PDEEC – Machine Learning 2018/19

Lecture - Classification: non-generative models

Jaime S. Cardoso
jaime.cardoso@inesctec.pt

INESC TEC and Faculdade Engenharia, Universidade do Porto

Oct. 25, 2018

Taxonomy of classification methods

- generative models:
  - first determine the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$
  - infer the prior class probabilities $p(\mathcal{C}_k)$
  - Use Bayes' theorem in the form

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

    to find the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$
  - use these posterior probabilities to make class assignments, assigning each new $\mathbf{x}$ to one of the classes
- discriminative models:
  - infer directly $p(\mathcal{C}_k|\mathbf{x})$
  - use these posterior probabilities to make class assignments, assigning each new $\mathbf{x}$ to one of the classes
- models that find directly the discriminant function, mapping each input $\mathbf{x}$ directly onto a class label.

Logistic Regression

Idea:

- (Naive) Bayes allows computing $P(y|\mathbf{x})$ by learning $P(y)$ and $P(\mathbf{x}|y)$
- Why not learn $P(y|\mathbf{x})$ directly?

Logistic Regression

- Consider learning $f : \mathbf{x} \rightarrow y$, where
  - $\mathbf{x}$ is a vector of real-valued features, $< x_1 \cdots x_n >$
  - y is boolean
  - assume all x_i are conditionally independent given y
  - model $P(x_i|y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$
  - model $P(y)$ as Bernoulli(p)
- What does that imply about the form of $P(y|\mathbf{x})$?

$$P(y = 1|\mathbf{x} =< x_1, \cdots, x_n >= \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Logistic Regression

Derive form for $P(y|\mathbf{x})$ for continuous x_i

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

$$= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

$$= \frac{1}{1 + \exp(\ (\ln \frac{1-\pi}{\pi}) + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

$$P(x \mid y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

$$\sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Logistic Regression

Very convenient!

$$P(y = 1|\mathbf{x} =< x_1, \cdots, x_n >) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

implies

$$P(y = 0|\mathbf{x} =< x_1, \cdots, x_n >) =$$

implies

$$\frac{P(y = 0|\mathbf{x})}{P(y = 1|\mathbf{x})} =$$

implies

$$\ln \frac{P(y = 0|\mathbf{x})}{P(y = 1|\mathbf{x})} =$$

Logistic Regression

Very convenient!

$$P(Y = 1|X =< X_1, ...X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 0|X =< X_1, ...X_n >) = \frac{exp(w_0 + \sum_i w_i X_i)}{1 + exp(w_0 + \sum_i w_i X_i)}$$

implies

$$\frac{P(Y = 0|X)}{P(Y = 1|X)} = exp(w_0 + \sum_i w_i X_i)$$

linear classification rule!

implies

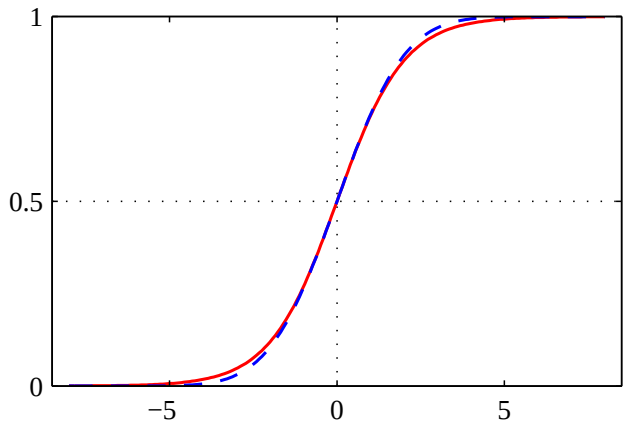
$$\ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_i w_i X_i$$

Logistic Regression

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}}$$

$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1+\exp(-z)}$ $\qquad$ $z = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$

