We describe a new learning procedure, back-propagation, for networks of neurone-like units.

ニューロンのようなユニットのネットワークのための新しい学習手順、逆伝播について説明します。

The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector.

この手順では、ネットワークの接続の重みを繰り返し調整して、ネットの実際の出力ベクトルと目的の出力ベクトルの差の測定値を最小化します。

As a result of the weight adjustments, internal 'hidden' units which are not part of the input

or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units.

重量調整の結果として、入力の一部ではない内部の「隠された」ユニット

または出力はタスクドメインの重要な機能を表し、タスクの規則性はこれらのユニットの相互作用によってキャプチャされます。

The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure1.

有用な新機能を作成する機能は、逆伝播を、パーセプトロン収束手順1などの以前の単純な方法と区別します。

There have been many attempts to design self-organizing neural networks.

自己組織化ニューラルネットワークを設計する多くの試みがありました。

The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain.

目的は、任意に接続されたニューラルネットワークが特定のタスクドメインに適した内部構造を開発できるようにする強力なシナプス修正ルールを見つけることです。

The task is specified by giving the desired state vector of the output units for each state vector of the input units.

タスクは、入力ユニットの各状態ベクトルに対して、出力ユニットの目的の状態ベクトルを与えることによって指定されます。

If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors.

入力ユニットが出力ユニットに直接接続されている場合、実際の出力ベクトルと目的の出力ベクトルの差が徐々に小さくなるように接続の相対強度を繰り返し調整する学習ルールを見つけるのは比較的簡単です。

Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task.

実際の状態または望ましい状態がタスクによって指定されていない非表示のユニットを導入すると、学習はより興味深いものになりますが、より難しくなります。

(In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.)

（パーセプトロンでは、入力と出力の間に、入力接続が手動で固定されているため、真の隠れユニットではない「機能分析器」があり、その状態は入力ベクトルによって完全に決定されます。表現を学習しません。）

The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour.

学習手順では、目的の入出力動作を実現するために、どのような状況で非表示ユニットをアクティブにするかを決定する必要があります。

This amounts to deciding what these units should represent.

これは、これらのユニットが何を表すべきかを決定することになります。

We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

汎用で比較的単純な手順は、適切な内部表現を構築するのに十分強力であることを実証します。

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top.

学習手順の最も単純な形式は、最下部に入力ユニットのレイヤーを持つレイヤードネットワークです。任意の数の中間層;上部に出力ユニットのレイヤー。

Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers.

レイヤー内または上位レイヤーから下位レイヤーへの接続は禁止されていますが、接続は中間レイヤーをスキップできます。

An input vector is presented to the network by setting the states of the input units.

入力ユニットの状態を設定することにより、入力ベクトルがネットワークに提示されます。

Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers.

次に、各レイヤーのユニットの状態は、下のレイヤーからの接続に式（1）および（2）を適用することによって決定されます。

All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

レイヤー内のすべてのユニットの状態は並列に設定されますが、異なるレイヤーの状態は順番に設定され、下から始まり、出力ユニットの状態が決定されるまで上に向かっていきます。

The total input, xi, to unit j is a linear function of the outputs, , of the units that are connected to j and of the weights, , on these connections

ユニットjへの合計入力xiは、jに接続されているユニットの出力y\_i、およびこれらの接続の重みw\_jiの一次関数です。

Units can be given biases by introducing an extra input to each unit which always has a value of 1.

ユニットには、常に1の値を持つ追加の入力を各ユニットに導入することにより、バイアスを与えることができます。

The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign.

この余分な入力の重みはバイアスと呼ばれ、反対符号のしきい値に相当します。

It can be treated just like the other weights.

バイアスは、他のウェイトと同様に扱うことができます。

A unit has a real-valued output, , which is a non-linear function of its total input

ユニットには実数値の出力y\_iがあり、これはその総入力の非線形関数(標準シグモイド関数)です。

It is not necessary to use exactly the functions given in equations (1) and (2).

方程式（1）および（2）で与えられた関数を正確に使用する必要はありません。

Any input-output function which has a bounded derivative will do.

有界微分を持つ関数が任意の入出力を行います。

However, the use of a linear function for combining the inputs to a unit before applying the nonlinearity greatly simplifies the learning procedure.

ただし、非線形性を適用する前に入力をユニットに結合するために線形関数を使用すると、学習手順が大幅に簡素化されます。

The aim is to find a set of weights that ensure that for each input vector the output vector produced by the network is the same as (or sufficiently close to) the desired output vector.

