Data Culture: Intro into Machine Learning

Linear classification. ERM. Figures of merits

Alexey Artemov[1,2]

[1] Yandex LLC [2] National Research University Higher School of Economics

22 September 2017

Lecture overview

- The classification task
- Linear models for classification
- The perceptron
- Empirical risk minimization
- Logistic regression
- Figures of merits
- (maybe) Linear discriminant analysis

This lecture is largely based on material from

- C. M. Bishop. Pattern recognition and machine learning. **Chapter 4. Linear models for classification.** *(practical)*
- S. Shalev-Shwartz, S. Ben-David. Understanding machine learning: From theory to algorithms. **Chapter 2. A gentle start.** *(theoretical)*

The binary classification task

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- The classification problem: choose a plausible hypothesis (classifier) $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of the classifier h is the probability (over D) that it will fail

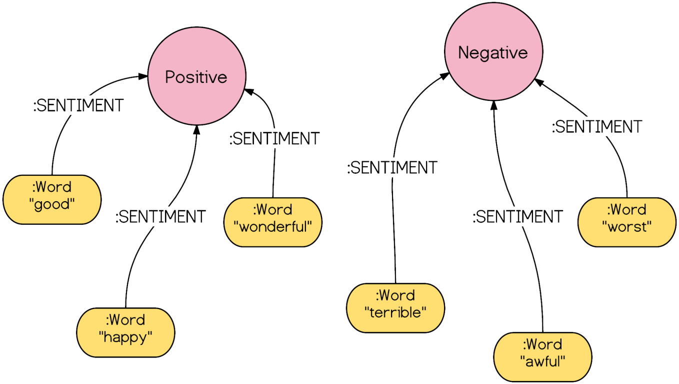
$$Q(h, D) = \mathrm{Pr}_{\mathbf{x} \sim D}[f(\mathbf{x}) \neq h(\mathbf{x})]$$

usually *estimated by* the accuracy metric

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$$

Examples of real-world binary classification tasks

- Sentiment analysis
  "i believe that these are the best looking longest wearing pants you will ever find"

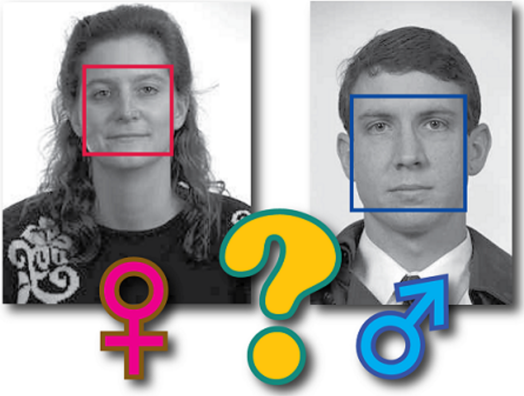


Picture credit:

http://kvangundy.com/wp/sentiment-analysis-amazon-reviews-using-neo4j/

Examples of real-world binary classification tasks

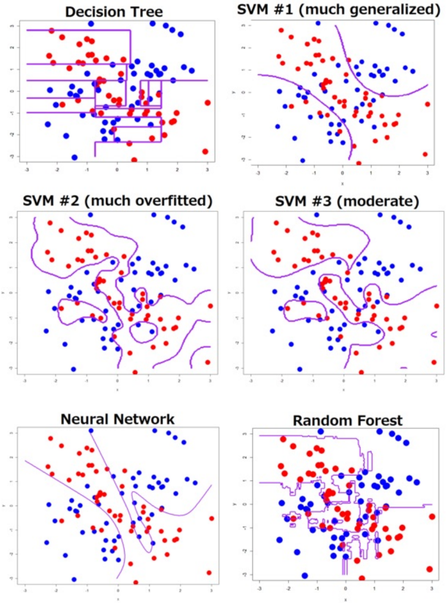
- Gender classification using face images