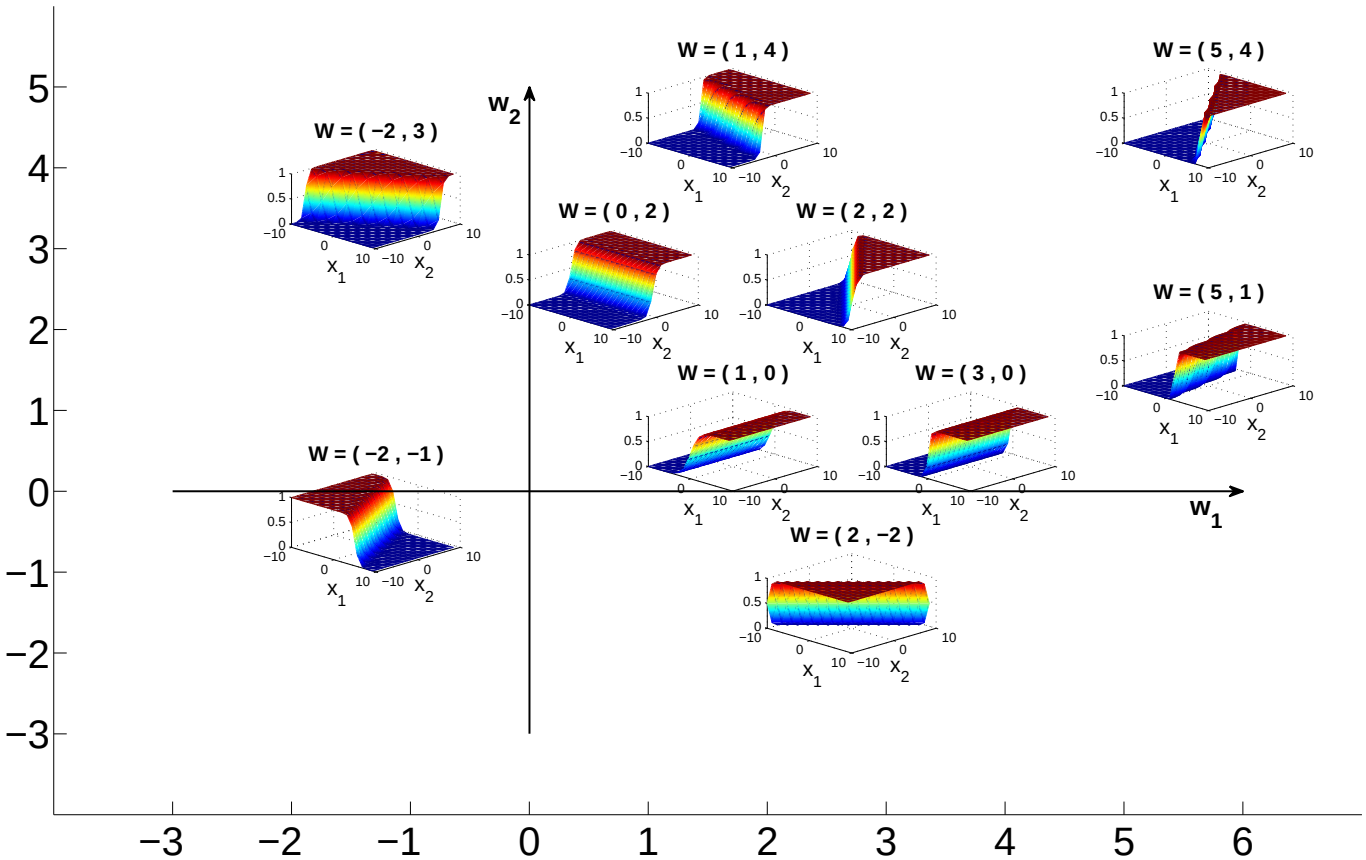
- $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the logistic sigmoid function



- $\sigma(-z) = 1 - \sigma(z)$
- $z = \ln \frac{\sigma}{1-\sigma}$ represents the log of the ratio of probabilities $\ln \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}$

Logistic Regression

logistic sigmoid function



Logistic Regression

Logistic regression more generally

- Logistic regression in more general case, where $y \in y_1 ... y_K$: learn $K-1$ sets of weights
- for $k < K$:

$$P(y = y_k|\mathbf{x}) = \frac{\exp\left(w_{k0} + \sum_{i=1}^{n} w_{ki} x_i\right)}{1 + \sum_{j=1}^{K-1} \exp\left(w_{j0} + \sum_{i=1}^{n} w_{ji} x_i\right)}$$

- for $k = K$:

$$P(y = y_k|\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp\left(w_{j0} + \sum_{i=1}^{n} w_{ji} x_i\right)}$$

Logistic Regression

Training Logistic Regression

- Call $p(\mathcal{C}_1|\mathbf{x}_i) = \mu_i = \frac{1}{1+\exp(-\mathbf{w}^t\mathbf{x}_i)}$
- Call $p(\mathcal{C}_0|\mathbf{x}_i) = 1 - \mu_i = \frac{\exp(-\mathbf{w}^t\mathbf{x}_i)}{1+\exp(-\mathbf{w}^t\mathbf{x}_i)} = \frac{1}{1+\exp(+\mathbf{w}^t\mathbf{x}_i)}$
- $y_i \in \{0, +1\}$ the observed label
- Goal Function: negative log-likelihood (NLL)
- $\text{NLL}(\mathbf{w}) = -\sum_{i=1}^{N} \log[\mu_i^{y_i} \times (1-\mu_i)^{(1-y_i)}] =$
  $-\sum_{i=1}^{N}[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)]$
- Optimal model:
  $\arg\min_{\mathbf{w}} -\sum_{i=1}^{N}[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)]$

