6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

Topics

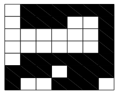
Classification

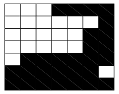
- A regression approach to classification
- Fisher linear discriminant analysis
- Generative probabilistic classifiers
- Generalized linear/additive models
- Perceptron learning rule
- Multi-layer neural networks, back-propagation

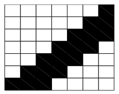
Classification

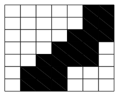
Example: digit recognition (8x8 binary digits)

binary digit | target label


"2"


"2"


"1"


"1"

... ...

(you'll be solving a problem like this in the problem set...)

Classification via regression

- We ignore the fact that the output is binary (0/1) rather than a continuous variable

  Given a linear regression function

  $$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d \quad (1)$$

  we minimize the squared difference between the predicted output (continuous) and the observed label (binary) in the training set:

  $$J_n(\mathbf{w}) = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2/2 \quad (2)$$

- How do we classify any new example $\mathbf{x}$?

Classification via regression cont'd

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d \tag{3}$$

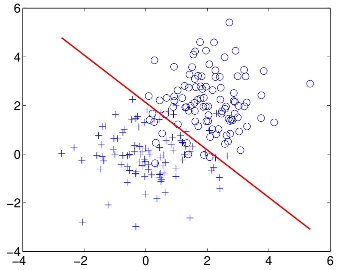
Any new (test) example $\mathbf{x}$ can be classified according to

$$\text{label} = 1 \text{ if } f(\mathbf{x}; \mathbf{w}) > 0.5, \text{ and label} = 0 \text{ otherwise} \tag{4}$$

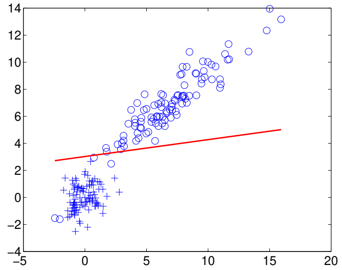
where $f(\mathbf{x}; \mathbf{w}) = 0$ defines the *decision boundary*.

Classification via regression cont'd

- This is not optimal... why not?



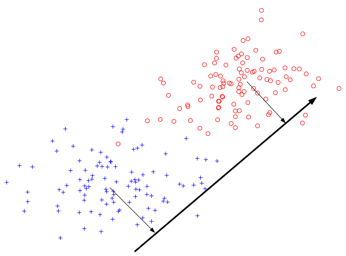
sometimes good



sometimes bad

Fisher linear discriminant analysis

- Assume two populations with means μ_1, μ_0 and covariances Σ_1, Σ_0 (not necessarily normal).



- We try to find a direction $\mathbf{w} = [w_1, \ldots, w_d]^T$ in the input space such that projecting the populations along this dimension makes them "well-separated".

Fisher linear discriminant analysis cont'd

- More mathematically: we optimize a linear projection $\mathbf{w}^T\mathbf{x}$ by maximizing

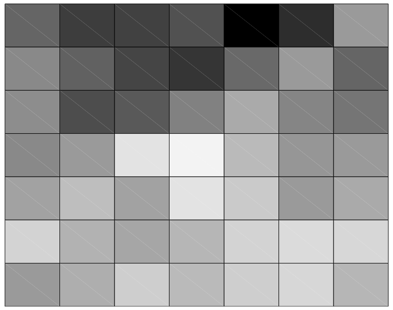
$$J_{Fisher}(\mathbf{w}) = \frac{(\text{Separation of projected means})^2}{\text{Sum of within population variances}} \tag{5}$$

$$= \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_0)^2}{\mathbf{w}^T(\Sigma_1 + \Sigma_0)\mathbf{w}} \tag{6}$$

- The solution $\mathbf{w} = (\Sigma_1 + \Sigma_0)^{-1}(\mu_1 - \mu_0)$ is optimal for two normal class populations with equal covariances ($\Sigma_1 = \Sigma_0$)

Fisher linear discriminant analysis cont'd

- Example: binary digits "1" versus "7"



This is approximately the matrix difference "1" - "7"

Generative approach to classification

- We can model each class conditional population with a multivariate normal (Gaussian) distribution

$$\mathbf{x} \sim N(\mu_1, \Sigma_1), \quad y = 1 \tag{7}$$
$$\mathbf{x} \sim N(\mu_0, \Sigma_0), \quad y = 0 \tag{8}$$

where

$$p(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\{ -\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu) \} \tag{9}$$

- How do we make decisions?