目的は、各入力ベクトルについて、ネットワークによって生成される出力ベクトルが目的の出力ベクトルと同じ（または十分に近い）ことを保証する重みのセットを見つけることです。

If there is a fixed, finite set of input-output cases, the total error in the performance of the network with a particular set of weights can be computed by comparing the actual and desired output vectors for every case.

入出力ケースの固定された有限セットがある場合、特定の重みのセットを持つネットワークのパフォーマンスの合計誤差は、すべてのケースの実際の出力ベクトルと望ましい出力ベクトルを比較することで計算できます。

The total error, E, is defined as where c is an index over cases (input-output pairs), j is an index over output units, y is the actual state of an output unit and d is its desired state.

Eは総誤差、cはケース（入出力ペア）のインデックス、jは出力ユニットのインデックス、yは出力ユニットの実際の状態、dはその望ましい状態として定義されます。

To minimize E by gradient descent it is necessary to compute the partial derivative of E with respect to each weight in the network.

勾配降下によってEを最小化するには、ネットワークの各重みに関してEの偏微分を計算する必要があります。

This is simply the sum of the partial derivatives for each of the input-output cases.

これは、各入出力ケースの偏導関数の合計です。

For a given case, the partial derivatives of the error with respect to each weight are computed in two passes.

特定の場合、各重みに関する誤差の偏導関数は2つのパスで計算されます。

We have already described the forward pass in which the units in each layer have their states determined by the input they receive from units in lower layers using equations (1) and (2).

式（1）および（2）を使用して、下位層のユニットから受け取る入力によって各層のユニットの状態が決定されるフォワードパスについては既に説明しました。

The backward pass which propagates derivatives from the top layer back to the bottom one is more complicated.

導関数を最上層から最下層に戻す逆方向パスはより複雑です。

The backward pass starts by computing aE/ay for each of the output units.

バックワードパスは、各出力ユニットのを計算することから始まります。

Differentiating equation (3) for a particular case, c, and suppressing the index c gives

特定のケースcについて方程式（3）を微分し、インデックスcを抑制すると、

We can then apply the chain rule to compute aE/ax

その後、チェーンルールを適用してを計算できます。

Differentiating equation (2) to get the value of dyi/ dx, and substituting gives

方程式（2）を微分しての値を取得し、代入して

This means that we know how a change in the total input x to an output unit will affect the error.

これは、出力ユニットへの合計入力xの変化がエラーにどのように影響するかを知っていることを意味します。

But this total input is just a linear function of the states of the lower level units and it is also a linear function of the weights on the connections, so it is easy to compute how the error will be affected by changing these states and weights.

ただし、この合計入力は下位レベルのユニットの状態の線形関数であり、接続の重みの線形関数でもあるため、これらの状態と重みを変更することで誤差がどのように影響を受けるかを簡単に計算できます。

For a weight from i to j the derivative is

iからjまでの重みw\_jiの導関数は

and for the output of the unit the contribution to resulting from the effect of i on j is simply

そして、i番目のユニットの出力の場合、jに対するiの効果から生じるへの寄与は単純に

so taking into account all the connections emanating from unit i we have

ユニットiから発生するすべての接続を考慮に入れます

We have now seen how to compute for any unit in the penultimate layer when given for all units in the last layer.

最後の層のすべてのユニットに対してが与えられた場合、最後から2番目の層のユニットに対してを計算する方法を見てきました。

We can therefore repeat this procedure to compute this term for successively earlier layers, computing aE / aw for the weights as we go.

したがって、この手順を繰り返して、連続して以前のレイヤーのこの項を計算し、進行中の重みのを計算します。

One way of using aE / aw is to change the weights after every input-output case.

を使用する1つの方法は、すべての入出力ケースの後に重みを変更することです。

This has the advantage that no separate memory is required for the derivatives.

これには、派生物に別個のメモリが必要ないという利点があります。

An alternative scheme, which we used in the research reported here, is to accumulate aE/aw over all the input-output cases before changing the weights.

ここで報告された研究で使用した別のスキームは、重みを変更する前に、すべての入出力ケースにわたってを蓄積することです。

The simplest version of gradient descent is to change each weight by an amount proportional to the accumulated aE/aw

勾配降下の最も単純なバージョンは、蓄積されたaE / awに比例する量で各重みを変更することです。

This method does not converge as rapidly as methods which make use of the second derivatives, but it is much simpler and can easily be implemented by local computations in parallel hardware.

この方法は、2次導関数を使用する方法ほど速く収束しませんが、はるかに単純であり、並列ハードウェアでのローカル計算によって簡単に実装できます。

It can be significantly improved, without sacrificing the simplicity and locality, by using an acceleration method in which the current gradient is used to modify the velocity of the point in weight space instead of its position

現在の勾配を使用して、位置の代わりにウェイトスペースのポイントの速度を変更する加速方法を使用することにより、単純さと局所性を犠牲にすることなく、大幅に改善できます。

where t is incremented by 1 for each sweep through the whole set of input-output cases, and a is an exponential decay factor between O and 1 that determines the relative contribution of the current gradient and earlier gradients to the weight change.

