence that is not guaranteed. Because gradient descent may converge to any local minimum on the error surface, the MLP may not reach the absolute minimum. Despite these limitations, multi-layer perceptrons using a backpropogation algorithm are the standard algorithm for any supervised-learning pattern recognition process. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to yield approximate solutions for extremely complex problems.

4.4 Restricted Boltzmann machines

4.5 The deep belief network

Deep learning is a recently explored field of machine learning based on algorithms that can learn multiple levels of representation of data in order to model more complex relationships. Data features are arranged in a hierarchy known as a deep architecture, in which higher-level features and concepts are defined in terms of lower ones. Deep learning algorithms have proven to be skilled at feature extraction in high-dimensional, structured data, and can implicitly learn the data's distribution function. Most of these algorithms are unsupervised, and thus are very useful at finding patterns that humans might not arrive at due to prejudices or predilections.

Deep belief networks (DBNs) are probabilistic generative models whose structures are composed of multiple layers of stochastic latent (deterministic and present, although not visible) variables typically having binary values. The top two layers have mutual undirected symmetric connections and form an associative memory. The lower layers get top-down, directed connections from the layer above. The states of the units in the lowest layer represent a data vector, as in multi-layer perceptrons. Typically, DBNs use a logistic function of the weighted input received from above (or below) to determine the probability that a binary latent variable has a value of 1 during top-down generation (or bottom-up inference), but other types of variables can be used as long as the variables are in the exponential family (so that the log probability is linear in the parameters).

The two most significant properties of deep belief networks are an efficient layer by layer procedure for learning top-down, generative weights which determine how variables in a layer depend on the state of the layer above, and that after learning, the values of latent variables in every layer can be inferred by a single bottom-up pass starting with the observed data vector and moving backwards using the generative

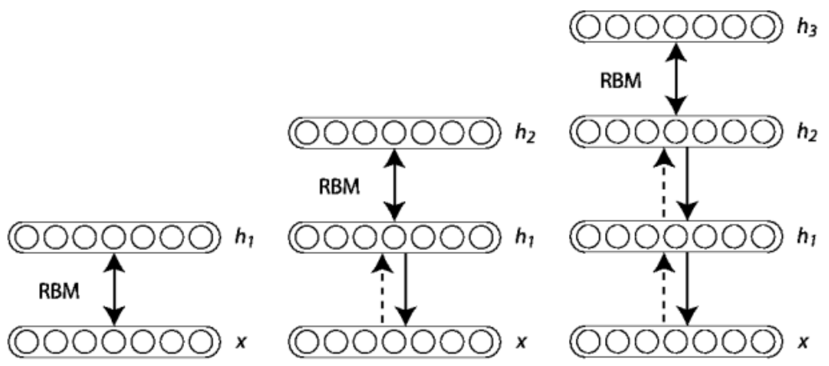


Figure 7: Deep belief network

weights. This means that you can add a final layer of variables (composing the desired output) and backpropogate the error derivatives, as in the MLP. Backpropogation will work better if feature detectors in the hidden layers are initialized by learning a deep belief network that models the structure of the input data.

DBNs are learned one layer at a time, using the values of the latent variables in one layer as the training data for the next layer. It's a greedy algorithm, optimizing at each step for that step alone with no foresight.

You can view a DBN as a composition of single learning modules, each of which is a type of restricted Boltzmann machine with a layer of visible units (representing data) and hidden units (representing features that capture higher-order correlations), whose layers are connected by a matrix of symmetrically weighted connections. There are no connections within a layer. They can be represented by the function $p(v) = \sum_h p(h|W)p(v|h, W)$ where W is matrix of weights, v is a vector of activities for the visible units, and h is a sample vector of hidden units.

Deep belief networks are best at generating and recognizing images, video sequences, and motion-capture data. Very importantly, they can be combined with other learning procedures to fine-tune the weights. This makes them very useful tools for machine learning, and particularly for data representation.

5 Considered and selected approaches

5.1 Questions of intent

Prior to any programming undertaking, you must decide on answers to a few key questions that will shape your approach to the solution. Each directing answer leads to a new set of questions, creating a candidate map along the many roads to resolution.

In regard to my thesis, the first crossroad was: how am I going to approach the problem of medical image recognition and what is my focus? There were many directions I could have gone; a focus on research, a focus on comparative performances of methods, a focus on a working product, a focus on an optimized product. Each had their merits and setbacks. For instance, while researching different algorithmic approaches and comparing their past performances would be beneficial to my general knowledge of algorithms and machine learning, it wouldn't improve my programming skills. Contrastly, generating a finished, working product would give me a plethora of practical experience, but my knowledge of machine learning would be more deep than broad. In the end, I chose practical knowledge over theoretical for the two facts that a finished product is beneficial as a display of my abilities for future employment opportunities, and that I prefer programming to research.