Logistic Regression

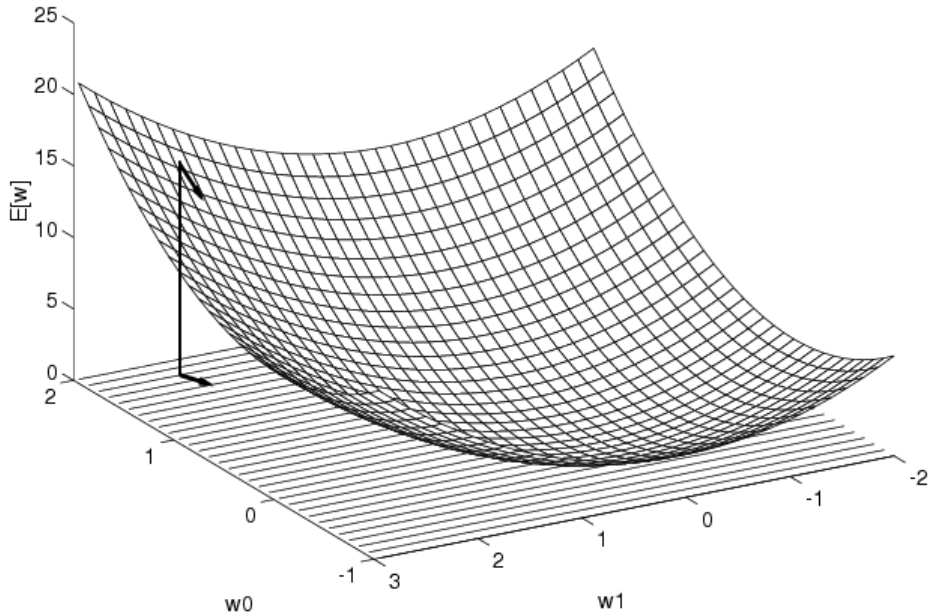
Training Logistic Regression

Unlike linear regression, we can no longer write down the MLE in closed form. Instead, we need to use an optimization algorithm to compute it. For this, we need to derive the gradient and Hessian.

- Gradient $= \mathsf{g} = \sum_i (\mu_i - y_i)\mathbf{x}_i = X^t(\mu - \mathbf{y})$
- Hessian $= \mathsf{H} = X^t S X$, where $S = diag(\mu_i(1 - \mu_i))$

Logistic Regression

Gradient Descent



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Logistic Regression

Newton Method for logistic Regression

One can derive faster optimization methods by taking the curvature of the space (i.e., the Hessian) into account. These are called second order optimization methods. The primary example is Newton's algorithm.

$$
\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k - \mathbf{H}^{-1}\mathbf{g}_k \\
&= \mathbf{w}_k + (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k) \\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\left[(\mathbf{X}^T\mathbf{S}_k\mathbf{X})\mathbf{w}_k + \mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k)\right] \\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T\left[\mathbf{S}_k\mathbf{X}\mathbf{w}_k + \mathbf{y} - \boldsymbol{\mu}_k\right]
\end{aligned}
$$

Logistic Regression

That's all for M(C)LE. How about MAP?

- One common approach is to define priors on W
  - Normal distribution, zero mean, identity covariance
- Helps avoid very large weights and overfitting
- MAP estimate

$$W \leftarrow \arg\max_W \ln P(W) \prod_l P(Y^l | \mathbf{x}^l, W)$$

- let's assume Gaussian prior: $W \sim N(0, \sigma)$

Logistic Regression

MLE vs MAP

Maximum conditional likelihood estimate

$$W \leftarrow \arg\max_W \ln \prod_l P(Y^l|X^l, W)$$

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

Maximum a posteriori estimate with prior W~N(0,σI)

$$W \leftarrow \arg\max_W \ln[P(W) \prod_l P(Y^l|X^l, W)]$$

$$w_i \leftarrow w_i - \eta\lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

Logistic Regression

MAP estimates and Regularization

$$W \leftarrow \arg\max_W \ln[P(W) \prod_l P(Y^l|X^l, W)]$$

$$w_i \leftarrow w_i - \eta\lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

called a "regularization" term

- helps reduce overfitting, especially when training data is sparse
- keep weights nearer to zero (if P(W) is zero mean Gaussian prior), or whatever the prior suggests
- used very frequently in Logistic Regression

Logistic Regression

The Bottom Line

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, < X_1 … X_n >
  - Y is boolean
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli (π)

- Then P(Y|X) is of this form, and we can directly estimate W

$$P(Y = 1|X =< X_1, ...X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

- Furthermore, same holds if the X_i are boolean
  - trying proving that to yourself

Logistic Regression

Generative vs. Discriminative Classifiers

- Training classifiers involves estimating $f : \mathbf{x} \rightarrow y$, or $P(y|\mathbf{x})$
- Generative classifiers (e.g., Naive Bayes)
  - Assume some functional form for $P(\mathbf{x}|y)$, $P(\mathbf{x})$
  - Estimate parameters of $P(\mathbf{x}|y)$, $P(\mathbf{x})$ directly from training data
  - Use Bayes rule to calculate $P(y|x = x_i)$
- Discriminative classifiers (e.g., Logistic regression)
  - Assume some functional form for $P(y|\mathbf{x})$
  - Estimate parameters of $P(y|\mathbf{x})$ directly from training data

Taxonomy of classification methods

- generative models:
  - first determine the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$
  - infer the prior class probabilities $p(\mathcal{C}_k)$
  - Use Bayes' theorem in the form

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

    to find the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$
  - use these posterior probabilities to make class assignments, assigning each new $\mathbf{x}$ to one of the classes
- discriminative models:
  - infer directly $p(\mathcal{C}_k|\mathbf{x})$
  - use these posterior probabilities to make class assignments, assigning each new $\mathbf{x}$ to one of the classes
- models that find directly the discriminant function, mapping each input $\mathbf{x}$ directly onto a class label.

