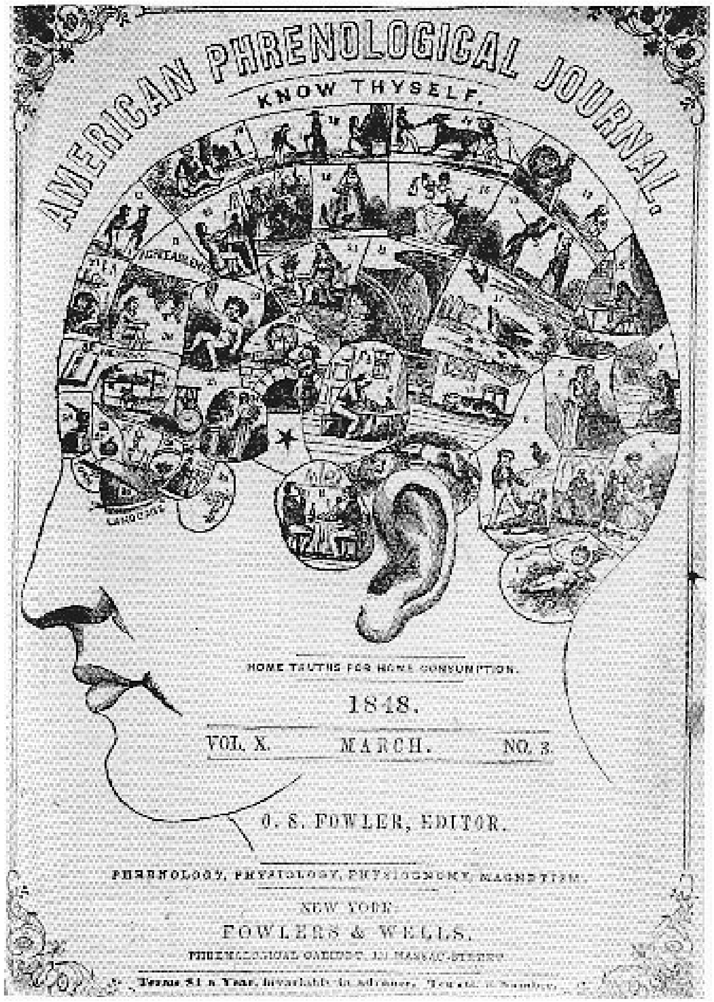
How the brain doesn't work

THE AUSTRALIAN NATIONAL UNIVERSITY



Biology and Learning

THE AUSTRALIAN NATIONAL UNIVERSITY

Basic Idea

- Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves the fitness of the system.

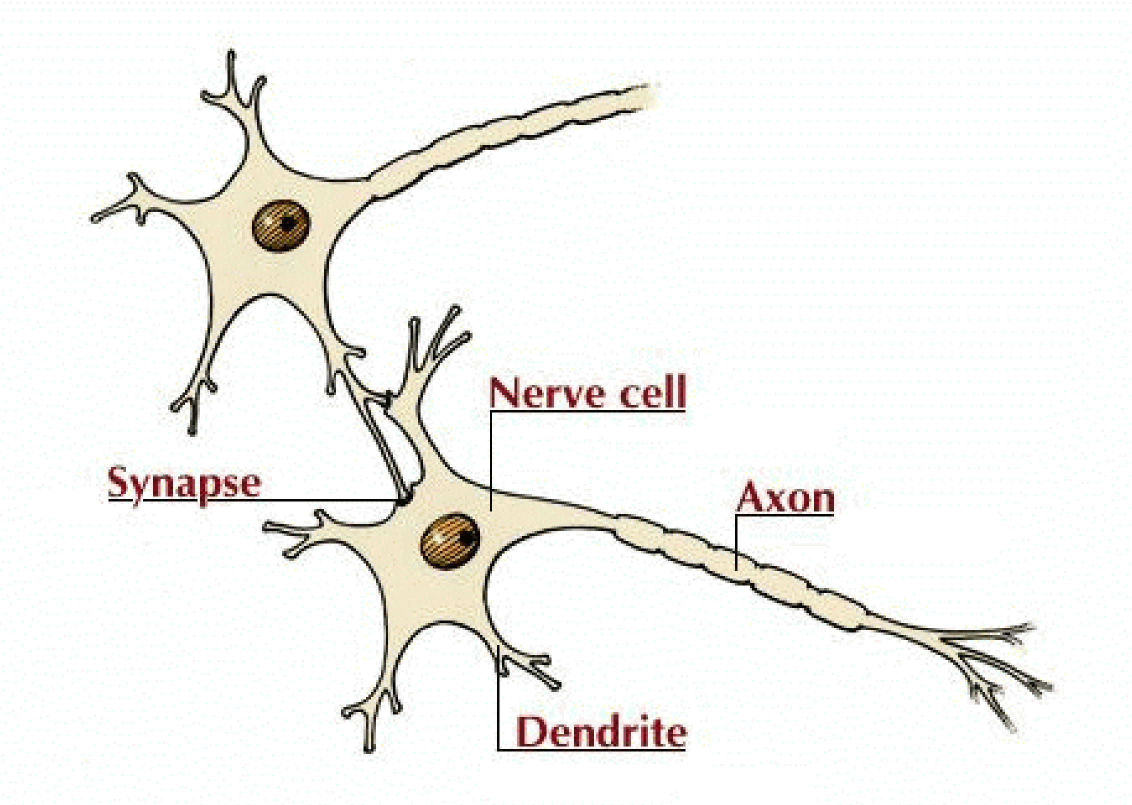
  Example: hitting a sabertooth tiger over the head should be rewarded ...
- Correlated events should be combined.

