Aturan Pembelajaran Perceptron

Tujuan

Salah satu pertanyaan kita yang muncul adalah: "Bagaimana kita menentukan Matrik bobot dan bias untuk jaringan perceptron dengan banyak input, dimana adalah mustahil untuk memvisualisasikan batasan-batasan keputusan?" Dalam bab ini kita akan menggambarkan suatu algoritma untuk *pelatihan* jaringan perceptron, sehingga mereka dapat *belajar* untuk memecahkan masalah klasifikasi. Kita akan mulai dengan menjelaskan apa yang dimaksud dengan aturan belajar (*learning rule*) dan akan belajar mengembangkan aturan perceptron. Kami akan menyimpulkan dengan mendiskusikan keuntungan dan keterbatasan dari jaringan single - layer perceptron. Diskusi ini akan membawa kita ke dalam bab-bab selanjutnya.

Teori dan Contoh

Pada tahun 1943, Warren McCulloch dan Walter Pitts memperkenalkan salah satu artificial neurons [McPi43]. Fitur utama dari model neuron mereka adalah bahwa jumlah bobot sinyal input dibandingkan dengan ambang batas untuk menentukan neuron output. Ketika jumlah lebih besar dari atau sama dengan ambang batas, output adalah 1. Ketika jumlah kurang dari ambang batas, keluaran adalah 0. Mereka tetap meneruskan penelitian dengan menunjukkan bahwa jaringan neuron ini pada prinsipnya bisa menghitung setiap fungsi aritmetika atau logika. Tidak seperti jaringan biologis, parameters jaringan mereka harus dirancang, karena tidak ada metode pelatihan yang tersedia. Namun, hubungan yang dirasakan antara biologi dan komputer digital menghasilkan banyak minat

Pada akhir 1950-an, Frank Rosenblatt dan beberapa peneliti lain mengembangkan suatu kelas jaringan saraf yang disebut Perceptrons. Neuron dalam jaringan yang mirip dengan McCulloch dan pitts. Kunci kontribusi Rosenblatt adalah pengenalan aturan belajar untuk pelatihan jaringan perceptron untuk memecahkan masalah pengenalan pola [Rose58]. Ia membuktikan bahwa aturan belajarnya akan selalu bertemu untuk bobot jaringan yang benar, jika bobot yang ada memecahkan masalah. Pembelajarannya sederhana dan otomatis. Contoh perilaku yang layak

diajukan ke jaringan yang belajar dari kesalahan. Perceptron bahkan bisa belajar ketika diinisialisasi dengan nilai acak untuk bobot dan bias.

Sayangnya, jaringan perceptron secara inheren terbatas. Keterbatasan ini dipublikasikan secara luas dalam buku *Perceptrons* [MiPa69] oleh Marvin Minsky dan Seymour Papert. Mereka menunjukkan bahwa jaringan perceptron tidak mampu melaksanakan fungsi dasar tertentu. Hal ini tidak sampai tahun 1980-an dimana keterbatasan ini diatasi dengan memperbaiki jaringan perceptron (multilayer) dan aturan belajar yang saling terkait/berhubungan.

Saat ini perceptron masih dipandang sebagai jaringan penting. Ia menyisakan suatu jaringan yang cepat dan handal untuk kelas masalah yang dapat dipecahkan. Selain daripada itu, pemahaman tentang operasi dari perceptron menyediakan dasar yang baik untuk memahami jaringan yang lebih kompleks. Jadi jaringan perceptron, dan aturan belajar yang terkait, adalah baik dan layak untuk didiskusikan di sini.

Pada sisa dari bab ini kita akan mendefinisikan apa yang kita maksud dengan aturan belajar, menjelaskan jaringan perceptron dan aturan belajar, dan mendiskusikan keterbatasan jaringan perceptron.

Aturan Pembelajaran

Pembelajaran berarti belajar dari kesalahan. Semakin sering jaringan syaraf tiruan ini digunakan semakin cerdaslah dia, artinya kesalahannya semakin kecil. Kecepatan peningkatan kecerdasan ditentukan oleh nilai parameter kecepatan pembelajaran (learning rate) disimbolkan dengan α. Bagaimanapun, jika sampai kali ke-N komputasi ulang (istilahnya: N iterasi) kesalahannya tetap besar maka proses pembelajaran dihentikan dan jaringan digunakan apa adanya.
Ketika kita mulai pembahasan kita tentang aturan pembelajaran perceptron, kami ingin mendiskusikan aturan pembelajaran secara umum. *Aturan belajar* yang kami maksud adalah sebuah prosedur untuk memodifikasi bobot dan bias dari jaringan. (Prosedur ini juga dapat disebut sebagai algoritma pelatihan). Tujuan aturan pembelajaran untuk melatih jaringan untuk

melakukan beberapa tugas. Ada banyak jenis aturan belajar jaringan saraf. Mereka dibagi dalam 3 kategori besar : pembelajaran dengan pengawasan (*supervised learning*), pembelajaran tanpa pengawasan (*unsupervised learning*) dan pembelajaran penguatan tanpa pengawasan (atau dinilai) (*reinforcement learning*).

Ada 3 mekanisme penting proses pembelajaran:

1. Hitung selisih keluaran (a) dari target (t)
2. Jika besarnya selisih dapat ditolerir maka a diterima, tetapi jika selisih tersebut tidak dapat ditolerir dan banyaknya iterasi belum N kali maka rubahlah w dan b, lalu lakukan komputasi ulang.
3. Nilai baru w dan b bergantung kepada nilai α .

Segera setelah jaringan mengalami pembelajaran, jaringan tersebut dapat digunakan untuk menyelesaikan masalah yang sesuai dengan konstruksi jaringan tersebut.

Supervised Learning

Pada pembelajaran dengan pengawasan, aturan pembelajaran dilengkapi dengan serangkaian contoh (himpunan pelatihan) dari perilaku jaringan yang tepat :

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2,\mathbf{t}_2\}, \ldots, \{\mathbf{p}_q,\mathbf{t}_q\}, \quad (4.1)$$

Dimana p_q adalah sebuah input ke jaringan dan yang berhubungan dengan (target) output yang sesuai. Seperti input yang diterapkan ke jaringan, output jaringan dibandingkan dengan target. Aturan pembelajaran kemudian digunakan untuk mengatur bobot dan bias dari jaringan untuk memindahkan output jaringan mendekati target. Aturan pembelajaran perceptron masuk dalam kategori pembelajaran pengawasan ini.

Reinforcement Learning

Pembelajaran dengan penguatan adalah serupa dengan pembelajaran dengan pengawasan, kecuali bahwa, pada supervised learning dilengkapi dengan output yang benar untuk setiap input jaringan, sementara algoritma Reinforcement Learning hanya diberikan satu nilai. Nilai adalah sebuah ukuran dari kinerja jaringan atas beberapa rangkaian input. Saat ini, jenis pembelajaran

ini, jauh lebih sedikit dibandingkan pembelajaran dengan pengawasan. Ia tampaknya menjadi paling cocok untuk aplikasi-aplikasi sistem kontrol (lihat [BaSu83], [WhSo92]).

Pembelajaran Tanpa Pengawasan

Pada pembelajaran tanpa pengawasan, bobot dan bias diubah sebagai tanggapan untuk jaringan input saja. Tidak ada target output yang tersedia. Awalnya sekilas ini mungkin kelihatannya tidak praktis. Bagaimana anda bisa melatih jaringan jika anda tidak tahu apa yang seharusnya dilakukan? Kebanyakan dari algoritma ini melakukan semacam operasi *clustering* (pengelompokan). Mereka belajar untuk mengkategorikan pola input dalam sejumlah kelas yang terbatas. Hal ini sangat berguna khususnya dalam aplikasi seperti vector kuantisasi.