Mixture classifier cont'd

- Examples $\mathbf{x}$ are classified on the basis of which Gaussian explains the data better

$$\log \frac{P(\mathbf{x}|\mu_1, \Sigma_1)}{P(\mathbf{x}|\mu_0, \Sigma_0)} > 0 \quad y = 1 \tag{10}$$
$$\leq 0 \quad y = 0 \tag{11}$$

or when the classes have different a priori probabilities

$$P(y=1|\mathbf{x}) = \frac{P(\mathbf{x}|\mu_1, \Sigma_1)P(y=1)}{P(\mathbf{x}|\mu_1, \Sigma_1)P(y=1) + P(X|\mu_0, \Sigma_0)P(y=0)} \tag{12}$$

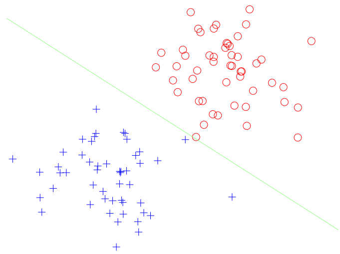
- Neyman – Pearson lemma!
- The decision boundary is defined by

$$\log \frac{P(\mathbf{x}|\mu_1, \Sigma_1)}{P(\mathbf{x}|\mu_0, \Sigma_0)} = 0 \quad \text{or} \quad P(y=1|\mathbf{x}) = 0.5 \tag{13}$$

Mixture classifier: decision rule

- Equal covariances

$$\mathbf{x} \sim N(\mu_1, \Sigma), \quad y = 1 \tag{14}$$
$$\mathbf{x} \sim N(\mu_0, \Sigma), \quad y = 0 \tag{15}$$

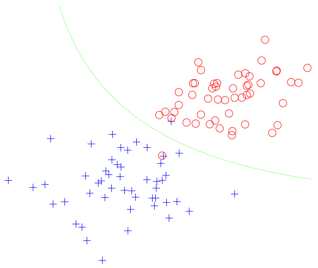


- The decision rule is *linear*

Mixture classifier: decision rule

- Unequal covariances

$$\mathbf{x} \sim N(\mu_1, \Sigma_1), \quad y = 1 \tag{16}$$
$$\mathbf{x} \sim N(\mu_0, \Sigma_0), \quad y = 0 \tag{17}$$



- The decision rule is *quadratic*

Maximum likelihood estimation

- We can estimate the class conditional distributions $p(\mathbf{x}|\mu,\Sigma)$ separately
- A multivariate Gaussian model

$$p(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\{ -\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \} \tag{18}$$

Given a random sample $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$, the maximum likelihood estimates of the parameters are:

1. Sample mean

$$\hat{\mu} = \frac{1}{n}\sum_i \mathbf{x}_i \tag{19}$$

2. Sample covariance

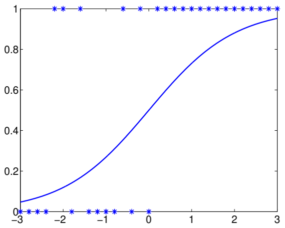
$$\hat{\Sigma} = \frac{1}{n}\sum_i (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T \tag{20}$$

Generalized linear models

- When the two Gaussian distributions have equal covariances, the posterior class probability $P(y=1|\mathbf{x})$ from the mixture model reduces to a *logistic regression model*

$$P(y=1|x,\mathbf{w}) = g\,(\,w_0 + w_1x_1 + \ldots + w_dx_d\,) \tag{21}$$

where the parameters $\mathbf{w}$ are functions of μ_1, μ_0, Σ. Here $g(z) = (1+\exp(-z))^{-1}$ is the logistic function.



- What if the covariances are not equal?

Linear separation and robustness

- Why would we prefer the logistic regression model?

$$P(y=1|x,\mathbf{w}) = g\,(\,w_0 + w_1x_1 + \ldots + w_dx_d\,) \tag{22}$$

- The logistic regression model arises from other generative distributions as well, not just from the mixture of two Gaussians
- We no longer need (as) strong assumptions about the parametric distribution generating the examples; the approach is more *robust*
- Different estimation criteria!