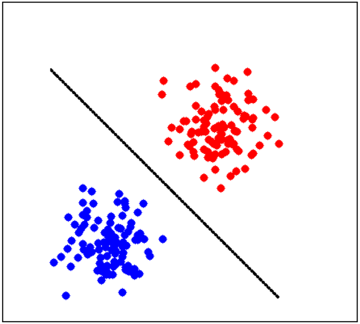
Examples of real-world binary classification tasks

- Road segmentation for self-driving cars

The classifier: decision boundaries



Linear classification model:



Linear models for classification

- Have features $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$
- Linear model: $h(\mathbf{x}) = \mathsf{sign}\big(\sum_{i=1}^{d} w_i x_i + w_0\big) = \mathsf{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- Weight feature activations to get a single scalar quantity
- If quantity is above some threshold, decide that the input vector is a positive example is the input class
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\mathsf{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

The history of perceptrons

- Popularized by Frank Rosenblatt in 1960's
  - Have a very powerful learning algorithm
  - Lots of grand claims were made for what they could learn to do
- In 1969, Minsky and Papert published a book called "Perceptrons" that analyzed what perceptrons could do and showed their limitations
  - Many people thought that these limitations applied to all neural network models
- The Perceptron algorithm is still used today

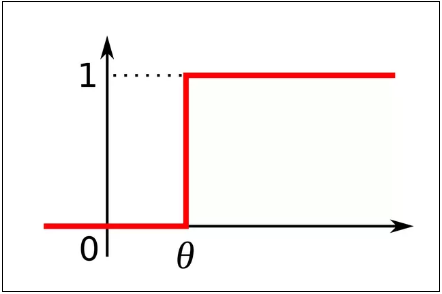
The Perceptron algorithm: inference

- Compute activation:

$$z = \sum_{i=1}^{d} w_i x_i + w_0$$

- Compute answer:

$$y = \begin{cases} 1, & \text{if } z \geqslant 0, \\ 0, & \text{otherwise.} \end{cases}$$



or (as before)

$$y = \text{sign}(z) = \begin{cases} +1, & \text{if } z \geqslant 0, \\ -1, & \text{otherwise.} \end{cases}$$

The Perceptron algorithm: learning

- Add an extra component with value 1 to each input vector
- Pick training instances using any policy that will ensure every training case will get picked
  - If the output unit is correct, **leave** its weights alone.
  - If the output unit incorrectly outputs a zero, **add** the input vector to the weight vector.
  - If the output unit incorrectly outputs a one, **subtract** the input vector to the weight vector.
- This is guaranteed to find a set of weights $\mathbf{w}$ that gets the right answer for all the training cases if such set exists.
- But does such set exist?..

The Perceptron algorithm: formalism

- **Initialise** weights randomly
- **Iterate** with updates

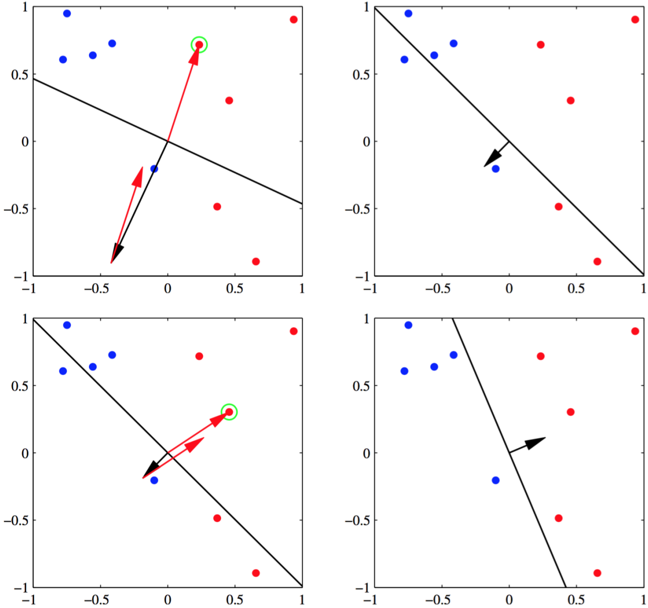
$$\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} + y_{(i)}\mathbf{x}_{(i)}[\mathsf{sign}(\mathbf{w}^{(i)\intercal}\mathbf{x}_{(i)}) \neq y_{(i)}]$$

  where $(\mathbf{x}_{(i)}, y_{(i)})$ is the example selected at iteration i
- **Stop** when all examples are correctly classified

