Adaline (Adaptive Linear)

Let $\mathbf{x}$ be the inputs and $\mathbf{w}$ be the weights. For mathematical convenience, let $x_{n+1} = 1$, so that w_{n+1} becomes the bias weight. The weighted sum u is a dot product:

$$u = \mathbf{w} \cdot \mathbf{x} = \sum_i w_i \, x_i$$

Let the identity function $o = u$ be the activation function. Let squared error $E = (d - o)^2$ be the error function.

The decision boundary is the hyperplane $o = 0$.

Adaline Learning Rule

Let η be the learning rate, and d the desired (real number) output. The learning rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(d - o)\mathbf{x}$$

Thie learning rule is also called the LMS (least mean square) algorithm and the Widrow-Hoff learning rule.

Approximately, the adaline converges to least squares (L_2) error.

Justifying the Learning Rule

The adaline learning rule is justified by gradient descent:

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{w}} &= \frac{\partial (o-d)^2}{\partial \mathbf{w}} = 2(o-d)\frac{\partial o}{\partial \mathbf{w}} \\
&= 2(o-d)\frac{\partial u}{\partial \mathbf{w}} = 2(o-d)\frac{\partial \mathbf{w}\cdot\mathbf{x}}{\partial \mathbf{w}} \\
&= 2(o-d)\mathbf{x}
\end{aligned}
$$

Moving in direction $(o-d)\mathbf{x}$ increases error, so the opposite direction $(d-o)\mathbf{x}$ decreases error.

Adaline Example ($\eta = 0.1$)

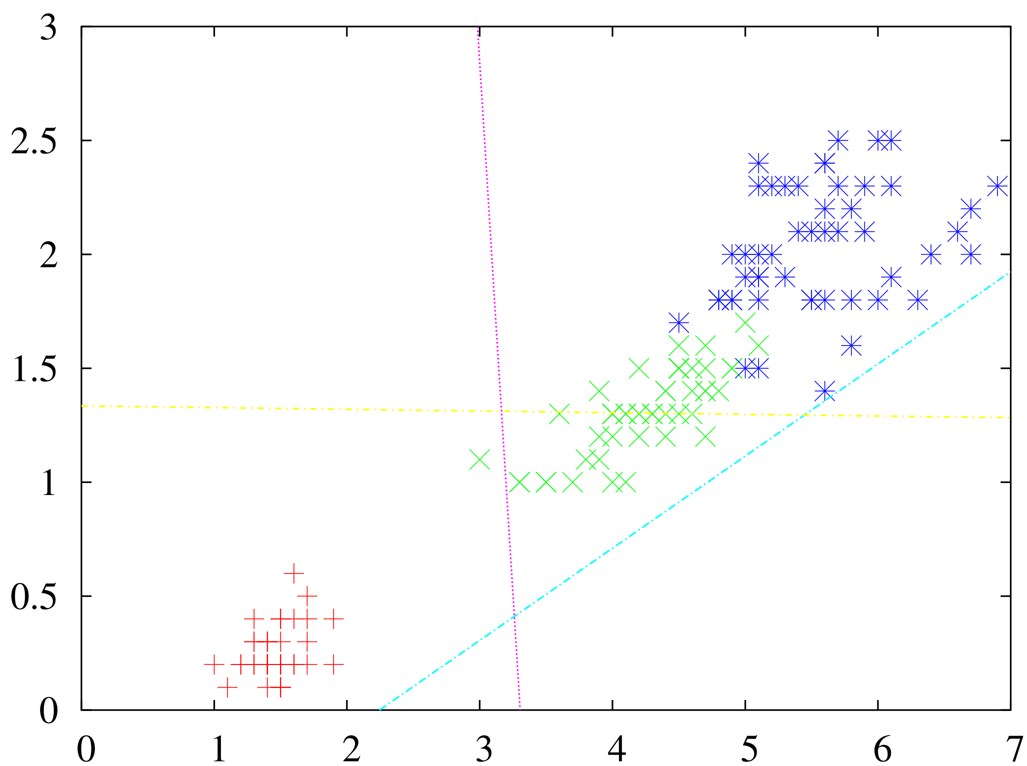
x_1	x_2	x_3	d	u	w_1	w_2	w_3	$w_4 = b$
					0.00	0.00	0.00	0.00
1	1	1	1	0.00	0.10	0.10	0.10	0.10
1	1	−1	−1	0.20	−0.02	−0.02	0.22	−0.02
1	−1	1	1	0.20	0.06	−0.10	0.30	0.06
1	−1	−1	−1	−0.08	−0.03	−0.01	0.39	−0.03
−1	1	1	1	0.38	−0.09	0.05	0.45	0.03
−1	1	−1	−1	−0.28	−0.02	−0.02	0.53	−0.04
−1	−1	1	1	0.52	−0.07	−0.07	0.57	0.00
−1	−1	−1	1	−0.43	−0.21	−0.21	0.43	0.15