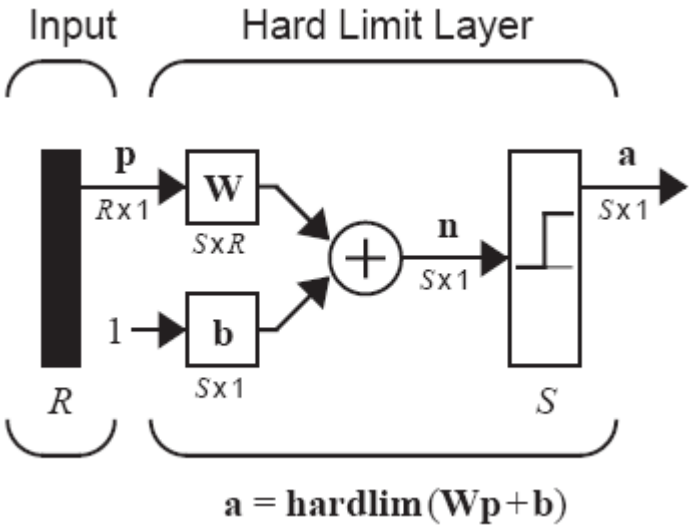
Arsitektur Perceptron

Sebelum kami menyajikan aturan pembelajaran perceptron, marilah kita memperluas pemahaman kita akan jaringan perceptron. Jaringan perceptron yang umum ditunjukkan pada Gambar 4.1.

Output dari jaringan diberikan oleh

$$\mathbf{a = hardlim(Wp + b).} \qquad (4.2)$$

(Perhatikan bahwa dalam bab 3 kita menggunakan fungsi transfer, bukan hardlim. Hal ini tidak mempengaruhi kemampuan jaringan).



Gambar 4.1. Jaringan Perceptron

Ini akan berguna dalam pengembangan aturan jaringan perceptron kita menjadi dapat dengan mudah mereferensikan elemen-elemen individu dari output jaringan. Marilah kita lihat bagaimana hal ini dapat dilakukan. Pertama, pertimbangkan matrik bobot jaringan berikut:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \cdots & \cdots & \cdots & \cdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{4.3}$$

Kita akan mendefinisikan sebuah vector yang terdiri dari elemen-elemen W dari baris ke-i

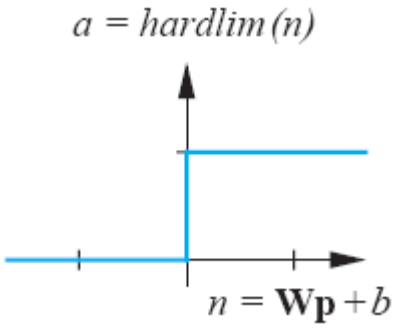
$$_i w = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \tag{4.4}$$

Sekarang kita dapat merpartisi matrik bobot:

$$W = \begin{bmatrix} _1 w^T \\ _2 w^T \\ \vdots \\ _S w^T \end{bmatrix} \tag{4.5}$$

Hal ini memungkinkan kita untuk menulis elemen ke-i dari vektor output jaringan sebagai

$$a_i = hard\lim(n_i) = hard\lim({}_i w^T p + b_i) \qquad (4.6)$$



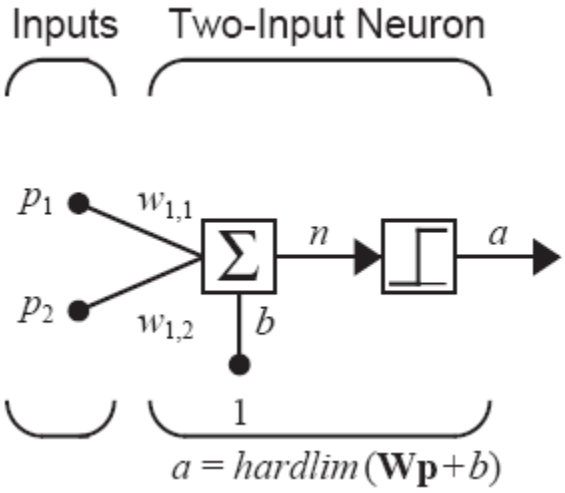
Ingatlah bahwa fungsi transfer *hardlim* (ditampilkan di sebelah kiri) didefinisikan sebagai

$$a = hard\lim(n) = \begin{cases} 1\ if\ n \geq 0 \\ 0\ otherwise \end{cases} \qquad (4.7)$$

Oleh karena itu, jika produk inner dari baris ke-*i* dari matrik bobot dengan vektor input lebih besar dari atau sama dengan $-b_i$, output akan menjadi 1, sebaliknya output akan menjadi 0. Jadi setiap neuron dalam jaringan membagi masukan ruang menjadi dua daerah. Hal ini berguna untuk menyelidiki batas-batas antara daerah ini. Kita akan mulai dengan kasus sederhana dari single-neuron perceptron dengan dua input.

Single-Neuron Perceptron

Marilah kita mempertimbangkan dua input perceptron dengan satu neuron, seperti yang ditunjukkan pada gambar 4.2.



Gambar 4.2 Two-Input/Single-Output Perceptron

Keluaran dari jaringan ini ditentukan oleh

$$\begin{aligned} a &= hard\lim(n) = hard\lim(Wp + b) \\ &= hard\lim({}_1 w^T p + b) = hard\lim(w_{1,1} p_1 + w_{1,2} p_2 + b) \end{aligned} \qquad (4.8)$$

Batas Keputusan (Decision Boundary)

Batas keputusan ditentukan oleh vektor input dimana *n* input jaringan adalah nol.

$$n =_1 w^T p + b = w_{1,1} p_1 + w_{1,2} p_2 + b = 0. \quad (4.9)$$

Untuk membuat contoh yang lebih konret, marilah kita menetapkan nilai-nilai berikut untuk bobot dan bias:

$$w_{1,1} = 1,\ w_{1,2} = 1,\ b = -1 \quad (4.10)$$

Batas keputusan adalah

$$n =_1 w^T p + b = w_{1,1} p_1 + w_{1,2} p_2 + b = p_1 + p_2 - 1 = 0. \quad (4.11)$$

Ini mendefinisikan sebuah garis dalam ruang input. Di satu sisi garis jaringan output akan menjadi 0; pada baris dan pada sisi lain dari garis output akan menjadi 1. Untuk menarik garis, kita dapat menemukan titik-titik dimana ia memotong sumbu p_1 dan p_2. Untuk menemukan p_2 yang memintas tentukan $p_1 = 0$.

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1 \quad if \ \ p1 = 0 \quad (4.12)$$

Untuk menemukan p_1 yang memintas, tentukan $p_2 = 0$.

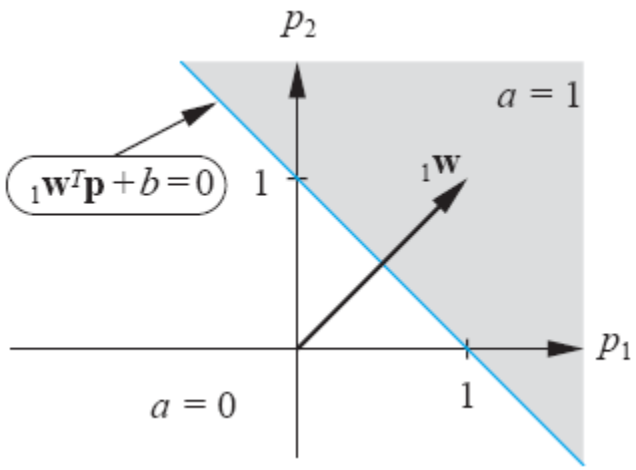
$$p1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1 \quad if \ \ p_2 = 0 \quad (4.13)$$

Batas keputusan yang dihasilkan diilustrasikan pada Gambar 4.3

Untuk mengetahui sisi mana dari batasan yang berhubungan dengan output 1, kita hanya perlu menguji satu titik. Untuk input p = [2 0]^T, keluaran jaringan akan menjadi

$$a = hard\lim(_1 w^T p + b) = hard\lim\left(\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} - 1 \right) = 1. \quad (4.14)$$

Oleh karena itu, Keluaran jaringan akan menjadi 1 untuk wilayah di atas dan ke kanan dari batasan keputusan. Wilayah ini ditandai dengan daerah yang diarsir pada Gambar 4.3.