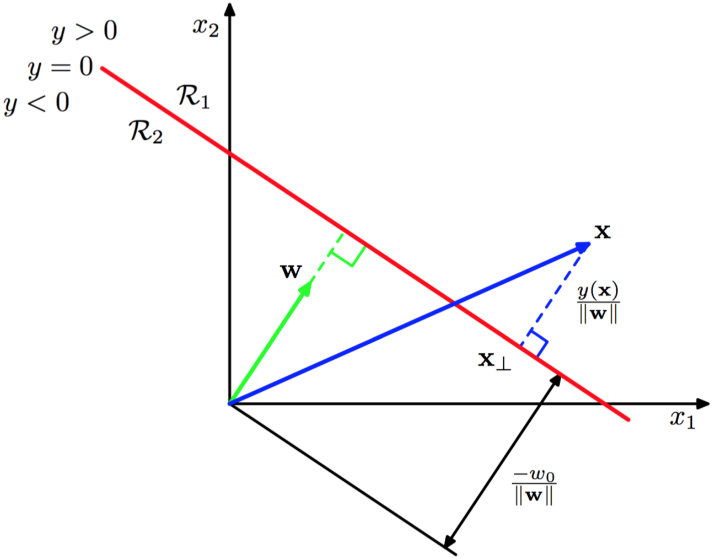
- Introduce the perceptron criterion $E_{\mathsf{P}}(\mathbf{w}) = -\sum_k \mathbf{w}^\intercal \mathbf{x}_k y_k$
- Iterations become (pick $(\mathbf{x}_{(i)}, y_{(i)})$ via SGD)

$$\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \eta \nabla_{\mathbf{w}} E_{\mathsf{P}}(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^{(i)}} =$$
$$= \mathbf{w}^{(i)} + y_{(i)}\mathbf{x}_{(i)}$$

The Perceptron algorithm: geometrical motivation



Linear classifier: geometrical interpretation



Empirical Risk Minimization framework

- Given a labeled sample $X^\ell = \big\{(\mathbf{x}_i, y_i)\big\}_{i=1}^{\ell}$ and some candidate h
  - Suppose $\mathbf{x}_i \in \mathbb{R}^n \equiv \mathbb{X}, \quad y_i \in \{-1, +1\} \equiv \mathbb{Y}$
- Define the empirical error (aka the **empirical risk**) of h as

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$$

  (the proportion of sample points on which h errs)
- $\mathrm{ERM}(X^\ell)$ — find h the minimizes $Q(h, X^\ell)$
- **No learning is possible without applying prior knowledge**
- A hypothesis class $\mathbb{H}$ is a set of hypotheses
- Re-define the ERM rule by searching only inside such a class $\mathbb{H}$
- $\mathrm{ERM}_{\mathbb{H}}(X^\ell)$ picks a classifier $h \in \mathbb{H}$ that minimizes the empirical error over members of $\mathbb{H}$

Guarantees for ERM

Theorem 1 (Guaranteed success for $\mathrm{ERM}_{\mathbb{H}}$)

- *Let $\mathbb{H}$ be a finite class.*
- *Let the unknown labeling rule, f, be a member of $\mathbb{H}$.*
- *Then*
  - *for every $\varepsilon, \delta > 0$, if $m > (\log(|\mathbb{H}|) + \log(1/\delta))/\varepsilon$,*
  - *with probability $> 1 - \delta$ (over the choice of X^ℓ),*
  - *any $\mathrm{ERM}_{\mathbb{H}}$ hypothesis has error below ε.*