  Example: Pavlov's salivating dog.

Training Mechanisms

- Behavioral modification of individuals (learning) — successful behavior is rewarded (e.g. food).
- Hard-coded behavior in the genes (instinct) — the wrongly coded animal dies.

Neurons



Soma Cell body. Here the signals are combined ("CPU").

Dendrite Combines the inputs from several other nerve cells ("input bus").

Synapse Interface between two neurons ("connector").

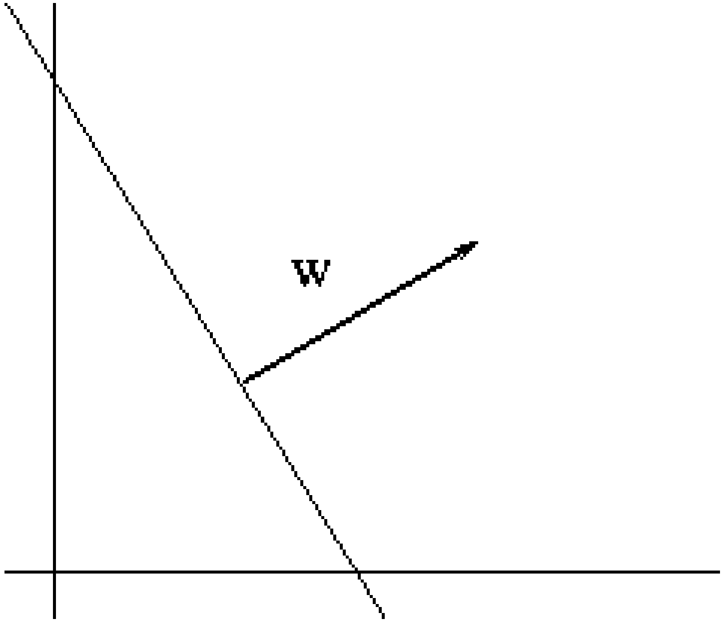
Axon This may be up to 1m long and will transport the activation signal to nerve cells at different locations ("output cable").