This verdict left me at another fork: should I write the program from scratch or make use of open source code? Programming from the ground up could result in a unique library geared towards machine learning and leave me with a very thorough understanding of each intracacy of the process. I'd conclude with a wealth of experience in algorithms and software development. However, creating a strongly associative classifier for data as complex as skin lesion images is most likely beyond the scope of an undergraduate thesis, and would presumably take longer than two semesters. My other option, utilizing preexisting software and libraries written by those who have successfully conquered aspects of the problem as building blocks, actually poses the same design challenges and requires a comparable level of comprehension of the algorithms, while at the same time being much more applicable to the world of software development. In a company vying for profit, time is valuable and none is wasted rewriting existing code. The time saved can be put to use revising and optimizing, allowing companies to focus on optimal functionality. Not only is this is highly advantageous in my situation, a single programmer and a complex problem, but employers are more likely to be impressed by functionality than determination. Additionally, identification of melanoma is a relevant problem and the software could have present and propritious applications. Obviously, it is both more responsible and reasonable to follow the route of functionality and marketability.

With that settled, the next task would usually be to determine the main algorithmic approach. However, from the start I wanted to explore artificial neural networks, so this was not a step for me. As I outlined previously, artificial neural networks function very well on classification problems. However, other machine learning algorithms exist that are also capable of classification. These include quadratic classifiers, decision trees, Bayesian networks, and hidden Markov models.

5.2 Preprocessing techniques

While neural networks are flexible in their functionality, passing them an unaltered image is unlikely to be the best option. Typically, preprocessing techniques are used to convert the image into a data representation of its key features which in turn the network is trained on. There are many methods through which this can be done.

Regardless of method, the first step of preprocessing is to normalize the image. The image must be centered and scaled to an appropriate (ideally uniformly relative) size and level of detail while retaining information about the diameter, lighting, and color. Usually, this involves first calculating data that the process of normalization alters, then cropping and magnifying the image to faciliate feature extraction. at the same time, it is possible to equip the input layer of a neural network with a sort of virtual retina that can determine features for itself, or any range of supplementation of the retina with numerical information extracted during preprocessing. On top of this, there are various ways of representing the extracted data to the network. The most common method is a binary vector where each input represents either the presence or absence of a feature. While assigning these nuanced variables to a binary does not account for the variability of feature representation and may diminish accuracy for a small set of selective features, most of the methods I've researched use a large set of selectors which reduces the amount of loss. Also, binary responses are inherently characteristic of neural networks, and they are excellent classifiers in spite of the apparent lack of nuance.

5.3 Selection of classifiers

Before you can begin feature extraction, you must choose the method according to which you select the classifiers. Many established schema for classifier selection already exist. Most of these schema numerically assign values for certain dermoscopic criteria, falling into the categories of local and global features. Global features allow an expeditive preliminary categorization of a skin lesion, following which local features are used to assess the lesion with detail. Local features can be thought of as the "letters of the dermascopic alphabet." ([2]) They consist of pigment networks, dots and globules, streaks, a blue-whitish veil, pigmentation, hypopigmentation, regression structures, and vascular structures. Global features, composed by local ones, are reticular, globular, cobblestone, homogenous, starburst, parallel, multicomponent, lacunar, and unspecific patterns.

Pigment networks resemble a thin brown grid superimposed over a light brown background. Dots and globules are the names given to irregular rounded structures in varying colors and sizes. Streaks are brownish-black linear structures of varying thickness, which may be observed throughout the lesion but are not clearly combined with pigment network lines. A blue-whitish veil is a hazy and diffuse pigmentation overlaid on other structures. Pigmentation refers to dark areas which preclude recognition of more subtle dermoscopic features. Hypopigmentation refers to an area of decreased pigmentation within an otherwise ordinary pigmented lesion. Regression structures are white, blue, or both areas resembling either a scar, or speckled peppering. They are sometimes mistaken for a blue-grey veil when the peppering is not thoroughly distinctive. There are seven types of vascular structures: comma vessels, wreath vessels, arborizing vessels, hairpin vessels, dotted vessels, linear irregular vessels, and vessels within regression structures. They become more readily apparent when the tumor is very lightly compressed.

Reticular patterns are the most common in melanocytic lesions, characterized by a pigment network covering most of the surface of the lesion. Globular patterns consist of numerous irregularly distributed globules. Cobblestone patterns are very similar, but the globules are larger and closely aggregated resembling cobblestones. Homogenous patterns appear in the absense of other features as diffuse pigmentation ranging from brown to blue-grey to reddish-black. Starburst patterns appear at the edges of skin lesions and are comprised of pigmented streaks in a radial arrangement. Parallel patterns are exclusive to lesions in glabrous regions and are characterized by a series of parallel pigmented streaks. Multicomponent pattern is the name given to combinations of three or more distinctive dermoscopic structures in a given region. Lacunar patterns are aggregations of red lacunas, rounded structures with smooth borders in either red, blue-purple, or black. When a structure cannot be categorized into any of the aforementioned patterns, it is deemed “unspecific.”