Gambar 4.3 Batas Keputusan untuk Dua-Input Perceptron

Kita juga dapat menemukan batas keputusan secara grafis. Langkah pertama adalah perhatikan bahwa batas selalu orthogonal, seperti yang digambarkan dalam angka yang berdekatan. Batas didefinisikan oleh

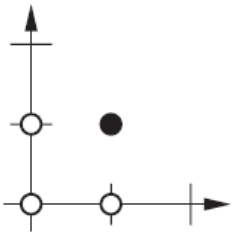
$$ {}_1w^T p + b = 0 \qquad (4.15)$$

Untuk semua titik di perbatasan, produk inner dari vektor input dengan bobot vektor adalah sama. Ini menunjukkan bahwa semua vektor input ini akan memiliki proyeksi yang sama ke vektor bobot, sehingga mereka harus terletak pada garis orthogonal dengan bobot vektor. Sebagai tambahan, setiap vektor di daerah bayangan dari Gambar 4.3 akan memiliki produk inner yang lebih besar dari *-b*, dan vektor di daerah yang tidak terarsir memiliki produk inner yang lebih kecil dari *-b*. Oleh karena itu vektor bobot ${}_1$w akan selalu menunjuk ke arah dimana neuron output adalah 1.

Setelah kita memilih salah satu vektor bobot dengan orientasi sudut yang benar, nilai bias dapat dihitung dengan memilih titik pada batas dan memenuhi persamaan (4.15)

Marilah kita menerapkan beberapa konsep ini untuk desain sebuah jaringan perceptron untuk melaksanakan fungsi logika sederhana: gerbang AND. Pasangan input/target untuk gerbang AND adalah

$$\left\{p1=\begin{bmatrix}0\\0\end{bmatrix}, t1=0\right\}\ \left\{p2=\begin{bmatrix}0\\1\end{bmatrix}, t2=0\right\}\ \left\{p3=\begin{bmatrix}1\\0\end{bmatrix}, t3=0\right\}\ \left\{p4=\begin{bmatrix}1\\1\end{bmatrix}, t4=1\right\}.$$

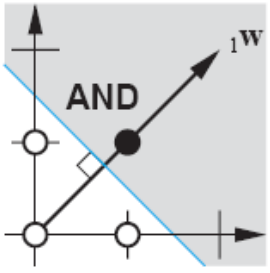


Gambar di sebelah kiri menggambarkan masalah secara grafis. Ia menampilkan ruang input, dengan masing-masing vector input ditandai sesuai dengan targetnya. Lingkaran gelap ● menunjukkan bahwa target adalah 1, dan lingkaran terang○ menunjukkan bahwa target adalah 0.

Langkah pertama dari desain adalah untuk memilih batas keputusan. Kami ingin memiliki garis yang memisahkan lingkaran hitam dan lingkaran terang. Ada sejumlah solusi yang tidak terbatas untuk masalah ini. Tampaknya masuk akal untuk memilih garis yang jatuh “di tengah” antara dua kategori input, seperti ditunjukkan pada gambar yang berdekatan.

Selanjutnya kami ingin memilih vector bobot yang orthogonal terhadap batas keputusan. Vektor bobot tsb dapat berupa sembarang panjang, sehingga terdapat kemungkinan yang tak terhingga. Satu pilihan adalah

$$_1w=\begin{bmatrix}2\\2\end{bmatrix} \tag{4.16}$$



Seperti yang ditunjukkan pada gambar ke kiri.

Akhirnya, kita perlu menemukan bias, *b*. Kita dapat melakukan ini dengan memilih sebuah titik pada batas keputusan dan memenuhi persamaan (4.15). jika kita menggunakan $p=\begin{bmatrix}1.5 & 0\end{bmatrix}$ kita menemukan

$$_1w^T p+b=\begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}1.5\\0\end{bmatrix}+b=3+b=0 \ \Rightarrow\ b=-3 \tag{4.17}$$

Sekarang kita bisa menguji jaringan pada salah satu pasangan input/target. Jika kita menerapkan p_2 ke jaringan, output akan

$$a = hard\lim({}_1 w^T p_2 + b) = hard\lim\left(\begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} - 3\right)$$
$$a = hard\lim(-1) = 0 \qquad (4.18)$$

Yang sama dengan output target t_2. Lakukan hal yang sama untuk menguji semua input apakah sudah diklasifikasi dengan benar.

Untuk berekperimen dengan batasan keputusan, gunakan disain jaringan Neural untuk memperagakan batas keputusan (nnd4db).

Multiple-Neuron Perceptron

Perhatikan bahwa untuk Perceptron dengan beberapa neuron, seperti dalam Gambar 4.1, akan ada satu batas keputusan untuk setiap neuron. Batas keputusan untuk neuron *i* akan ditentukan dengan

$${}_i w^T p + b_i = 0 \qquad (4.19)$$

Sebuah Single-neuron perceptron dapat mengklasifikasikan vektor-vektor input ke dalam dua kategori, karena output dapat berupa 0 atau 1. Sebuah multiple-neuron perceptron dapat mengelompokkan masukan/input ke dalam banyak kategori. Masing-masing kategori diwakili oleh vektor output yang berbeda. Karena setiap elemen dari vektor output dapat berupa 0 atau 1, terdapat total dari 2^S kategori yang mungkin, dimana *S* merupakan jumlah neuron.

Aturan Belajar Perceptron

Sekarang kita telah memeriksa kinerja jaringan perceptron, kita berada dalam posisi untuk memperkenalkan aturan pembelajaran perceptron. Aturan pembelajaran ini adalah contoh dari pelatihan dengan pengawasan, dimana aturan pembelajaran disediakan dengan serangkaian contoh perilaku jaringan yang tepat.

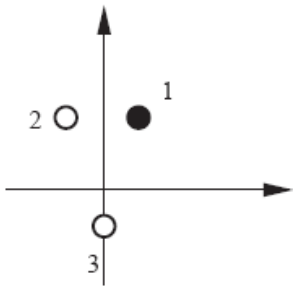
$$\{p_1, t_1\}, \{p_2, t_2\}, \cdots, \{p_q, t_q\}, \qquad (4.20)$$

Dengan p_q adalah sebuah input ke jaringan dan t_q berhubungan dengan target output. Karena setiap input diterapkan ke jaringan, output jaringan dibandingkan ke target. Aturan belajar kemudian menyesuaikan bobot dan bias jaringan untuk memindahkan keluaran/output jaringan lebih dekat ke sasaran.

Uji Masalah

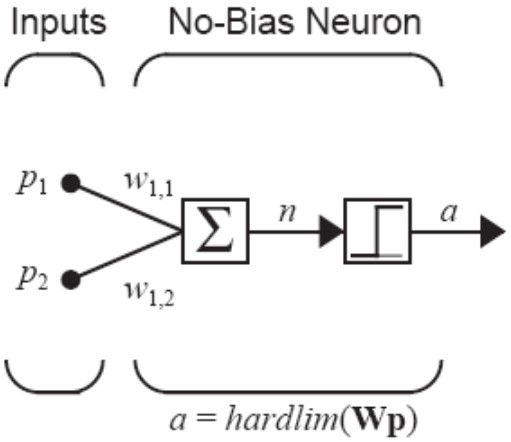
Pada presentasi kita mengenai aturan belajar perceptron kita akan mulai dengan sebuah uji masalah yang sederhana dan akan bereksperimen dengan aturan yang mungkin untuk mengembangkan beberapa intuisi tentang bagaimana aturan harus bekerja. Pasangan input/target untuk masalah pengujian kita adalah

$$\left\{p_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \left\{p_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \left\{p_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}.$$

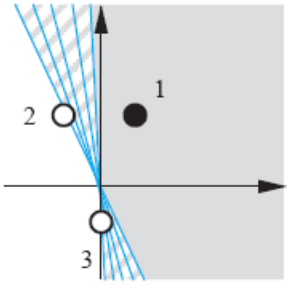


Masalah tersebut ditampilkan secara grafik dalam gambar di sebelah, dimana kedua vektor input yang targetnya adalah 0 direpresentasikan dengan lingkaran terang ○ , dan vector yang targetnya 1 direpresentasikan dengan lingkaran gelap ●. Ini merupakan masalah yang sangat sederhana, dan kita hampir bisa mendapatkan solusi dengan cara memeriksa. Kemudahan ini akan membantu kita mendapatkan beberapa pemahaman intuitif tentang konsep-konsep dasar aturan pembelajaran perceptron.