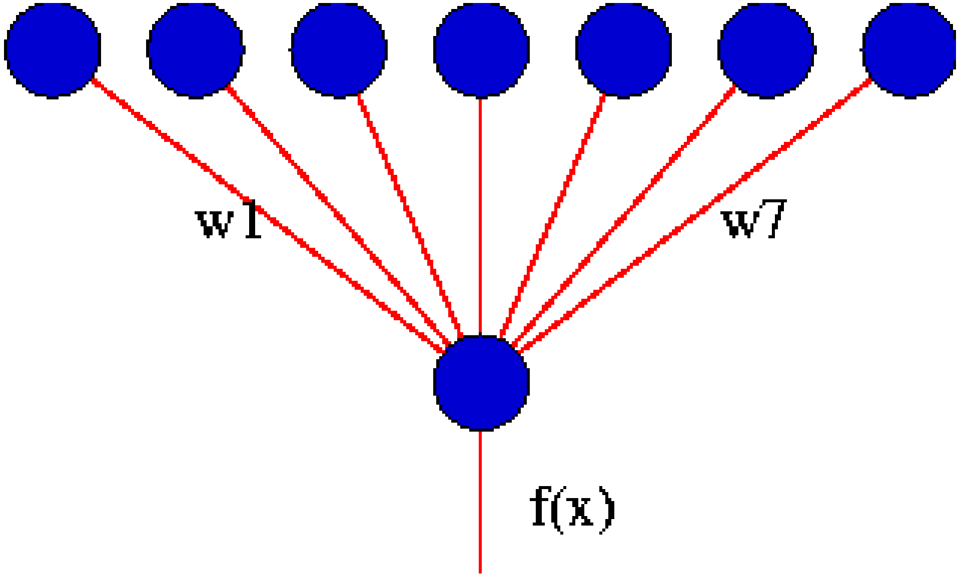
Perceptrons

Linear Separation

The output of the neuron is a linear combination of the inputs (from the other neurons via their axons) rescaled by the synaptic weights.

Often the output does not directly correspond to the activation level but is a monotonic function thereof.





Separating Half Spaces

THE AUSTRALIAN NATIONAL UNIVERSITY

Linear Functions

An abstract model is to assume that $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ where $\mathbf{x}, \mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$.

Biological Interpretation

The weights w_i correspond to the synaptic weights (activating or inhibiting), the multiplication corresponds to the processing of inputs via the synapses, and the summation is the combination of signals in the cell body (soma).

Applications

Spam filtering (e-mail), echo cancellation (old analog overseas cables)

Learning

The weights are "plastic" can be adapted via the training data.

Perceptron Learning Rule

argument: `Training sample,` $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{X}$, $\{y_1, \ldots, y_m\} \subset \{\pm 1\}$

`Learning rate,` η

returns: `Weight vector` $\mathbf{w}$ `and threshold` b.

function `Perceptron`(X, Y, η)

  initialize $\mathbf{w}, b = 0$

  repeat

    for all i `from` $i = 1, \ldots, m$

      `Compute` $g(\mathbf{x}_i) = \mathrm{sgn}((\mathbf{w} \cdot \mathbf{x}_i) + b)$

      `Update` $\mathbf{w}, b$ `according to`

$$\begin{aligned} \mathbf{w}' &= \mathbf{w} + (\eta/2)\,(y_i - g(\mathbf{x}_i))\,\mathbf{x}_i \\ b' &= b + (\eta/2)\,(y_i - g(\mathbf{x}_i))\,. \end{aligned}$$

    endfor

  until `for all` $1 \leq i \leq m$ `we have` $g(\mathbf{x}_i) = y_i$

  return $f : \mathbf{x} \mapsto (\mathbf{w} \cdot \mathbf{x}) + b$

end

Theoretical Analysis

THE AUSTRALIAN NATIONAL UNIVERSITY

Incremental Algorithm

Already while the perceptron is learning, we can use it. Updates are only small steps.

Convergence Theorem

Suppose that there exists a $\rho > 0$, a weight vector $\mathbf{w}^*$ satisfying $\|\mathbf{w}^*\| = 1$, and a threshold b^* such that

$$y_i \left(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*\right) \geq \rho \text{ for all } 1 \leq i \leq m.$$

Then for all $\eta > 0$, the hypothesis maintained by the perceptron algorithm converges after no more than $(b^{*2}+1)(R^2+1)/\rho^2$ updates, where $R = \max_i \|x_i\|$. Clearly, the limiting hypothesis is consistent with the training data (X, Y).

This theorem is due to Rosenblatt and Novikoff.

Proof, Part I

THE AUSTRALIAN NATIONAL UNIVERSITY

Starting Point

We start from $\mathbf{w}_1 = 0$ and $b_1 = 0$.