ここで、tは入出力ケースのセット全体をスイープするたびに1ずつ増加します。aはOと1の間の指数関数的な減衰係数で、現在の勾配と以前の勾配の重み変化への相対的な寄与を決定します。

To break symmetry we start with small random weights.

対称性を破るには、小さなランダムな重みから始めます。

Variants on the learning procedure have been discovered independently by David Parker (personal communication) and by Yann Le Cun.

学習手順のバリエーションは、David Parker（パーソナルコミュニケーション）とYann Le Cunによって独自に発見されました。

One simple task that cannot be done by just connecting the input units to the output units is the detection of symmetry.

入力ユニットを出力ユニットに接続するだけではできない単純なタスクの1つは、対称性の検出です。

To detect whether the binary activity levels of a one-dimensional array of input units are symmetrical about the centre point, it is essential to use an intermediate layer because the activity in an individual input unit, considered alone, provides no evidence about the symmetry or non-symmetry of the whole input vector, so simply adding up the evidence from the individual input units is insufficient.

入力ユニットの1次元配列のバイナリアクティビティレベルが中心点に関して対称であるかどうかを検出するには、単独で考慮される個々の入力ユニットのアクティビティは対称性に関する証拠を提供しないため、中間層を使用することが不可欠です入力ベクトル全体が非対称であるため、個々の入力ユニットからの証拠を単純に足すだけでは不十分です。

(A more formal proof that intermediate units are required is given in ref. 2.) The learning procedure discovered an elegant solution using just two intermediate units, as shown in Fig. 1.

（中間ユニットが必要であるというより正式な証拠は、参考文献2に記載されています。）学習手順は、図1に示すように、たった2つの中間ユニットを使用するエレガントなソリューションを発見しました。

Another interesting task is to store the information in the two family trees (Fig. 2).

もう1つの興味深いタスクは、2つの家系図に情報を保存することです（図2）。

Figure 3 shows the network we used, and Fig. 4 shows the 'receptive fields' of some of the hidden units after the network was trained on 100 of the 104 possible triples.

図3は、使用したネットワークを示しています。図4は、ネットワークが104の可能性のあるトリプルのうち100でトレーニングされた後の隠れユニットの「受容フィールド」を示しています。

So far, we have only dealt with layered, feed-forward networks.

ここまでは、階層化されたフィードフォワードネットワークのみを扱ってきました。

The equivalence between layered networks and recurrent networks that are run iteratively is shown in Fig. 5.

層状ネットワークと反復的に実行されるリカレントネットワークの等価性を図5に示します。

The most obvious drawback of the learning procedure is that the error-surface may contain local minima so that gradient descent is not guaranteed to find a global minimum.

学習手順の最も明らかな欠点は、エラーサーフェスに局所的な最小値が含まれるため、勾配降下がグローバルな最小値を見つけることが保証されないことです。

However, experience with many tasks shows that the network very rarely gets stuck in poor local minima that are significantly worse than the global minimum.

ただし、多くのタスクの経験から、ネットワークがグローバルな最小値よりも著しく悪い貧弱なローカル最小値でスタックすることはほとんどありません。

We have only encountered this undesirable behaviour in networks that have just enough connections to perform the task.

この望ましくない動作は、タスクを実行するのに十分な接続しかないネットワークでのみ発生しています。

Adding a few more connections creates extra dimensions in weight-space and these dimensions provide paths around the barriers that create poor local minima in the lower dimensional subspaces.

いくつかの接続を追加すると、ウェイトスペースに余分な次元が作成され、これらの次元はバリアの周りのパスを提供し、低次元のサブスペースに局所的な最小値を作成します。

The learning procedure, in its current form, is not a plausible model of learning in brains.

現在の形態の学習手順は、脳での学習のもっともらしいモデルではありません。

However, applying the procedure to various tasks shows that interesting internal representations can be constructed by gradient descent in weight-space, and this suggests that it is worth looking for more biologically plausible ways of doing gradient descent in neural networks.

ただし、この手順をさまざまなタスクに適用すると、重み空間での勾配降下によって興味深い内部表現を構築できることがわかります。これは、ニューラルネットワークで勾配降下を行うより生物学的に妥当な方法を探す価値があることを示唆しています。

We thank the System Development Foundation and the Office of Naval Research for financial support.

システム開発財団と海軍研究局に経済的支援を感謝します。