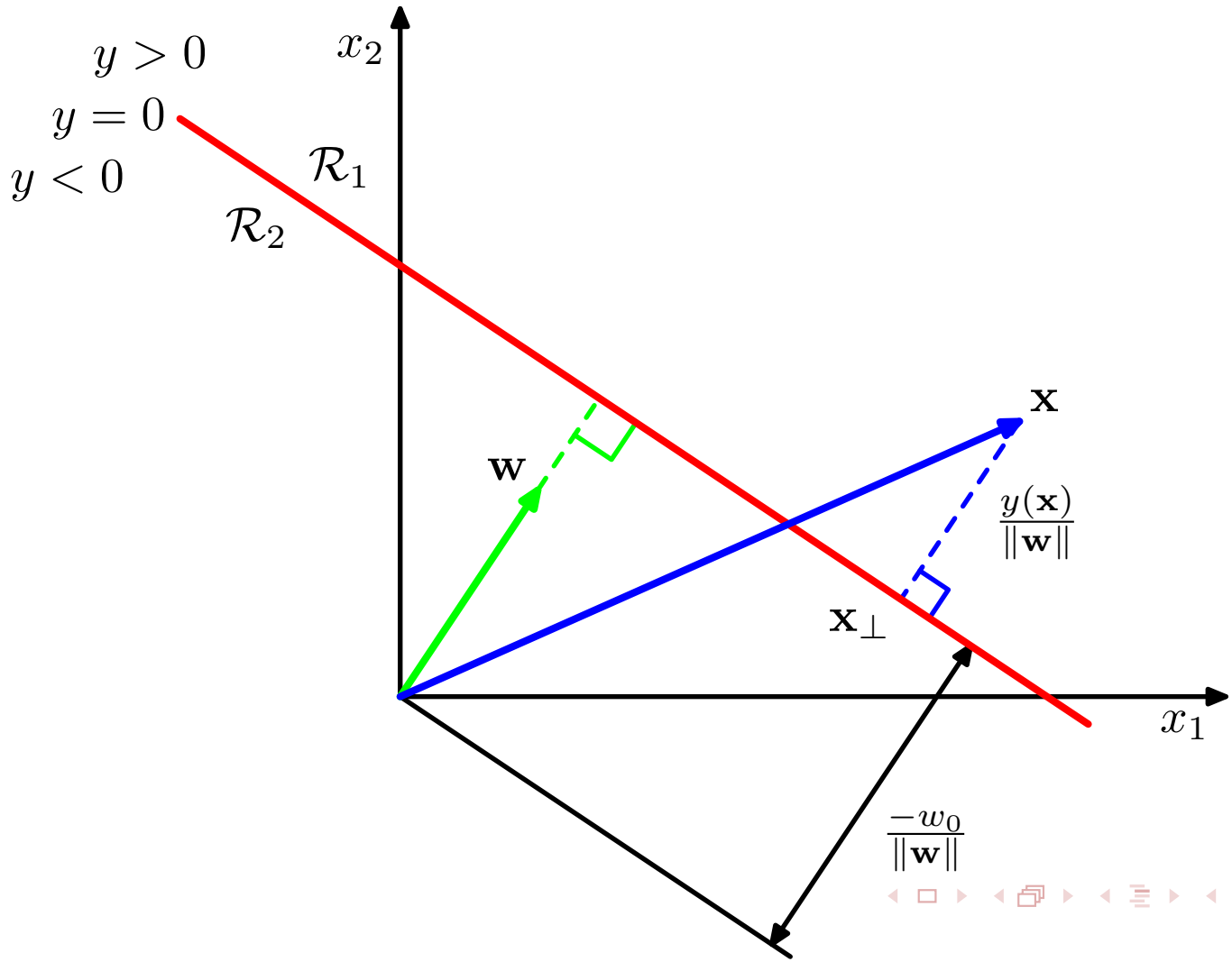
Linear Models for Classification

- Classification:
  - Given a dataset D of pairs $\{\mathbf{x}_i, y_i\}$ where $\mathbf{x}_i$ are coordinates in our feature space and y belong to a discrete set, find the best predictor for the given data.
  - Binary classification is the case when $y \in \{-1, 1\}$ or $\{0, 1\}$.
  - Classification is intrinsically non-linear

Linear Models for Classification

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

$$\begin{cases} \mathcal{C}_1 & y(\mathbf{x}) \geq 0 \\ \mathcal{C}_2 & y(\mathbf{x}) < 0 \end{cases}$$

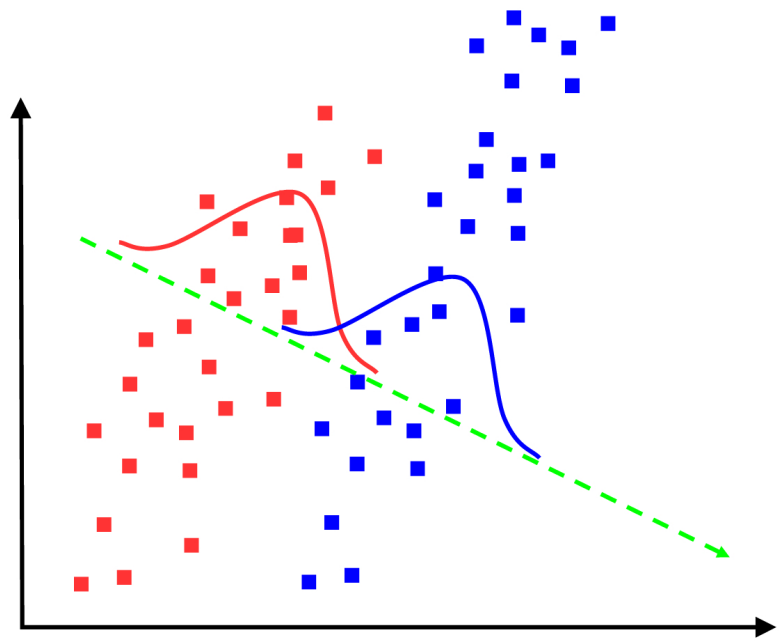


Linear Models for Classification

Fisher's linear discriminant

Also known as Linear Discriminant Analysis

- One way to view a linear classification model is in terms of dimensionality reduction.
- Projection that best separates the data in a least-squares sense.
- Projection of D-dimensional data onto a line.