Bound on the Increase of $\langle(\mathbf{w}_i, b_i), (\mathbf{w}^*, b^*)\rangle$

Denote by $\mathbf{w}_i$ the value of $\mathbf{w}$ at step i, and analogously b_i. Then we have

$$\begin{aligned}
\langle(\mathbf{w}_{j+1}, b_{j+1}) \cdot (\mathbf{w}^*, b^*)\rangle &= \langle[(\mathbf{w}_j, b_j) + (\eta/2)(y_i - g_j(\mathbf{x}_i))(\mathbf{x}_i, 1)], (\mathbf{w}^*, b^*)\rangle \\
&= \langle(\mathbf{w}_j, b_j), (\mathbf{w}^*, b^*)\rangle + \eta y_i \langle(\mathbf{x}_i, 1) \cdot (\mathbf{w}^*, b^*)\rangle \\
&\geq \langle(\mathbf{w}_j, b_j), (\mathbf{w}^*, b^*)\rangle + \eta\rho \\
&\geq j\eta\rho.
\end{aligned}$$

Cauchy-Schwartz for the Dot Product

$$\begin{aligned}
\langle(\mathbf{w}_{j+1}, b_{j+1}) \cdot (\mathbf{w}^*, b^*)\rangle &\leq \|(\mathbf{w}_{j+1}, b_{j+1})\| \, \|(\mathbf{w}^*, b^*)\| \\
&= \sqrt{1 + (b^*)^2} \|(\mathbf{w}_{j+1}, b_{j+1})\|
\end{aligned}$$

Proof, Part II

THE AUSTRALIAN NATIONAL UNIVERSITY

Combination of First Two Steps

$$j\eta\rho \leq \sqrt{1+(b^*)^2}\|(\mathbf{w}_{j+1}, b_{j+1})\|$$

Upper Bound on $\|wb_j, b_j\|$

If we make a mistake we have

$$\begin{aligned}
\|(\mathbf{w}_{j+1}, b_{j+1})\|^2 &= \|(\mathbf{w}_j, b_j) + \eta y_i(\mathbf{x}_i, 1)\|^2 \\
&= \|(\mathbf{w}_j, b_j)\|^2 + 2\eta y_i\langle(\mathbf{x}_i, 1), (\mathbf{w}_j, b_j)\rangle + \eta^2\|(\mathbf{x}_i, 1)\|^2 \\
&\leq \|(\mathbf{w}_j, b_j)\|^2 + \eta^2\|(\mathbf{x}_i, 1)\|^2 \\
&\leq j\eta^2(R^2+1).
\end{aligned}$$

Combination with Third Step

$$jn\rho \leq \sqrt{1+(b^*)^2}\|(\mathbf{w}_{j+1}, b_{j+1})\| \leq \sqrt{1+(b^*)^2}\sqrt{j\eta^2(R^2+1)}$$

Solving for j proves the theorem.

What does it mean?

THE AUSTRALIAN NATIONAL UNIVERSITY

Learning Algorithm

We perform an update only if we make a mistake.

Convergence Bound

This bounds the maximum number of mistakes **in total**. If we do not stop learning we will make at most $(b^{*2} + 1)(R^1 + 1)/\rho^2$ mistakes in the case where a "correct" solution $\mathbf{w}^*, b^*$ exists.

This also bounds the expected error (if we know ρ, R, and $|b^*|$).

Dimension Independent

Note that this bound does not depend at all on the dimensionality of $\mathcal{X}$. Also the learning algorithm itself only depends on $\mathcal{X}$ via the observations $\mathbf{x}_i$.

Sample Expansion

We obtain $\mathbf{x}$ as a **linear combination** of $\mathbf{x}_i$.