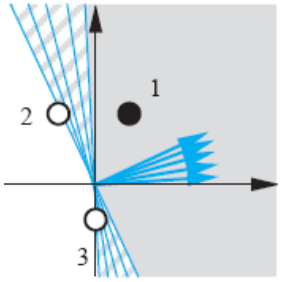
Jaringan untuk masalah ini harus memiliki dua-masukan dan satu keluaran. Untuk menyederhanakan pengembangan aturan belajar kita, kita akan mulai dengan jaringan tanpa bias. Jaringan akan memiliki dua parameter, $w_{1,1}$ dan $w_{1,2}$, seperti yang ditunjukkan pada Gambar 4.4



Gambar 4.4. Uji Masalah Jaringan



Dengan menghapus bias kita dihadapkan dengan sebuah jaringan yang batas keputusannya harus melalui titik asal. Kita perlu memastikan bahwa jaringan ini masih mampu memecahkan masalah uji. Harus ada batas keputusan yang diperbolehkan yang dapat memisahkan vektor p_2 dan p_3 dari vektor p_1. Gambar sebelah kiri menunjukkan bahwa memang terdapat sejumlah batasan yang tidak terbatas.



Gambar di sebelah menunjukkan vektor bobot yang sesuai dengan batas keputusan yang diperbolehkan. (Ingat bahwa vektor bobot adalah orthogonal terhadap batas keputusan). Kita ingin sebuah aturan belajar yang akan menemukan suatu vektor bobot yang menunjuk pada salah satu arah. Ingat bahwa panjang vektor bobot tidak menjadi masalah, hanya arahnya penting.

Membangun Aturan Belajar

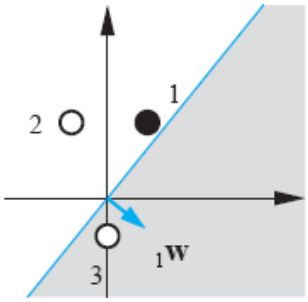
Pelatihan dimulai dengan menetapkan beberapa nilai awal untuk parameter jaringan. Dalam hal ini kita melatih jaringan dengan jaringan two-input/single-output tanpa bias, jadi kita hanya perlu menginisialisasi dua bobot. Di sini kita menetapkan elemen-elemen vektor bobot, ${}_1w$ Untuk menghasilkan nilai random berikut:

$$ {}_1w^T = \begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \tag{4.21}$$

Sekarang kita akan mulai menampilkan vektor input ke jaringan. Kita mulai dengan p_1

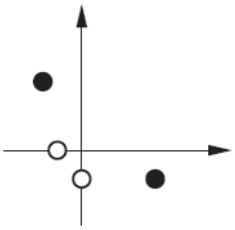
$$ a = hard\lim\left({}_1w^T p_1\right) = hard\lim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \tag{4.22}$$

$$ a = hard\lim(-0.6) = 0 $$



Jaringan tidak mengembalikan nilai yang benar. Output jaringan adalah 0, sementara target respon, t_1, adalah 1.

Kita bisa melihat apa yang terjadi dengan melihat diagram yang di sebelah. Hasil vektor bobot awal dalam suatu batas keputusan yang salah mengklasifikasikan vector p_1. Kita perlu mengubah vektor bobot



sehingga poin lebih ke arah p_1, sehingga di masa mendatang memiliki kesempatan yang lebih baik dalam mengklasifikasikannya dengan benar.

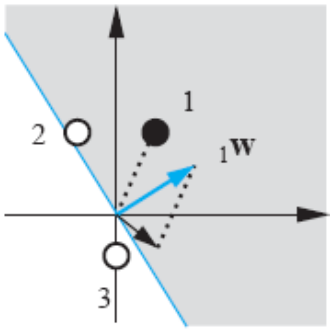
Salah satu pendekatan akan menetapkan ${}_1w$ sama dengan p_1. Ini adalah sederhana dan akan memastikan bahwa p_1 telah diklasifikasi dengan benar di masa mendatang. Diagram di kiri bawah menunjukkan masalah yang tidak dapat diselesaikan dengan vektor bobot menunjuk langsung pada salah satu dari dua kelas 1 vektor. Jika kita menerapkan aturan bahwa ${}_1w = p$ setiap kali salah satu vektor ini tidak terklasifikasi, bobot jaringan hanya akan bergerak mundur terus dan tidak akan pernah menemukan suatu solusi.

Kemungkinan lainnya ialah akan menambahkan p_1 ke ${}_1w$. Penambahan p_1 ke ${}_1w$ akan membuat titik ${}_1w$ lebih ke arah p_1. Pengulangan presentasi dari p_1 akan menyebabkan arah dari ${}_1w$ untuk secara asimtotik mendekati arah p_1. Aturan ini dapat dinyatakan :

$$If\ t = 1\ dan\ a = 0, then\ {}_1w^{new} =_1 w^{old} + p \tag{4.23}$$

Penerapan aturan ini untuk masalah pengujian kita menghasilkan nilai baru untuk ${}_1w$:

$${}_1w^{new} =_1 w^{old} + p_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}. \tag{4.24}$$



Operasi ini diilustrasikan dalam gambar sebelah. Kita sekarang beralih ke vektor input berikutnya dan akan melanjutkan membuat perubahan terhadap bobot dan cycling melalui input sampai mereka semua diklasifikasikan dengan benar.

Vektor input berikutnya adalah p_2. Ketika itu disajikan ke jaringan kita menemukan :

$$\begin{aligned} a &= hard\lim\left({}_1w^T p_2\right) = hard\lim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) \\ &= hard\lim(0.4) = 1 \end{aligned} \tag{4.25}$$

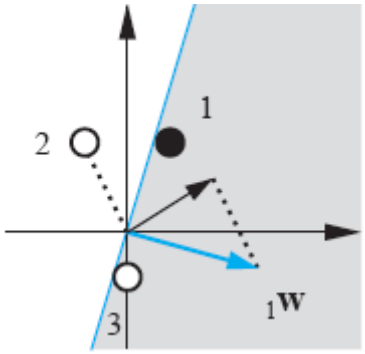
Target t_2 yang terkait dengan p_2 adalah 0 dan output a adalah 1. Kelas vektor 0 tidak terklasifikasi sebagai 1. Karena kita sekarang ingin memindahkan vektor bobot ${}_1w$ menjauh dari

input, kita dapat dengan mudah mengubah penambahan dalam persamaan (4.23) menjadi pengurangan:

$$If\ t = 0\ dan\ a = 1, then\ {}_1w^{new} =_1 w^{old} - p \tag{4.26}$$

Jika kita menerapkan hal ini untuk menguji masalah kita akan menemukan :

$${}_1w^{new} =_1 w^{old} - p_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}, \tag{4.27}$$



Yang digambarkan dalam gambar di sebelah.

Sekarang kita tampilkan vektor ketiga yakni p_3:

$$a = hard\lim\left({}_1w^T p_3\right) = hard\lim\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) \tag{4.28}$$
$$= hard\lim(0.8) = 1.$$

Hasil ${}_1w$ saat ini dalam batas keputusan bahwa p_3 tidak terklasifikasi. Ini adalah situasi dimana kita telah memiliki sebuah aturan, sehingga ${}_1w$ akan diubah kembali, tergantung pada persamaan (4.26):

$${}_1w^{new} =_1 w^{old} - p_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}, \tag{4.29}$$

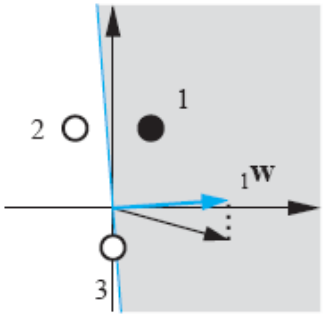


Diagram di sebelah kiri menunjukkan bahwa perceptron akhirnya belajar untuk menggolongkan tiga vektor dengan benar. Jika kita menghadirkan salah satu vektor input untuk neuron, ia akan menampilkan kelas yang benar untuk vektor input tersebut.