Linear Models for Classification

Fisher's linear discriminant

Also known as Linear Discriminant Analysis

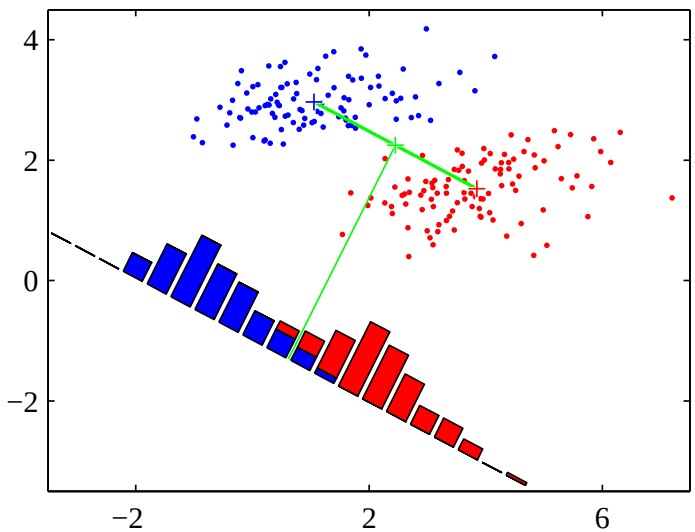
- A simple linear discriminant function is a projection of the data down to 1-D.
  - So choose the projection that gives the best separation of the classes. What do we mean by "best separation"?
- An obvious direction to choose is the direction of the line joining the class means.
  - But if the main direction of variance in each class is not orthogonal to this line, this will not give good separation (see the next figure).
- LDA chooses the direction that maximizes the ratio of between class variance to within class variance.
  - This is the direction in which the projected points contain the most information about class membership (under Gaussian assumptions)

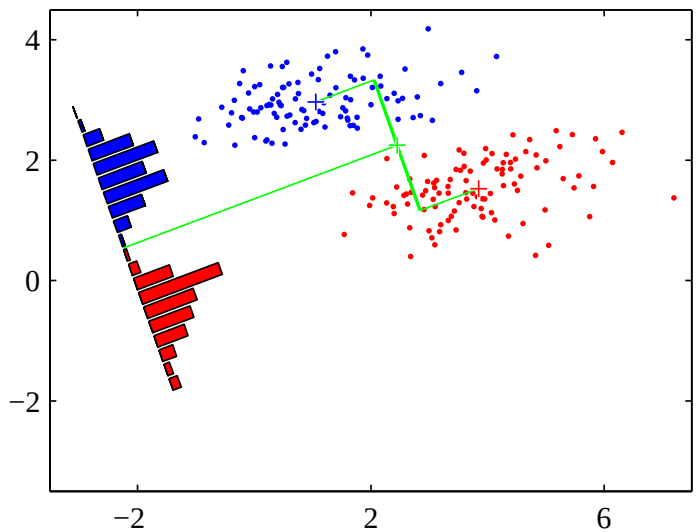
Linear Models for Classification

Fisher's linear discriminant

Also known as Linear Discriminant Analysis

- When projected onto the line joining the class means, the classes are not well separated.
- Fisher chooses a direction that makes the projected classes much tighter, even though their projected means are less far apart.





Linear Models for Classification

Fisher's linear discriminant

Math of Fisher's linear discriminant

- What linear transformation is best for discrimination?
  $y = \mathbf{w}^T \mathbf{x}$
- The projection onto the vector separating the class means seems sensible: $\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$
  with $\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n$ $\mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$
  This $\mathbf{w}$ maximizes $m_2 - m_1 = \mathbf{w}^t(\mathbf{m}_2 - \mathbf{m}_1)$, subject to $||\mathbf{w}|| = 1$
- But we also want small variance within each class:
  $s_k^2 = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} (\mathbf{w}^t \mathbf{x}_n - m_k)^2$
- Fisher's objective function is

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

Linear Models for Classification

Fisher's linear discriminant

More Math of Fisher's linear discriminant

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

$$Optimal\ \ solution: \quad \mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