Realizable and Non-realizable Concepts

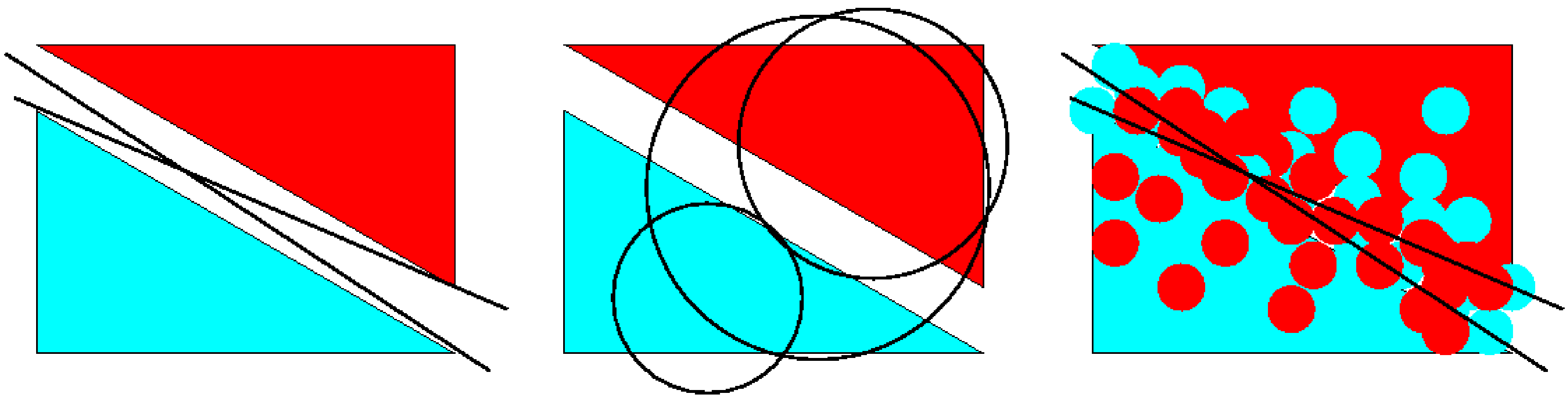
THE AUSTRALIAN NATIONAL UNIVERSITY

Realizable Concept

Here some $\mathbf{w}^*, b^*$ exists such that y is generated by $y = \operatorname{sgn}(\langle \mathbf{w}^*, \mathbf{x} \rangle + b)$. In general realizable means that the exact functional dependency is included in the class of admissible hypotheses.

Unrealizable Concept

In this case, the exact concept does not exist or it is not included in the function class.



Perceptron as Stochastic Gradient Descent

THE AUSTRALIAN NATIONAL UNIVERSITY

Linear Function Expansion $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$

Objective Function

$$R[f] := \sum_{i=1}^{m} \max(0, -y_i f(\mathbf{x}_i)) = \sum_{i=1}^{m} \max\left(0, -y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right)$$

Stochastic Gradient

We use each term in the sum as a stochastic approximation of the overall objective function. This leads to the stochastic gradient

$$\partial_{\mathbf{w}} \max\left(0, -y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right) = \begin{cases} -y_i \mathbf{x}_i & \text{for } f(\mathbf{x}_i) < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\partial_{b} \max\left(0, -y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right) = \begin{cases} -y_i & \text{for } f(\mathbf{x}_i) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Update Equation

$\mathbf{w} \rightarrow \mathbf{w} - \lambda \partial_{\mathbf{w}}[\ldots], b \rightarrow b - \lambda \partial_b[\ldots]$

Nonlinearity via Preprocessing

THE AUSTRALIAN NATIONAL UNIVERSITY

Problem

Linear functions are often too simple to provide good estimators.

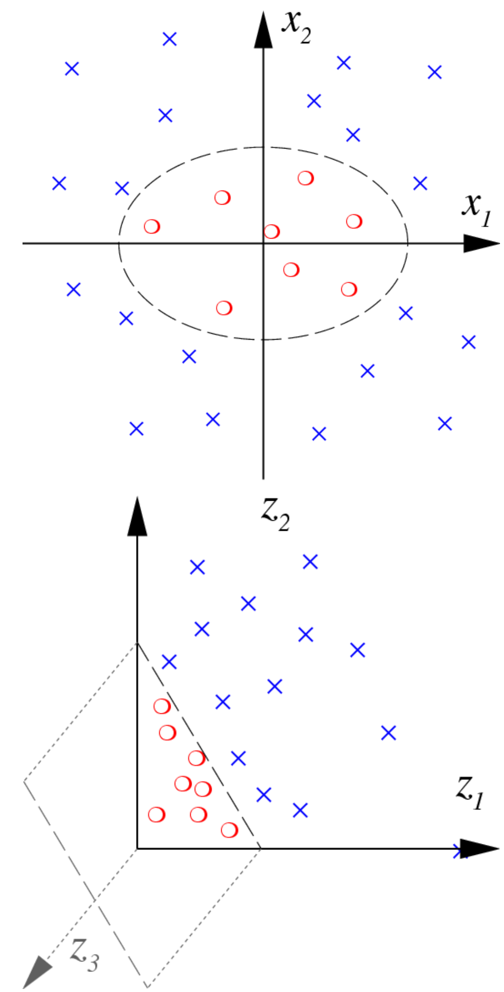
Idea

Map to a higher dimensional feature space via $\Phi : x \longrightarrow \Phi(x)$ and solve the problem there.