In 4 more epochs, the adaline converges, but not to optimal.

Epoch	w_1	w_2	w_3	$w_4 = b$	error	0-1
1	-0.21	-0.21	0.43	0.15	2.92	0
2	-0.28	-0.29	0.58	0.21	2.27	0
3	-0.31	-0.33	0.63	0.24	2.19	0
4	-0.31	-0.34	0.64	0.24	2.19	0
∞	-0.32	-0.35	0.65	0.25	2.19	0
Optimal	-0.25	-0.25	0.75	0.25	2.00	0

To be closer to optimal, use a lower learning rate and more epochs.

1000 epochs of adaline on iris data ($\eta = 0.001$)



Algorithms for Learning Adalines

Method 1 (p. 226): Method of least squares.

Method 2 (p. 229): Generalization of method 1.

Method 3 (p. 230): Batch version of adaline, i.e., $\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_i (d_i - o_i)\mathbf{x}_i$

Method 4 (p. 234): Adaline learning rule.

Method 5 (p. 237): Recursive least squares updates an exact solution for p patterns given the $p+1$st pattern.

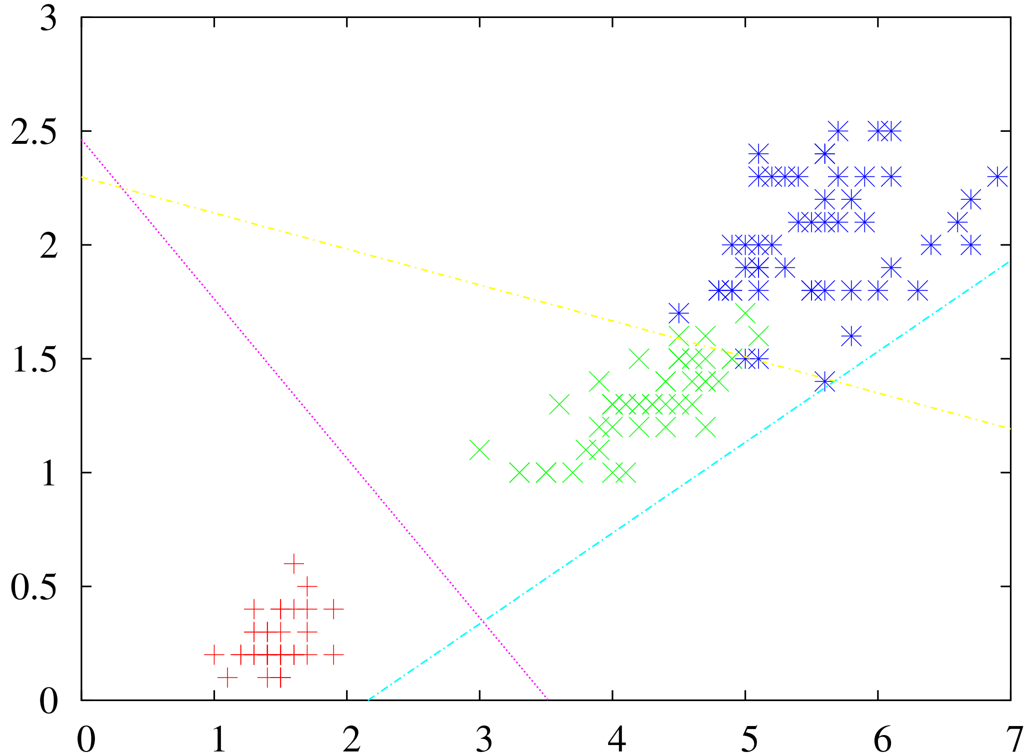
Using Adaline for Classification

Given a fixed set of patterns with a single real output, use method of least squares.

For classification, use ramp activation and $E(d, u) = 0$ if $\text{ramp}(u) = d$ else $(u - d)^2$.

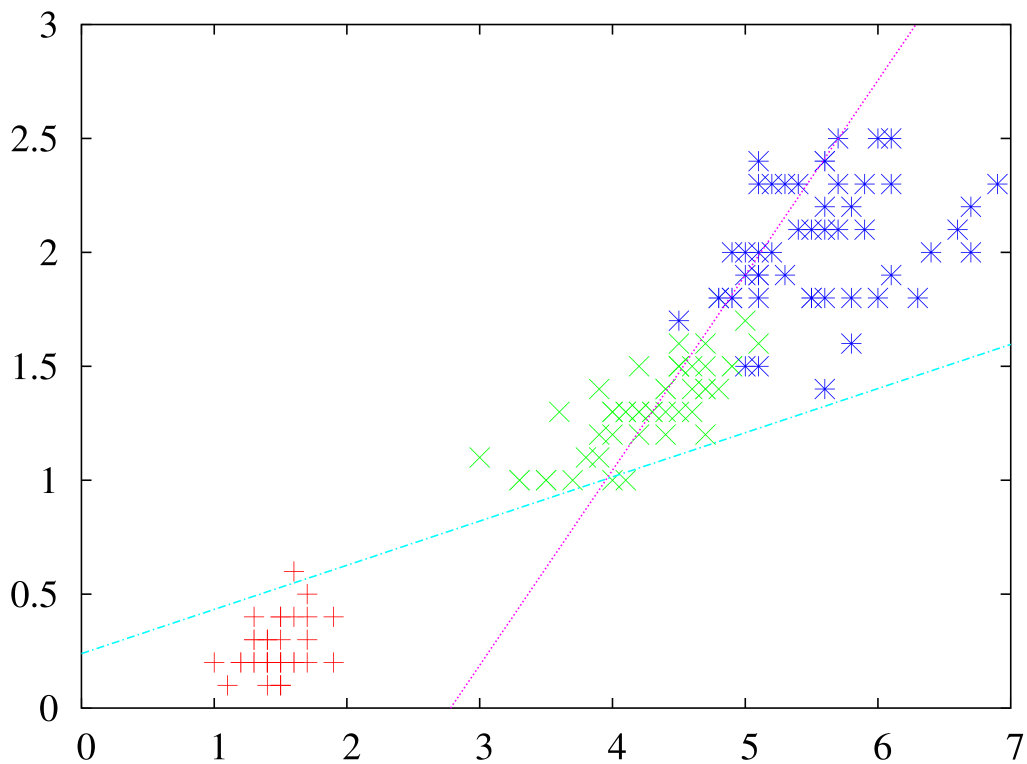
For multiclass datasets, use a linear machine. Predicted class is the neuron with highest u. Let $u_d = u$ of the neuron for the desired class. Let $u' =$ highest u of other neurons. Subtract $(u_d + u')/2$ from u of all neurons. Apply learning rule to all neurons.

adaline with ramp activation on iris data



---

adaline with linear machine on iris data



adaline with ramp and linear machine on iris