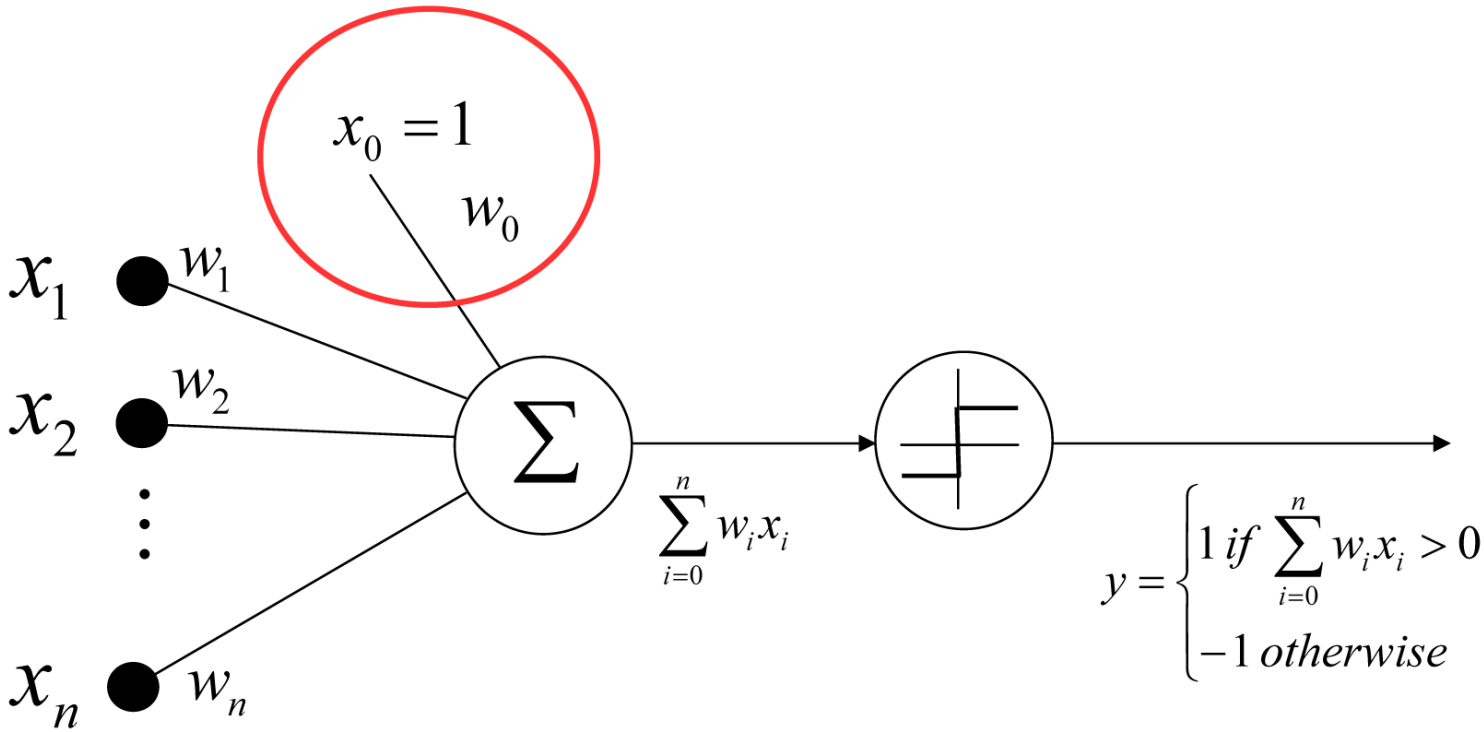
Perceptron

"Perceptrons" describes a whole family of learning machines, but the standard type consisted of a layer of fixed non-linear basis functions followed by a simple linear discriminant function.

- They were introduced in the late 1950's and they had a simple online learning procedure
- Grand claims were made about their abilities. This led to lots of controversy.
- Researchers in symbolic AI emphasized their limitations (as part of an ideological campaign against real numbers, probabilities, and learning)

$$y(\mathbf{x}, \mathbf{w}) = \left(\sum_{j=1}^{M} w_j \phi(\mathbf{x}) \right)$$

Perceptron

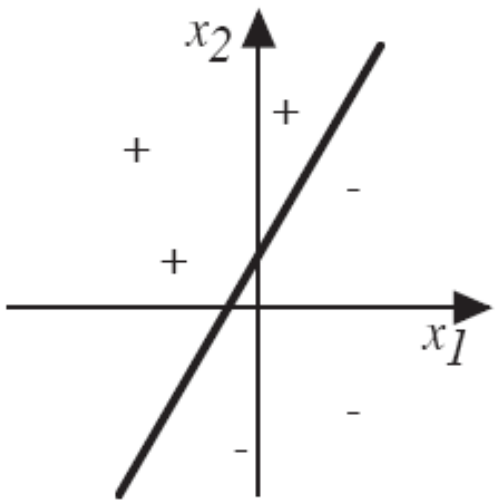


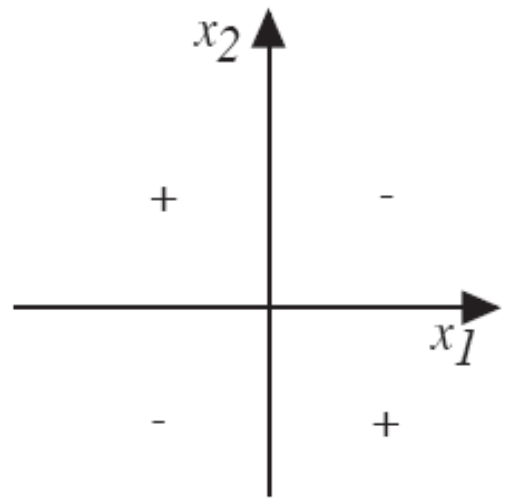
$$y(\mathbf{x}, \mathbf{w}) = sgn(\mathbf{w}^T \mathbf{x})$$

Perceptron

Decision surface

- The decision surface for a single perceptron is a line
- Can represent some functions (AND($\mathbf{x}_1, \mathbf{x}_2$) for example).
- Can only work on linearly separable problems (fails on the XOR).
- For non-linearly separable problems perceptrons must be organized in networks.