Replace every $\langle \mathbf{x}, \mathbf{x}' \rangle$ by $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ in the perceptron algorithm.

Consequence

We have nonlinear classifiers.



Perceptron on Features

THE AUSTRALIAN NATIONAL UNIVERSITY

argument: `Training sample,` $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{X}$, $\{y_1, \ldots, y_m\} \subset \{\pm 1\}$, η

returns: `Weight vector` $\mathbf{w}$ `and threshold` b.

function `Perceptron`(X, Y, η)

initialize $\mathbf{w}, b = 0$

repeat

for all i `from` $i = 1, \ldots, m$

`Compute` $g(\mathbf{x}_i) = \mathrm{sgn}\left(\left\langle \sum_{l=1}^{i} \alpha_l \Phi(x_l), \Phi(\mathbf{x}_i) \right\rangle + b\right)$

`Update` $\mathbf{w}, b$ `according to`

$$\mathbf{w}' = \mathbf{w} + (\eta/2)\alpha_i \Phi(\mathbf{x}_i) \text{ where } \alpha_i \left(y_i - g(\mathbf{x}_i)\right)$$

$$b' = b + (\eta/2)\left(y_i - g(\mathbf{x}_i)\right).$$

endfor

until `for all` $1 \leq i \leq m$ `we have` $g(\mathbf{x}_i) = y_i$

return $f : \mathbf{x} \mapsto \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$

end

An Observation: Polynomial Features

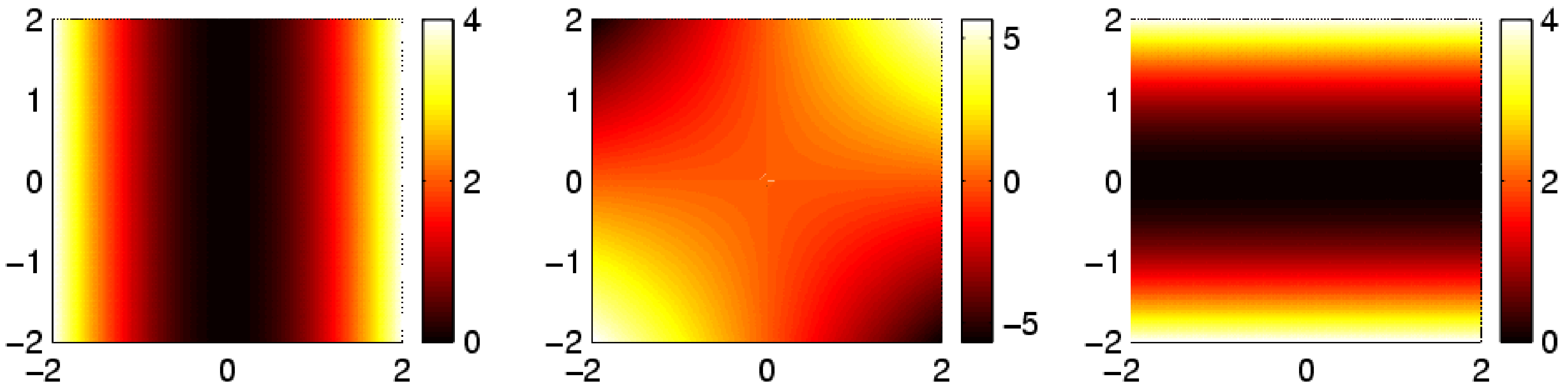
Quadratic Features in $\mathbb{R}^2$

$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

Dot Product

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x'_1}^2, \sqrt{2}x'_1x'_2, {x'_2}^2\right) \right\rangle = \langle x, x' \rangle^2.$$

This trick does not only work for 2nd order polynomials but for any $\langle x, x' \rangle^d$.



Perceptron with Kernels

Idea

The dot product in feature space can sometimes be computed more cheaply than actually computing the feature map $\Phi : \mathbf{x} \to \Phi(\mathbf{x})$. We define

$$k(\mathbf{x}, \mathbf{x}') := \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

to be the **kernel** function between $\mathbf{x}$ and $\mathbf{x}'$.

Consequence

Replace $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ by $k(\mathbf{x}, \mathbf{x}')$ to obtain a nonlinear algorithm from a linear algorithm.

Problem

Will any $k(\mathbf{x}, \mathbf{x}')$ do? No, and the details will be revealed in two weeks ...