Ini membawa kita ke aturan ketiga dan yang terakhir: jika ia bekerja, jangan lakukan perbaikan.

$$If\ t = a, then\ {}_1w^{new} =_1 w^{old}. \tag{4.30}$$

Berikut ini adalah tiga aturan, yang mencakup semua kemungkinan kombinasi output dan nilai-nilai sasaran:

$$\begin{aligned} &If\ t = 1\ dan\ a = 0, then\ {}_1w^{new} =_1 w^{old} + p \\ &If\ t = 0\ dan\ a = 1, then\ {}_1w^{new} =_1 w^{old} - p \\ &If\ t = a, then\ {}_1w^{new} =_1 w^{old}. \end{aligned} \tag{4.31}$$

Aturan belajar Yang Diseragamkan (Unified Learning Rule)

Tiga aturan dalam Persamaan. (4.31) dapat ditulis kembali sebagai satu ekspresi. Pertama kita akan mendefinisikan sebuah variable baru, maka kesalahan perceptron *e* :

$$e = \text{t} - \text{a} \tag{4.32}$$

Sekarang kita dapat menulis kembali tiga aturan dari persamaan di (4.31) sebagai :

$$\begin{aligned} &If\ e = 1,\ then\ {}_1w^{new} =_1 w^{old} + p \\ &If\ e = -1, then\ {}_1w^{new} =_1 w^{old} - p \\ &If\ e = 0, then\ {}_1w^{new} =_1 w^{old}. \end{aligned} \tag{4.34}$$

Lihat dengan hati-hati pada dua aturan pertama dalam persamaan (4.33), kita dapat melihat bahwa tanda p adalah sama dengan tanda pada kesalahan, *e*. Selanjutnya, tidak adanya p dalam aturan ketiga sesuai dengan e dari 0. Dengan demikian, kita dapat menyatukan tiga aturan tersebut menjadi satu ekspresi :

$${}_1w^{new} =_1 w^{old} + ep =_1 w^{old} + (t-a)p. \tag{4.34}$$

Aturan ini dapat diperluas untuk melatih bias dengan catatan bahwa bias hanya bobot yang inputnya selalu 1. Dengan demikian kita dapat mengganti input pada persamaan (4.34) dengan input ke bias adalah 1. Hasilnya adalah aturan perceptron untuk bias :

$$b^{new} = b^{old} + e \tag{4.35}$$

Pelatihan Multiple-Neuron Perceptron

Aturan perceptron, seperti yang diberikan oleh persamaan (4.34) dan persamaan (4.35), memperbaharui vektor bobot dari suatu neuron perceptron. Kita bisa generalisasi aturan ini untuk Multiple-Neuron perceptron dari Gambar 4.1. sebagai berikut. Memperbaharui baris ke –i dari matrik bobot yang digunakan:

$${}_i w^{new} =_i w^{old} + e_i p \tag{4.36}$$

Untuk memperbaharui elemen ke-i dari vector bias menggunakan:

$$b_i^{new} = b_i^{old} + e_i. \tag{4.37}$$

Aturan perceptron

Aturan perceptron dapat ditulis dalam notasi matriks:

$$W^{new} = W^{old} + ep^T, \tag{4.38}$$

Dan

$$b^{new} = b^{old} + e. \tag{4.39}$$

Untuk menguji aturan pembelajaran perceptron, pertimbangkan kembali masalah pengenalan apel/jeruk dari bab sebelumnya.

Vektor input/output prototipe akan menjadi

$$\left\{ p_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix},\ t_1 = [0] \right\} \left\{ p_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ t_2 = [1] \right\}. \tag{4.40}$$

(Perhatikan bahwa kita menggunakan 0 sebagai target output untuk pola jeruk, p_1, bukan -1, seperti yang digunakan dalam bab sebelumnya. Ini adalah karena kita menggunakan fungsi transfer hardlim, bukan hardlims.)

Biasanya bobot dan bias yang diinisialisasi ke angka acak yang kecil. Anggaplah bahwa di sini kita mulai dengan bobot awal matrik dan bias :

$$W = [0.5 \quad -1 \quad -0.5],\ b = 0.5. \tag{4.41}$$

Langkah pertama adalah menerapkan vektor input pertama, p_1, ke jaringan:

$$\begin{aligned} a &= hard\lim\left(Wp_1 + b\right) = hard\lim\left([0.5 \quad -1 \quad -0.5] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5 \right) \\ &= hard\lim(2.5) = 1 \end{aligned} \tag{4.42}$$

Kemudian kita menghitung kesalahan:

$$e = t_1 - a = 0 - 1 = -1. \tag{4.43}$$

Perbaikan bobot adalah

$$\begin{aligned} W^{new} &= W^{old} + ep^T = [0.5 \quad -1 \quad -0.5] + (-1)[1 \quad -1 \quad -1] \\ &= [-0.5 \quad 0 \quad 0.5]. \end{aligned} \tag{4.44}$$

Perbaikan bias adalah :

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5. \tag{4.45}$$

Hal ini melengkapi iterasi pertama.

Iterasi kedua dari aturan perceptron adalah:

$$a = hard\lim(Wp_2 + b) = hard\lim\left(\begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (-0.5) \right) \tag{4.46}$$

$$= hard\lim(-0.5) = 0$$

$$e = t_2 - a = 1 - 0 = 1 \tag{4.47}$$

$$\begin{aligned} W^{new} &= W^{old} + ep^T = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} + (1)\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix}. \end{aligned} \tag{4.48}$$

$$b^{new} = b^{old} + e = -0.5 + (1) = 0.5. \tag{4.49}$$

Iterasi ketiga dimulai lagi dengan vektor input pertama:

$$a = hard\lim(Wp_1 + b) = hard\lim\left(\begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + (0.5) \right) \tag{4.50}$$

$$= hard\lim(0.5) = 1$$

$$e = t_1 - a = 0 - 1 = -1 \tag{4.51}$$

$$\begin{aligned} W^{new} &= W^{old} + ep^T = \begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix} + (-1)\begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 & 2 & 0.5 \end{bmatrix}. \end{aligned} \tag{4.52}$$

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5. \tag{4.53}$$

Jika anda lanjutkan dengan iterasi anda akan menemukan bahwa kedua masukan vector akan sekarang diklasifikasikan dengan benar. Algoritma telah bertemu ke sebuah solusi. Catatan bahwa batas keputusan akhir tidak sama dengan yang kami kembangkan di bab sebelumnya, meskipun kedua batasan benar mengklasifikasikan dua masukan vektor secara benar.

Untuk percobaan dengan aturan pembelajaran perceptron, gunakan *Neural Network Design Demonstration Perceptron Rule* (nnd4pr).

Keterbatasan

Aturan pembelajaran Perceptron dijamin untuk mengumpul pada suatu solusi dalam satu jumlah tahap yang terbatas, sepanjang terdapat suatu solusi. Ini membawa kita pada satu pertanyaan penting. Apa permasalahan yang bisa dipecahkan suatu perceptron? Mengingatkan kembali bahwa suatu single-neuron perceptron memungkinkan untuk membagi ruang input ke dalam dua daerah. Perbatasan antara daerah didefinisikan oleh persamaan

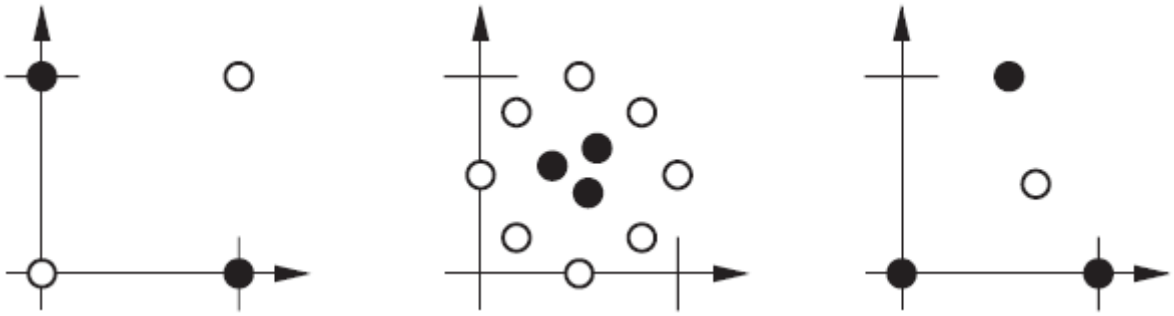
$$_1w^T p + b = 0 \qquad (4.77)$$

Ini merupakan suatu batasan linier (hyperplane). Perceptron dapat digunakan untuk mengklasifikasikan vektor input yang dapat dipisahkan oleh suatu batasan linier. Kita menyebut vektor seperti ini vektor yang dapat dipisahkan secara linear. Contoh gerbang logika AND pada halaman 4-7 menggambarkan satu contoh dua-dimensi dari masalah yang dapat dipisah-pisah secara linear. Masalah pengenalan buah apel/jeruk dalam Chapter 3 adalah satu contoh tiga-dimensi.