Linear models for classification

- Linear model: $h(\mathbf{x}) = \text{sign}\big(\sum_{i=1}^{d} w_i x_i + w_0\big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq f(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

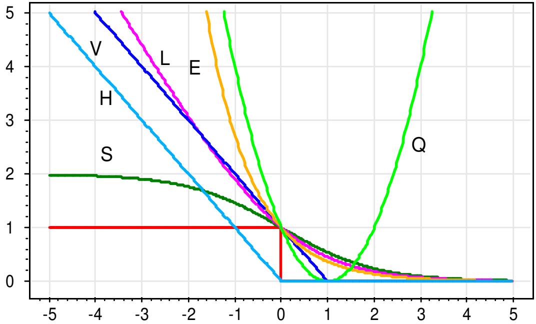
  (**cannot optimize using gradient descent, as the gradient is zero almost everywhere!**)
- **The solution:** optimize a differentiable *upper bound* for $Q(h, X^\ell)$!
- $Q(h, X^\ell)$ can be written using $Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(M_i)$ where $L(M_i) = [M_i < 0] \equiv [y_i \mathbf{w}^\intercal \mathbf{x}_i < 0]$
- Upper-bounding $L(M)$ yields upper bounds for $Q(h, X^\ell)$

Linear models for classification: upper bounds

Multiple approximations to accuracy loss

- $L_{\mathsf{L}}(M) = \log(1 + e^{-M})$
- $L_{\mathsf{H}}(M) = \max(0, 1 - M)$
- $L_{\mathsf{P}}(M) = \max(0, -M)$
- $L_{\mathsf{E}}(M) = e^{-M}$
- $L_{\mathsf{S}}(M) = 2/(1 + e^{M})$

give rise to various learning algorithms



The logistic regression model

- Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$
- A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]}(1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

- Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]}(1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

is often written via log-likelihood (of which the negative is log-loss or cross-entropy)

$$\log L(X^\ell) = \sum_{i=1}^{\ell} [y_i = +1] \log h(\mathbf{x}_i) + [y_i = -1] \log(1 - h(\mathbf{x}_i))$$

The logistic regression model

- The choice of h: sigmoid function

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

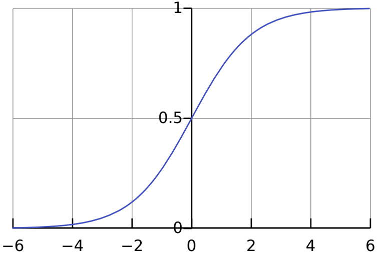
where $\sigma(x) \in [0, 1]$

- Typical choice: the logistic function

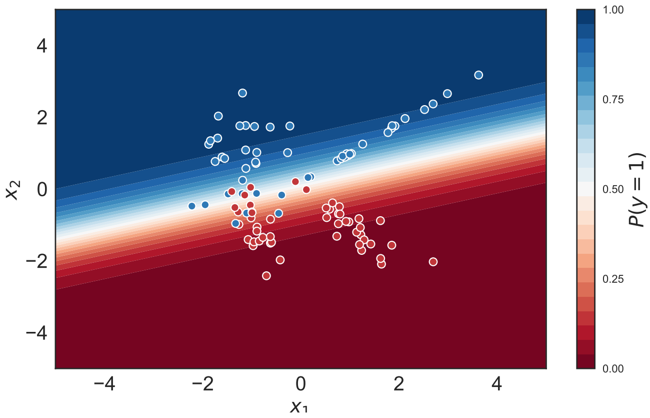
$$\sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\intercal \mathbf{x})}$$

- Plugging the logistic function into the loss yields

$$\sum_{i=1}^{\ell} (1 + \exp(\mathbf{w}^\intercal \mathbf{x})) \to \min_{\mathbf{w} \in \mathbb{R}^d}$$