Four of the most well-established and benchmarked methods for classifier selection are the ABCD rule, pattern analysis, Menzies' method, and the seven-point checklist.

5.3.1 ABCD rule (Stolz method)

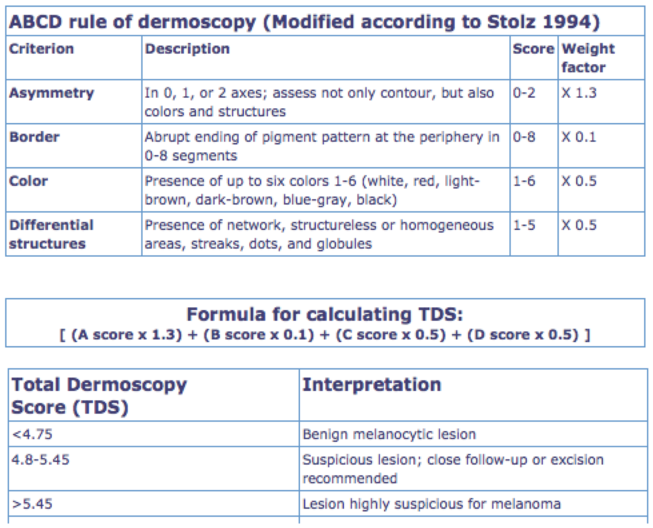
The ABCD method is an algorithmic analogue of the heuristic used by dermatologists to identify melanocytic lesions. Same as the heuristic, it is an acronym in which the A stands for asymmetry, the B for border, C for color, and D for differential structures. It was developed by W. Stolz and co. in 1994, and its reliability was proven by F. Nachbar et al. the same year.[1] It is easily learned and rapidly calculated, providing a reliable, objective, and reproducible diagnosis of melanoma.

The procedure of Stolz's method is to semiquantitatively assses each of the four categories and assign them a numeric score. Each score is scaled by a weight factor, then the sum of these weighted scores yields the total dermatoscopy score, or TDS. In this case, $TDS = 1.3 \cdot A + 0.1 \cdot B + 0.5 \cdot C + 0.5 \cdot D$. A TDS of less than 4.75 is considered benign, between 4.8 and 5.45 suspicious, and greater than 5.45 highly suspicious.

Asymmetry is calculated by bisecting the lesion with two optimally positioned perpindicular axes. No points are awarded if the lesion is symmetric over both axes, one is awarded if it is symmetric over one axis, and two are awarded if neither axis yields symmetry. Symmetry is assessed using colors and structures in addition to shape. At 54% of the TDS, this is the most highly weighted criterion; most melanomas receive a score of two compared to only 25% of benign lesions [2].

The border score ranges from zero to eight, where each point represents a sharp and distinct cut-off in a quarter radian (45^{o}) "pie piece." This is the weakest criterion, accounting for only 4% of the TDS. Most melanomas score between three and eight points, whereas most benign lesions score below three.

[1]In 172 melanocytic lesions (69 melanomas, 103 melanocytic nevi), specificity was 90.3% and sensitivity was 92.8%. [2]



ABCD rule of dermoscopy (Modified according to Stolz 1994)			
Criterion	**Description**	**Score**	**Weight factor**
Asymmetry	In 0, 1, or 2 axes; assess not only contour, but also colors and structures	0-2	X 1.3
Border	Abrupt ending of pigment pattern at the periphery in 0-8 segments	0-8	X 0.1
Color	Presence of up to six colors 1-6 (white, red, light-brown, dark-brown, blue-gray, black)	1-6	X 0.5
Differential structures	Presence of network, structureless or homogeneous areas, streaks, dots, and globules	1-5	X 0.5

Formula for calculating TDS:
[(A score x 1.3) + (B score x 0.1) + (C score x 0.5) + (D score x 0.5)]

Total Dermoscopy Score (TDS)	Interpretation
<4.75	Benign melanocytic lesion
4.8-5.45	Suspicious lesion; close follow-up or excision recommended
>5.45	Lesion highly suspicious for melanoma

[images obtained from www.dermoscopy.org]

Figure 8: Stolz's ABCD rule for identifying melanoma

Color points are awarded one for each color present. The palate consists of light brown, dark brown, red, blue-grey, black, and white (lighter than adjacent skin). Thus the score ranges from one to six, where the majority of melanomas have three or more, and 40% have five to six. Color accounts for 21% of the TDS.