Sayangnya, banyak permasalahan yang tidak dapat dipisah-pisah secara linear. Contoh klasik adalah gerbang XOR. Input/pasangan target untuk gerbang XOR adalah

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\}.$$

Masalah ini digambarkan secara grafis pada sisi kiri dari gambar 4.6, yang juga menunjukan dua permasalahn lain yang tidak dapat dipisahkan secara linear. Cobalah menggambar satu garis lurus antara vektor dengan target dari 1 dan yang dengan target 0 pada diagram apapun dari gambar 4.6.

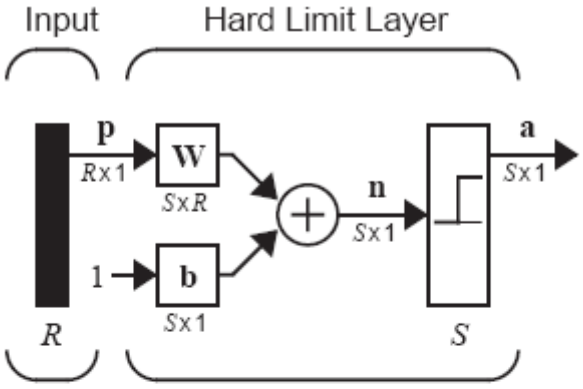


Gambar 4.6 Permasalahan Yang Tidak Dapat Dipisahkan Secara Linear

Itu adalah ketidak-mampuan dari dasar perceptron untuk memecahkan seperti permasalahan sederhana yang berupa led, pada sebagian, untuk pengurangan dalam tertarik akan penelitian jaringan neural selama 1970-an. Rosenblatt telah menyelidiki jaringan lebih rumit, dimana dia merasakan akan mengatasi keterbatasan dari dasar perceptron, tetapi dia tidak pernah mampu untuk secara efektif memperluas aturan perceptron ke beberapa jaringan. Pada Bab 11 kita akan memperkenalkan perceptrons banyak lapisan, yang dapat memecahkan permasalahan klasifikasi, dan akan menggambarkan algoritma *backpropagation*, yang dapat digunakan untuk melatih mereka.

Ringkasan Hasil

Arsitektur Perceptron



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b}) \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

Batasan Keputusan

$${}_i\mathbf{w}^T\mathbf{p} + b_i = 0 .$$

Batasan keputusan adalah selalu ortogonal untuk vektor bobot. Single-layer perceptrons hanya dapat mengklasifikasikan vektor yang terpisah secara linear.

Perceptron Learning Rule

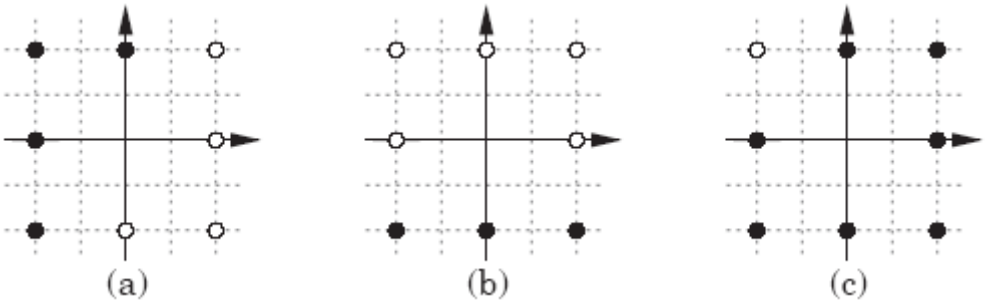
$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

where $\mathbf{e} = \mathbf{t} - \mathbf{a}$.

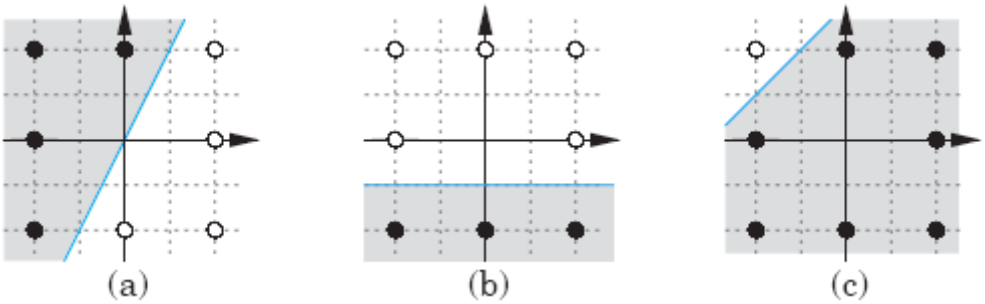
Pemecahan Masalah

P4.1 Pecahkan tiga permasalahan klasifikasi sederhana yang diperlihatkan di dalam gambar P4.1 dengan cara menggambar suatu batasan keputusan. Temukan nilai bobot dan bias dimana hasil dalam single-neuron perceptrons dengan batasan keputusan yang terpilih.

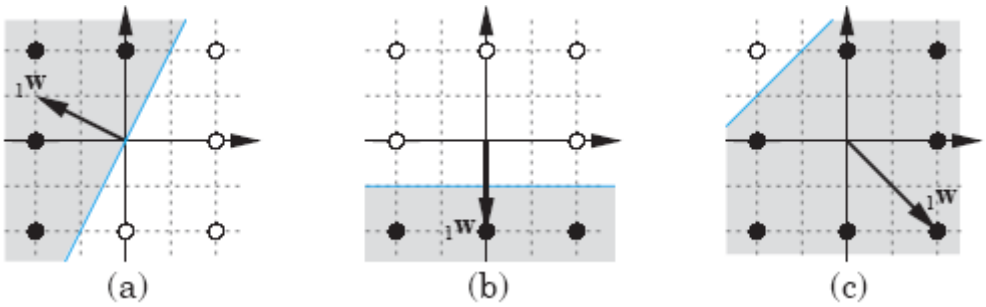


Gambar P4.1 Permasalah Klasifikasi Sederhana

Pertama kita menggambar satu baris antara setiap sekumpulan titik data gelap dan ringan.



Tahap berikutnya adalah mencari bobot dan bias. Vektor bobot harus ortogonal terhadap batasan-batasan keputusan, dan penunjukan dalam arah titik untuk digolongkan sebagai 1 (titik gelap). Vektor bobot dapat mempunyai panjang berapapun yang kita suka.



Inilah sekumpulan pilihan untuk vektor bobot:

$$(a)\ {}_1w^T = [-2 \quad 1], (b)\ {}_1w^T = [0 \quad -2], (c)\ {}_1w^T = [2 \quad -2].$$

Sekarang kita mencari nilai bias untuk setiap perceptron dengan cara mengambil suatu titik pada batasan keputusan dan memenuhi Persamaan (4.15).

$$ {}_1w^T p + b = 0$$
$$ b = -{}_1w^T p$$

Hal ini memberikan kita tiga bias berikut:

$$(a)\ b = [-2 \quad 1]\begin{bmatrix}0\\0\end{bmatrix} = 0,\ (b)\ b = -[0 \quad -2]\begin{bmatrix}0\\-1\end{bmatrix} = -2,\ (c)\ b = -[2 \quad -2]\begin{bmatrix}-2\\1\end{bmatrix} = 6.$$

Kita sekarang bisa memeriksa solusi kita terhadap titik asli. Di sini kita menguji jaringan pertama pada vektor input $p = [-2 \quad 2]$.

$$\begin{aligned} a &= hard\lim\left({}_1w^T p + b\right) \\ &= hard\lim\left([-2 \quad 1]\begin{bmatrix}-2\\2\end{bmatrix} + 0\right) \\ &= hard\lim(6) = 1 \end{aligned}$$