Perceptron

Error function

- One possible error measure is the number of misclassified examples.
- Suppose $y_n = \{-1, 1\}$ and the estimated classifier $y(\mathbf{x}_n)$ also outputs a result which is either $+1$ or -1.
- An example $< \mathbf{x}_n, y_n >$ is misclassified if $y_n * y(\mathbf{x}_n)$ is negative.
- So a reasonable error function is just counting the number of examples correctly classified:

$$E(\mathbf{w}) = - \sum_{i \in \text{misclassified}}^{N} y_i y(\mathbf{x}_i)$$

- this is called the $0-1$ loss

How can we find the best weights $\mathbf{w}$?

Perceptron

Perceptron error minimization

Perceptron rule:

- Start with random weights for the perceptron.
- Apply the perceptron to each training example, modify the weights whenever the perceptron misclassifies an example.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Delta\mathbf{w}_t$$

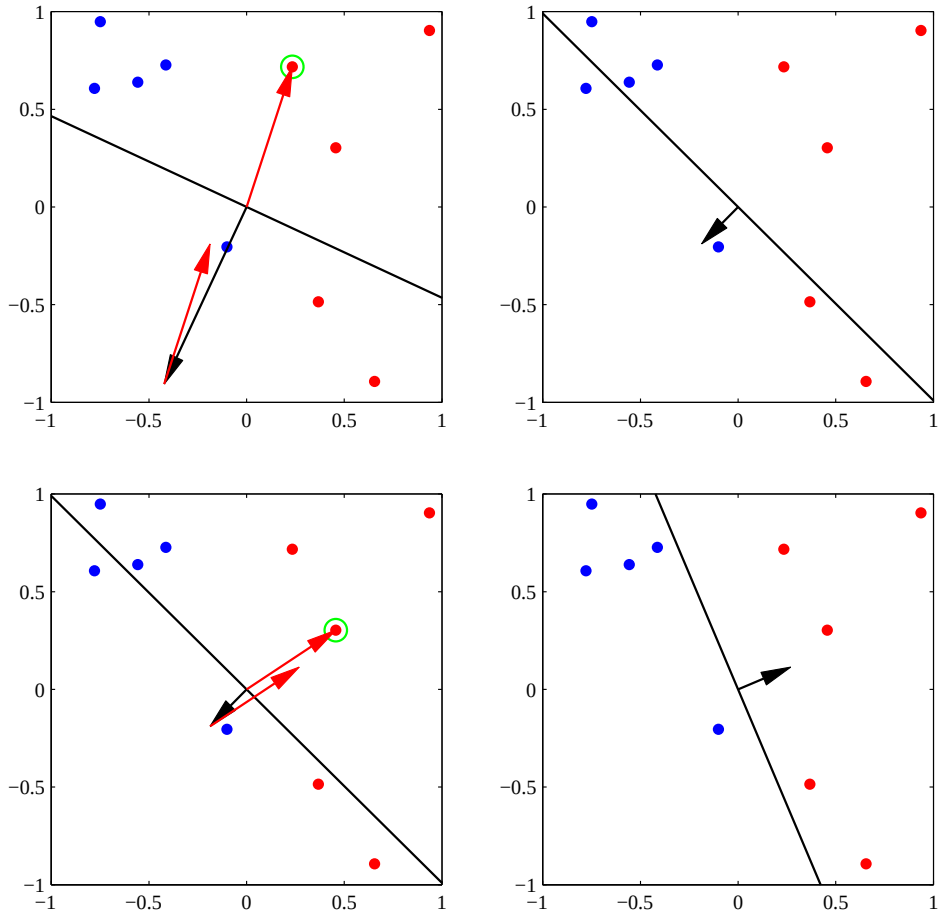
$$\Delta\mathbf{w}_t = \eta(y_n - y(\mathbf{x}_n))\mathbf{x}_n$$

- Process is repeated until no example is incorrectly classified.
- η is the learning rate, which moderates the variation of the weights at each step (can also decay).
- Only converges for linearly separable datasets

Perceptron

Perceptron error minimization

Perceptron rule:



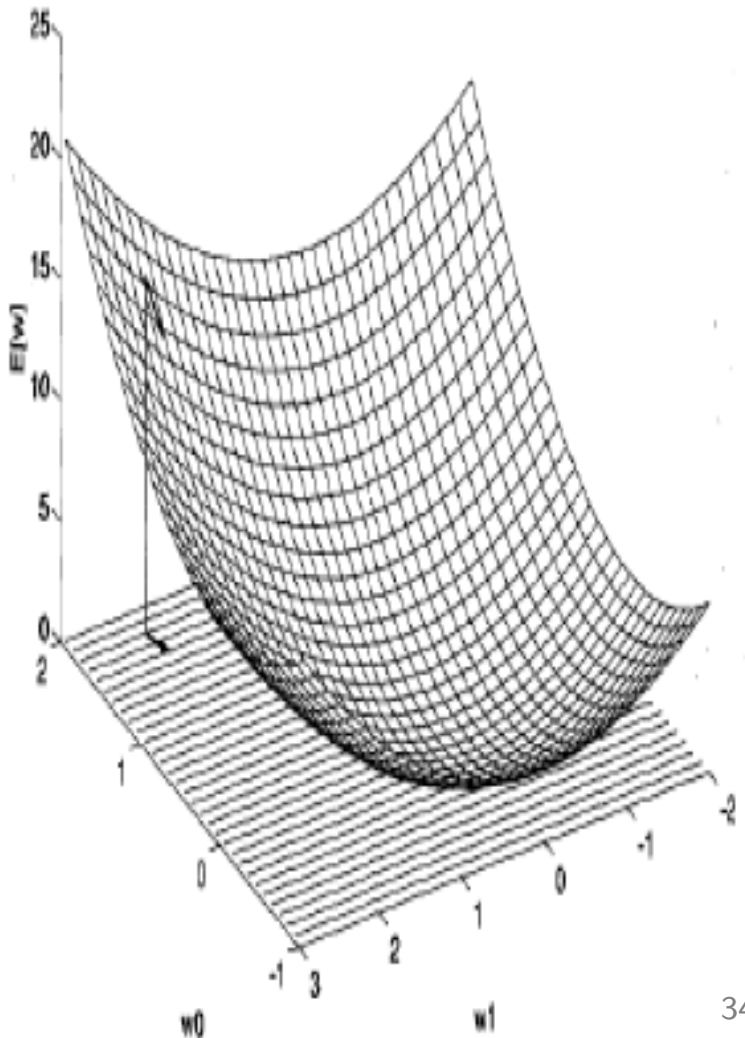
The top left plot shows the initial parameter vector $\mathbf{w}$ shown as a black arrow. The data point circled in green is misclassified and so its feature vector is added to the current weight vector.