The final criterion is the presence of differential structures. The five candidates are a pigment network, streaks (two or more), dots (two or more), globules (one or more), and structureless or homogenous areas (covering 10% or more of the lesion). A point is awarded for each clearly visible structure, ranging from one to five. Like color, differential structures account for 21% of the TDS.

5.3.2 Menzies' method

Menzies' method is a simple equation, $M = p - n$, where p stands for the number of positive features and n for the number of negative. The higher the score, the more chance of melanoma. The negative features consist of symmetry and a single even color. The positive features consist of a blue-whitish veil, brown dots, pseudopods, radical steaming, scar-like depigmentation, five to six colors, blue or grey dots, and a broadened network. Possible scores range from negative two to eight.

5.3.3 Seven-point checklist

Proposed by G. Argenziano in 1998, the seven-point checklist is a list of seven features selected for their high correlation with melanoma. Three features are considered “major” and recieve two points, while four are considered “minor” and only recieve one. The scores for each feature are summed, and if the score is greater than three the likelihood of melanoma is high. Argenziano found that this method classified melanoma with a sensitivity of 95% and a specificity of 75%, correctly diagnosing 82% of melanocytic lesions.

The major criteria consist of an atypical pigment network, a blue-whitish veil, and an atypical vascular pattern. The minor criteria consist of irregular streaks, irregular pigmentation, irregular dots and/or globules, and regression structures.

Obviously, the point of an approach utilizing an artificial neural net is to establish a new method for diagnosing melanoma. However, these approaches are helpful in suggesting benchmarks against which to compare the network’s performance. Preprocessing will extract information about global and local features present in the lesion, after which the network is expected to reach the same conclusion about the representativeness of certain features over others.

5.4 Software considered

5.5 Practical limitations

Unfortunately, classifier programs are practically bounded by certain limitations, the biggest of which are financial capital, computing power, and manhours. These constraints limit practicable image acquisition techniques, affordable software, and general scope.

The first problem is the acquisition of a training set of images. Since I am already in possession of a set of training data, image acquisition techniques are solely an issue for the practical post-thesis application of my work. Instead, these are the issues that compromise the accuracy of my network's classification.

Camera quality is the main issue I'll be facing with my application. Not only are the majority of input images going to come from low-quality personal cameras (point-and-shoot or camera phones), but my network was trained on medical-quality images which makes the disparity even greater. Commercially available cameras capture poorly lesions less than 0.5 cm in diameter, which is a relatively large lesion to begin with. The magnification is simply not enough to get a properly detailed image at a smaller scale. This means that there might not be enough detail for proper edge or feature detection.

One possible work-around for this issue is to include a user questionnaire regarding features of the lesion, especially those that require greater detail to calculate, then to use the reported information alongside what the camera can detect. However, this will take more code and really should be used with a network properly trained to incorporate the data. It would be simple to run the data through my program and create such a network, however in that case I would need a training data set of this type with known outcomes.

The second issue is lighting. Colors are only true under natural white light. In any other light, our perception of color distorts. Proper color identification is an important aspect of classification. I have no control over the lighting in user-input images. Several possible fixes for this problem include: prompting the user to take pictures in multiple different lightings (florescent, sunlight, etc) or the brightest lighting possible, including an integrated detector that rejects pictures that are too poorly lit, and placing a color bar on the side of the application screen and having the user report the actual color, and then including an algorithm to adjust the image appropriately.

The issue that this raises is a new compromise to accuracy: user error. The best way to counteract this is to include user error in the training set so the network can account for it, however generating or collecting an appropriate data set is not feasible. Unfortunately, there is no solution but to trust the user and make the questionnaire as fool-proof as possible.

Due to time constraints and a being single programmer, I must make use of software that has already been written rather than customizing my own along each facet. Additionally, due to lack of funding, no professional software can be used, only freeware. Of this freeware, some very promising libraries are now inactive due to conflicting retrograde dependencies with host programs.

Despite all these limitations, artificial neural networks remain strong and flexible classifiers, and it remains feasible to produce a high-functioning and highly-correlated network.

5.6 Selected approaches

To be completed after programming.

6 Final product

6.1 Structure of ANN

6.2 Marked-up code

6.3 Results with benchmarks

7 Future (or current) plans/applications

7.1 Phone app

7.2 Web app

References

[1] “Retreival and Ranking of Biomedical Images using Boosted Haar Features”

[2] http://www.dermoscopy.org

List of Figures

1 The Haar wavelet 10
2 Haar features 12
3 Artificial neuron 18
4 Gradient descent 22
5 The perceptron function 25
6 Multi-layer perceptron 29
7 Deep belief network 34
8 Stolz's ABCD rule for identifying melanoma 40