The logistic regression model



Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \big\{(\mathbf{x}_i, y_i)\big\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?
- Let $a(x) = [h(x) > t]$
- Obvious choice: accuracy

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(\mathbf{x}_i) = y_i]$$

- Bad for imbalanced data: for $\ell = 1000$, $n_- = \sum_{i=1}^{\ell} [y = -1] = 50$, $n_+ = \sum_{i=1}^{\ell} [y = +1] = 950$, a trivial rule $h(\mathbf{x}) = +1$ would yield $\text{accuracy}(a, X^\ell) = 0.95$

Classification quality: confusion matrix

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False negative (FN)	True Negative (TN)

- More informative criteria:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

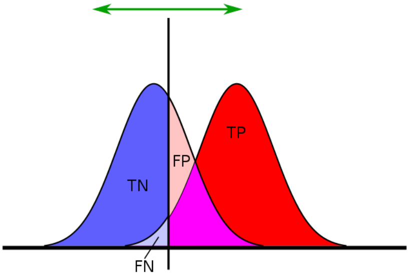
Classification quality: operating curves

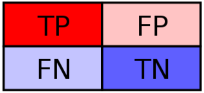
- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates

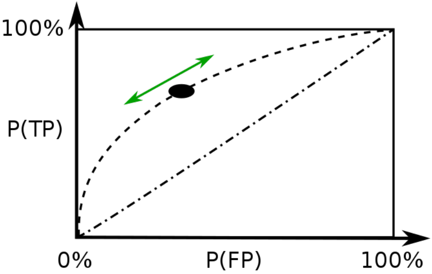
$$\mathsf{FPR} = \frac{\mathsf{FP}}{\mathsf{FP} + \mathsf{TN}}, \qquad \mathsf{TPR} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}},$$

  corresponding to various choices of the threshold t
- The plot TPR(FPR) is called the receiver operating characteristic (or ROC) curve
- Area under curve (ROC-AUC) reflects classification quality

Receiver operating characteristic curve







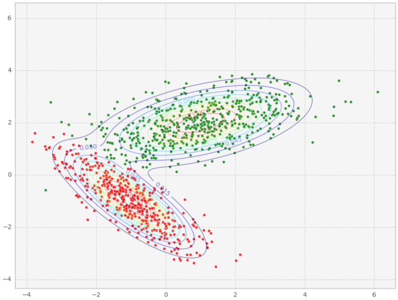
Linear discriminant analysis [R. Fisher, 1936]

- Training set
  $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ where
  $y_i \in \{-1, +1\}$
- The model:

$$\mathrm{P}(\mathbf{x}_i | y_i = -1) = \mathcal{N}(\boldsymbol{\mu}_-, \boldsymbol{\Sigma}_-),$$
$$\mathrm{P}(\mathbf{x}_i | y_i = +1) = \mathcal{N}(\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$$

- Minimize error probability via Neyman-Pearson lemma

$$\frac{\mathrm{P}(\mathbf{x} | y_i = +1)}{\mathrm{P}(\mathbf{x} | y_i = -1)} > t$$



Linear discriminant analysis [R. Fisher, 1936]

- Log-likelihood ratio

$$\log \mathrm{P}(\mathbf{x}|y_i = +1) - \log \mathrm{P}(\mathbf{x}|y_i = -1) > T$$

- Recall that
  $\mathrm{P}(\mathbf{x}|y_i = \pm 1) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}_\pm|}} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\pm)^\intercal \boldsymbol{\Sigma}_\pm^{-1}(\mathbf{x} - \boldsymbol{\mu}_\pm)\}$
- When $\boldsymbol{\Sigma}_+ = \boldsymbol{\Sigma}_- = \boldsymbol{\Sigma}$, the optimal solution is

$$\mathbf{x}^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) > \frac{1}{2}(T - (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-))$$

  which has the form $\mathbf{w}^\intercal \mathbf{x} > c$
- Homoscedastic case: linear discriminant hyperplane

Linear discriminant analysis