Perceptron

Perceptron error minimization with the Gradient Descent algorithm

The sgn function is replaced by a continuous and diferentiable function: $y(\mathbf{x}_n) = \phi(\mathbf{x}_n)$

- Determinates a weight vector that minimizes the error by starting with random weight and modifying it in small incremental steps along the direction that produces the steepest decent on the error surface.
- Given a small enough step this minimization continues until the global minimum is reach.



Perceptron

Perceptron error minimization with the Gradient Descent algorithm

The sgn function is replaced by a continuous and diferentiable function: $y(\mathbf{w}, \mathbf{x}_n) = \phi(\mathbf{w}^t \mathbf{x}_n)$

- $$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E(\mathbf{w}_t)$$
$$\nabla E(\mathbf{w}_t) = \left[\frac{\partial E}{\partial w_1} \dots \frac{\partial E}{\partial w_d} \right]$$
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Delta \mathbf{w}_t$$
$$\Delta \mathbf{w}_t = -\eta \nabla E(\mathbf{w}_t)$$
- η is the learning rate, which moderates the variation of the weights at each step (can also decay).

Perceptron

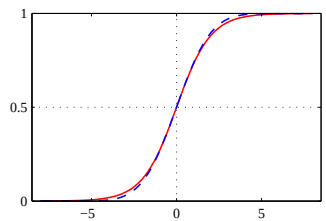
Typical $\phi(z)$ functions

- logistic sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

with $z = \mathbf{w}^t\mathbf{x} + w_0$

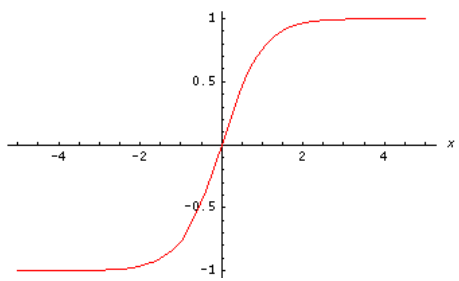
- $\frac{\partial \phi}{\partial z} = \phi(1 - \phi)$



- hyperbolic tangent

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- $\frac{\partial \phi}{\partial z} = 1 - \phi^2$



Perceptron

Gradient

- define $z_n = \mathbf{w}^t \mathbf{x}_n$ (assume that $\mathbf{x}$ was extended with an extra coordinate x_0 as pictured in slide 21)
- predicted output for $\mathbf{x}_n$: $\phi(z_n) = \phi(\mathbf{w}^t \mathbf{x}_n)$
- online update
  - $E(\mathbf{w}) = \frac{1}{2}(\phi(z_n) - y_n)^2$
  - $\frac{\partial E}{\partial \mathbf{w}} = (\phi(z_n) - y_n)\frac{\partial \phi}{\partial z_n}\frac{\partial z_n}{\partial \mathbf{w}} = (\phi(z_n) - y_n)\frac{\partial \phi}{\partial z_n}\mathbf{x}_n$
    - for the logistic sigmoide function $\frac{\partial \phi}{\partial z_n} = \phi \times (1 - \phi)$
    - for the hyperbolic tangent function $\frac{\partial \phi}{\partial z_n} = 1 - \phi^2$
- batch update
  - $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\phi(z_n) - y_n)^2$, with $z_n = \mathbf{w}^t \mathbf{x}_n$
  - $\frac{\partial E}{\partial \mathbf{w}} = \sum_{n=1}^{N}\left\{(\phi(z_n) - y_n)\frac{\partial \phi}{\partial z_n}\frac{\partial z_n}{\partial \mathbf{w}}\right\} =$ $\sum_{n=1}^{N}\left\{(\phi(z_n) - y_n)\frac{\partial \phi}{\partial z_n}\mathbf{x}_n\right\}$

References

Bishop

Pattern recognition and machine learning,

Book.

Eric Xing' Homepage

`http://www.cs.cmu.edu/~epxing/`

`http://www.cs.cmu.edu/~epxing/Class/10701-08s/`

Tom M. Mitchell

Machine learning

McGraw-Hill, 1997.

Sergios Theodoridis and Konstantinos Koutroumbas

Pattern recognition

Elsevier (book)

Linear Models for Classification

Pedro Quelhas,

powerpoint presentation, 2008.