Kita bisa menggunakan MATLAB untuk mengotomatiskan proses pengujian dan untuk mencoba titik yang baru. Inilah jaringan pertama yang digunakan untuk mengklasifikasikan suatu titik yang tidak berada dalam masalah sebenarnya.

```
w=[-2 1]; b = 0;
a = hardlim(w*[1;1]+b)
a =
        0
```

P4.2 Convert masalah klasifikasi yang didefinisikan di bawah ke dalam suatu definisi masalah ekivalen terdiri dari ketidaksamaan batasan nilai bobot dan bias.

$$\left\{p_1 = \begin{bmatrix}0\\2\end{bmatrix}, t_1 = 1\right\} \left\{p_2 = \begin{bmatrix}1\\0\end{bmatrix}, t_2 = 1\right\} \left\{p_3 = \begin{bmatrix}0\\-2\end{bmatrix}, t_3 = 0\right\} \left\{p_4 = \begin{bmatrix}2\\0\end{bmatrix}, t_4 = 0\right\}$$

Masing-masing target t_i mengindikasikan apakah input jaringan dalam respon untuk p_i harus lebih kecil dari 0, atau lebih besar dari atau sama dengan 0 atau tidak. Sebagai contoh, karena t_1 adalah 1, kita tahu bahwa input jaringan yang erhubungan dengan p1 harus lebih besar dari atau sama dengan 0. Sehingga kita dapat ketidaksamaan berikut:

$$\begin{aligned} Wp_1 + b &\geq 0 \\ 0w_{1,1} + 2w_{1,2} + b &\geq 0 \\ 2w_{1,2} + b &\geq 0 \end{aligned}$$

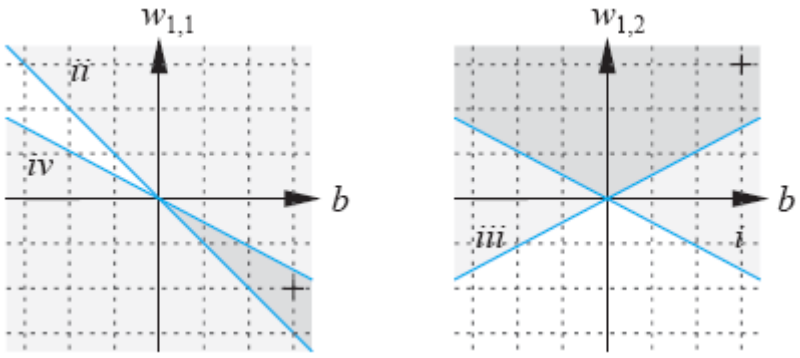
Terapkan prosedur yang sama ke pasangan input/target untuk $\{p_2, t_2\}$, $\{p_3, t_3\}$dan $\{p_4, t_4\}$hasil-hasil dalam himpunan ketidaksamaan berikut.

$$\begin{aligned} 2w_{1,2} + b &\geq 0 \quad (i) \\ w_{1,1} + b &\geq 0 \quad (ii) \\ -2w_{1,2} + b &< 0 \quad (iii) \\ 2w_{1,1} + b &< 0 \quad (iv) \end{aligned}$$

Memecahkan suatu himpunan ketidaksamaan adalah lebih sulit dibandingkan memecahkan suatu himpunan persamaan. Satu kompleksitas ditambahkan seringkali sejumlah solusi tanpa batas (seperti sering ada sejumlah batasan-batasan keputusan linier tanpa batas yang bisa memecahkan suatu masalah klasifikasi yang terpisah secara linear).

Bagaimanapun, oleh karena kesederhanaan dari masalah ini, kita bisa memecahkannya dengan membuat grafik ruang solusi yang didefinisikan oleh ketidaksamaan. Catat bahwa $w_{1,1}$ hanya

muncul dalam ketidaksamaan (*ii*) dan (*iv*), dan $w_{1,2}$ hanya muncul dalam ketidaksamaan (*i*) dan (*iii*). Kita dapat meletakkan masing-masing pasangan ketidaksamaan dengan dua grafik.



Nilai bobot dan bias apapun yang jatuh dalam kedua daerah abu-abu yang gelap tersebut akan memecahkan masalah klasifikasi.

Berikut ini salah satu dari solusi :

$$W = \begin{bmatrix} -2 & 3 \end{bmatrix} \qquad b = 3.$$

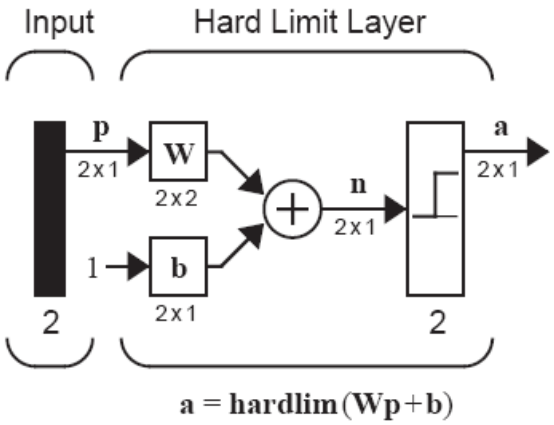
P4.3 Kita mempunyai satu masalah klasifikasi dengan empat kelas dari vektor input. Empat kelas tersebut adalah

$$class\ 1: \left\{ p1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, p2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}, class\ 2: \left\{ p3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, p4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\},$$

$$class\ 3: \left\{ p5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, p6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\}, class\ 4: \left\{ p7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, p8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right\}$$

Rancang sebuah jaringan perceptron untuk memecahkan masalah ini.

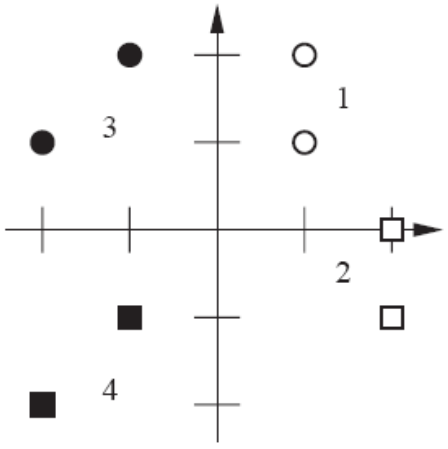
Untuk memecahkan sebuah masalah dengan 4 kelas vector input kita akan membutuhkan sebuah perceptron dengan paling sedikit dua neuron, karena sebuah *S*-neuron perceptron dapat mengkategorikan 2^S kelas. Perceptron dua neuron ditunjukkan dalam Gambar P4.2.



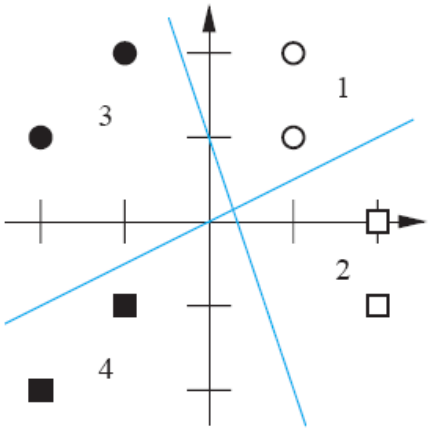
Gambar P4.2 Two-Neuron Perceptron

Marilah kita mulai dengan menampilkan vector input, seperti dalam Gambar P4.3. Lingkaran terang ◯mengindikasikan kelas vector 1, segiempat terang □ mengindikasikan kelas vector 2, lingkaran gelap ● mengindikasikan kelas vector 3, dan segiempat gelap ■ mengindikasikan kelas vector 4.

Sebuah perceptron dua neuron membuat dua batasan keputusan. Oleh karena itu, untuk membagi ruang input ke dalam empat kategori, kita perlu untuk memiliki satu batasan keputusan yang membagi empat kelas ke dalam dua himpunan dari dua. Sisa batasan kemudian harus mengisolasi masing-masing kelas. Dua batasan tersebut digambarkan dalam Gambar P4.4. Sekarang kita tahu bahwa pola kita adalah terpisah secara linear.