Fitting logistic regression models

- Since the classification model gives a probability distribution over the labels y given the input $\mathbf{x}$ we can fit these models using the maximum likelihood criterion

$$L(D;\mathbf{w}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i,\mathbf{w}) \tag{23}$$

where

$$P(y=1|\mathbf{x},\mathbf{w}) = g\,(\,w_0 + w_1 x_1 + \ldots + w_d x_d\,) \tag{24}$$

- **Note:** this is very different from the generative maximum likelihood fitting in the context of mixture models

  (you have to derive the details in the problem set)

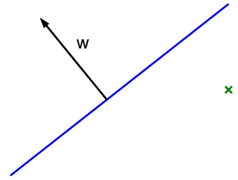
Other ways of fitting linear separators

$f(\mathbf{x};\mathbf{w},w_0) = w_0 + \mathbf{w}^T\mathbf{x}$

- Mistake driven *perceptron* algorithm

$$\text{if } \mathbf{x}_i \text{ missclassified} \quad \begin{array}{lcl} \mathbf{w} & \leftarrow & \mathbf{w} + \epsilon(2y_i - 1)\mathbf{x}_i \\ w_0 & \leftarrow & w_0 + \epsilon(2y_i - 1) \end{array} \tag{25}$$

where ϵ is the *learning rate*.



Additive models and classification

- All the classification methods discussed so far can be used equally well with additive models

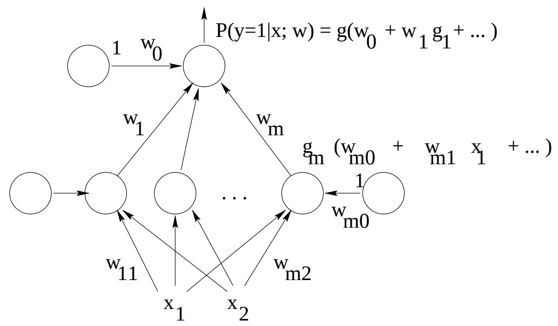
$$f(\mathbf{x};\mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + \ldots w_m\phi_m(\mathbf{x}) \tag{26}$$

  e.g., generalized linear models $\rightarrow$ generalized additive models

- How should we then choose the basis functions $\phi_i(\mathbf{x})$?
- This is a "feature selection" problem
- One approach is to make the features adjustable...

Multi-layer neural networks

- The basis functions themselves are generalized linear models



- Theoretically only two layers are needed
- When $\mathbf{w} \approx 0$, the mapping from inputs to outputs is linear!

Multi-layer neural networks cont'd

- Estimation as before, maximizing the likelihood of the labels given the examples
- Given a training example $(\mathbf{x}, y)$, we can perform gradient ascent in the log-probability:

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \frac{\partial}{\partial \mathbf{w}} \log P(y|\mathbf{x}, \mathbf{w}) \tag{27}$$

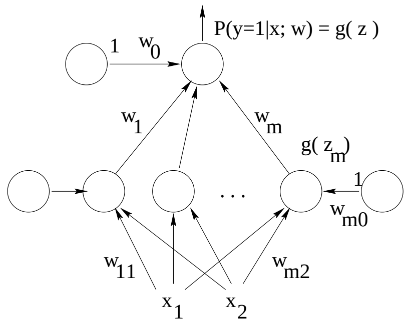
- The gradient can be computed via error back-propagation
- Many local optima

Back-propagation

Let z, $z_i, i = 1, \ldots, m$ be the total "input" to each unit (neuron) in the context of $\mathbf{x}$

$$\begin{aligned} z &= w_0 + w_1 g(z_1) + \ldots + w_m g(z_m) \\ z_i &= w_{i0} + w_{i1} x_1 + w_{i2} x_2, \;\; i = 1, \ldots, m \end{aligned}$$

For example, $\log P(y|\mathbf{x}, \mathbf{w}) = y \log g(z) + (1 - y) \log(1 - g(z))$



Back-propagation cont'd

$$\begin{aligned} \delta &= \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w}) \\ \delta_i &= \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w}) \\ &= \frac{\partial g(z_i)}{\partial z_i} \times \frac{\partial z}{\partial g(z_i)} \times \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w}) \\ &= g'(z_i) \times w_i \times \delta \end{aligned} \tag{28}$$

- And thus

$$\begin{aligned} \frac{\partial}{\partial w_{ij}} \log P(y|\mathbf{x}, \mathbf{w}) &= \frac{\partial z_i}{\partial w_{ij}} \times \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w}) \\ &= x_j \times \delta_i \end{aligned} \tag{29}$$