Gambar P4.3 Vektor Input untuk Masalah P4.3



Gambar P4.4 Batasan Keputusan yang bersifat sementara untuk masalah P4.3

Vektor bobot harus ortogonal terhadap batasan-batasan keputusan dan harus menunjuk ke arah daerah dimana output neuron adalah 1. Tahap berikutnya harus memutuskan bahwa sisi dari setiap perbatasan harus menghasilkan sebuah 1. Satu pilihan digambarkan di dalam gambar P4.5, dimana daerah yang di-shadow merepresentasikan output 1. Shadow yang paling gelap nilai target dari

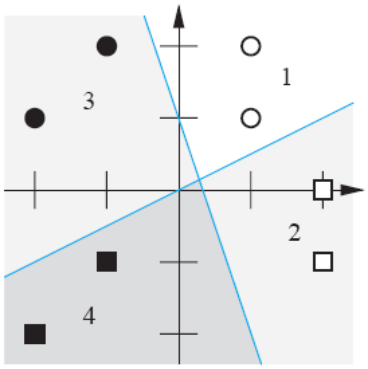
$$class\ 1: \left\{ t1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}, class\ 2: \left\{ t3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\},$$
$$class\ 3: \left\{ t5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}, class\ 4: \left\{ t7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

Kita sekarang dapat memilih vektor-vektor bobot:

$$_1w = \begin{bmatrix} -3 \\ -1 \end{bmatrix} \quad dan \quad _2w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Perhatikan bahwa panjang vector bobot tidak penting, hanya arah mereka. Mereka harus orthogonal terhadap batasan keputusan. Kita sekarang dapat menghitung bias dengan mengambil sebuah titik pada batasan dan memenuhi Persamaan (4.15):

$$b_1 = -_1w^T p = -\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1,$$
$$b_2 = -_2w^T p = -\begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0,$$



Gambar P4.5 Daerah Keputusan untuk Masalah P4.3

Dalam bentuk matrik kita memiliki

$$W = \begin{bmatrix} {}_1w^T \\ {}_2w^T \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & -2 \end{bmatrix} \quad dan \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

Yang melengkapi disain/rancangan kita.

P4.4 Pecahkan masalah klasifikasi berikut dengan aturan perceptron. Terapkan masing-masing vektor input sebanyak pengulangan yang diperlukan untuk memastikan bahwa masalah dapat dipecahkan. Gambar suatu grafik dari masalah hanya setelah anda telah menemukan satu solusi.

$$\left\{p_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0\right\} \left\{p_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1\right\} \left\{p_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0\right\} \left\{p_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1\right\}$$

Gunakan bobot dan bias awal sbb:

$$W(0) = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad b(0) = 0.$$

Kita mulai dengan menghitung output perceptron *a* untuk vector input yang pertama yakni p_1, menggunakan bobot dan bias awal.

$$a = hard\lim(W(0)p1) + b(0)$$
$$= hard\lim\left(\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hard\lim(0) = 1$$

Output *a* tidak sama dengan nilai target t_1, sehingga kita menggunakan aturan perceptron untuk menemukan bobot dan bias yang baru berdasarkan pada error.

$$e = t_1 - a = 0 - 1 = -1$$
$$W(1) = W(0) + ep_1^T = \begin{bmatrix}0 & 0\end{bmatrix} + (-1)\begin{bmatrix}2 & 2\end{bmatrix} = \begin{bmatrix}-2 & -2\end{bmatrix}$$
$$b(1) = b(0) + e = 0 + (-1) = -1$$

Kita sekarang menerapkan vector input yang kedua yakni p_2, menggunakan bobot dan bias yang baru (sudah diubah).

$$a = hard\lim(W(1)p_2) + b(1)$$
$$= hard\lim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix} - 1\right) = hard\lim(1) = 1$$

Kali ini output *a* adalah sama dengan target t_2. Penerapa aturan perceptron tidak akan menghasilkan perubahan apapun.

$$W(2) = W(1) \qquad b(2) = b(1)$$

Kita sekarang menerapkan vector input ketiga.

$$a = hard\lim(W(2)p_3) + b(2)$$
$$= hard\lim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-2\\2\end{bmatrix} - 1\right) = hard\lim(-1) = 0$$

Output menanggapi vector input p_3 adalah sama dengan target t_3, sehingga tidak ada perubahan terhadap nilai bobot maupun bias.

$$W(3) = W(2) \qquad b(3) = b(2)$$

Sekarang kita lanjutkan pada vector input yang terakhir yakni p_4.

$$a = hard\lim(W(3)p_4) + b(3)$$
$$= hard\lim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix} - 1\right) = hard\lim(-1) = 0$$

Kali ini output *a* tidak tepat sama dengan target t_4. Aturan perceptron akan menghasilkan nilai W dan b yang baru.

$$e = t_4 - a = 1 - 0 = 1$$
$$W(4) = W(3) + ep_4^T = \begin{bmatrix} -2 & -2 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -1 \end{bmatrix}$$
$$b(4) = b(3) + e = -1 + (1) = 0$$

Sekarang kita harus menguji vector pertama yakni p_1 kembali. Kali ini output a sama dengan target t_1.

$$a = hard\lim(W(4)p_1) + b(4)$$
$$= hard\lim\left(\begin{bmatrix} -3 & -1 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right) = hard\lim(-8) = 0$$

Oleh karena itu tidak ada perubahan yang terjadi.

$$W(5) = W(4) \qquad b(5) = b(4)$$

Presentasi kedua dari p2 menghasilkan dalam satu error dan oleh karena itu suatu nilai bobot dan bias yang baru harus ditetapkan.

$$a = hard\lim(W(5)p_2) + b(5)$$
$$= hard\lim\left(\begin{bmatrix} -3 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0\right) = hard\lim(-1) = 0$$

Berikut ini nilai-nilai yang baru tersebut:

$$e = t_2 - a = 1 - 0 = 1$$
$$W(6) = W(5) + ep_2^T = \begin{bmatrix} -3 & -1 \end{bmatrix} + (1)\begin{bmatrix} 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -3 \end{bmatrix}$$
$$b(6) = b(5) + e = 0 + 1 = 1$$

Perputaran melalui masing-masing vektor input sekali lagi menghasilkan tidak ada error.

$$a = hard\lim(W(6)p_3) + b(6) = hard\lim\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \begin{bmatrix} -2 \\ 2 \end{bmatrix} + 1 \right) = 0 = t_3$$

$$a = hard\lim(W(6)p_4) + b(6) = hard\lim\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 1 \right) = 1 = t_4$$

$$a = hard\lim(W(6)p_1) + b(6) = hard\lim\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1 \right) = 0 = t_1$$

$$a = hard\lim(W(6)p_2) + b(6) = hard\lim\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1 \right) = 1 = t_2$$

Oleh karena itu algoritma telah memusat. Solusi akhir adalah:

$$\mathbf{W} = \begin{bmatrix} -2 & -3 \end{bmatrix} \qquad b = 1 .$$

Sekarang kita bisa membuat grafik data pelatihan dan batasan keputusan dari solusi.

Batasan keputusan diberikan oleh

$$n = W_p + b = w_{1,1} p_1 + w_{1,2} p_2 + b = -2p_1 - 3p_2 + 1 = 0.$$

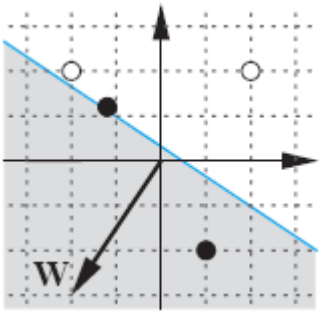
Untuk menemukan p_2 memintas batasan keputusan, tentukan $p_1 = 0$:

$$p2 = -\frac{b}{w_{1,2}} = -\frac{1}{-3} = \frac{1}{3} \qquad jika \quad p_1 = 0.$$

Untuk menemukan p_1 yang memintas, tentukan $p_2 = 0$:

$$p1 = -\frac{b}{w_{1,1}} = -\frac{1}{-2} = \frac{1}{2} \qquad jika \quad p_2 = 0.$$

Batasan keputusan yang dihasilkan digambarkan di dalam Gambar P4.6.



Gambar P4.6 Batasan Keputusan untuk Masalah P4.4

Perhatikan bahwa batasan keputusan yang jatuh secara kebetulan salah satu dari vektor pelatihan. Hal ini adalah bisa diterima, dengan mengetahui definisi masalah, karena fungsi hard limit mengembalikan 1 ketika diberikan satu input 0, dan target untuk vektor di dalam pertanyaan adalah tentu saja 1.