Statistical Learning VI (3-12-2015)

1. Support Vector Machines
   a) Optimal separating hyperplane
   b) SVMs
   c) Interpretation as penalized ERM with hinge loss
   d) Dual Formulation
   e) Kernel trick
   f) Derivation of dual formulation

2. Decision Trees
   a) General set-up
   b) Growing
   c) Pruning

Vapnik & Chervonenkis: foundational work on statistical learning starting in sixties and seventies, leading to SVMs ~1990.

Vapnik, 1998: "solve the problem directly and never solve a more general problem as an intermediate step"

1. SVMs

a) Optimal separating hyperplane

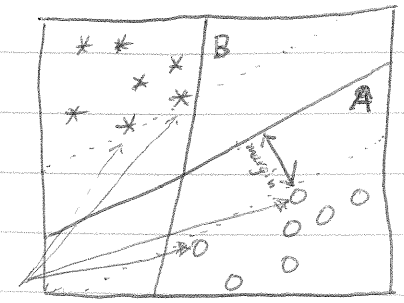
estimate

generative: $P(X, Y)$
discriminative: $P(Y|X)$
now: decision boundary directly

Assume: 2 classes, linearly separable

$\binom{y_1}{x_1}, \ldots, \binom{y_N}{x_N}$ $\quad y \in \{-1, +1\}$

Linear Model: classifiers that compute $x^T\beta + \beta_0$ ← writing intercept separately and return its sign



Which decision boundary do you like best? A or B?

A has larger "margin" = distance to nearest point

Fig. 12.1, left

"Optimal" separating hyperplane maximizes the margin.

support vectors: points on the margin

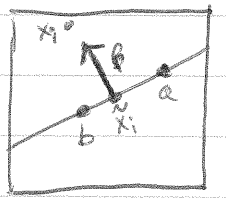
decision boundary: $\{x : x^T\beta + \beta_0 = 0\}$

Let $f(x) = x^T\beta + \beta_0$. Then distance of x_i to decision boundary is $\frac{|f(x_i)|}{\|\beta\|}$

margin: $\frac{y_i \cdot f(x_i)}{\|\beta\|}$ is distance if x_i classified correctly

is negative distance if classified incorrectly

Optional: for any a, b on decision boundary:



$(b-a)^T\beta = 0$, so $\frac{\beta}{\|\beta\|}$ is normal vector to the decision boundary of length 1.

suppose $\tilde{x}_i$ is projection of x_i onto decision boundary. Then

$$x_i = \tilde{x}_i + \alpha \cdot \frac{\beta}{\|\beta\|}$$ $|\alpha|$ is distance of x_i from decision boundary

$$\alpha \frac{\beta}{\|\beta\|} = x_i - \tilde{x}_i$$

$$\alpha \cdot \|\beta\| = (x_i - \tilde{x}_i)^T\beta$$ (because $\beta^T\beta = \|\beta\|^2$)

$$= x_i^T\beta - \tilde{x}_i^T\beta$$

$$= x_i^T\beta + \beta_0$$ (because $\tilde{x}_i$ on decision boundary)

$$= f(x_i)$$

$$\alpha = \frac{f(x_i)}{\|\beta\|}$$

to maximize the margin M:

$$\max_{\beta, \beta_0, M} \quad M$$

$$\text{subject to:} \quad \frac{y_i (x_i^T \beta + \beta_0)}{\|\beta\|} \geq M \quad \text{for } i = 1, \dots, N$$

↑ margin of $\begin{pmatrix} y_i \\ x_i \end{pmatrix}$

Same decision boundary and margin if we multiply β, β_0 by a constant, so can always choose this constant such that $\|\beta\| = \frac{1}{M}$:

$$\max_{\substack{\beta, \beta_0, M \\ \|\beta\| = \frac{1}{M}}} \quad M$$

$$\text{s.t.} \quad \cancel{\frac{y_i (x_i^T \beta + \beta_0)}{\|\beta\|}} \geq M \cdot \|\beta\| \quad \forall i$$

$$\max_{\beta, \beta_0} \quad \frac{1}{\|\beta\|}$$

$$\text{s.t.} \quad y_i (x_i^T \beta + \beta_0) \geq 1 \quad \forall i$$

solution achieved by same parameters β, β_0

---

convex function → $\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2$

linear inequality constraints → $\text{s.t.} \quad y_i (x_i^T \beta + \beta_0) \geq 1 \quad \forall i$

b) SVMs

What if classes are not linearly separable?

Fig. 12.1, right: introduce slack variable $\xi_i \geq 0$ for each data point (Greek letter "xi") $i = 1, \ldots, N$

$$\max_{\beta, \beta_0, M, \xi_i} \quad M$$

subject to: $\xi_i \geq 0$, $\frac{y_i(x_i^T\beta + \beta_0)}{\|\beta\|} \geq M(1-\xi_i)$ $i = 1, \ldots, N$

$$\sum_{i=1}^{N} \xi_i \leq t$$

t: parameter of alg. (Assume we take large enough to have a solution)

support vectors: all points inside the margin

$$\min_{\beta, \beta_0, \xi_i} \quad \tfrac{1}{2}\|\beta\|^2$$

s.t. $\xi_i \geq 0$, $y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$ $i = 1, \ldots, N$

$$\sum_{i=1}^{N} \xi_i \leq t$$

equivalent ↓

$$\min_{\beta, \beta_0, \xi_i} \quad \tfrac{1}{2}\|\beta\|^2 + C \cdot \sum_{i=1}^{N} \xi_i \qquad (*)$$

(C: parameter)

s.t. $\xi_i \geq 0$, $y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$ $i = 1, \ldots, N$

c) Interpretation as penalized ERM

see slides 5

$L(y_i, f(x_i)) = \max\{0, 1 - y_i \cdot f(x_i)\}$ is "hinge loss"

$$\min_{\beta,\beta_0,\xi_i} \quad \tfrac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \xi_i \geq 0 \;,\; \xi_i \geq 1 - y_i(x_i^T\beta+\beta_0) \qquad i = 1,\dots,N$$

$$\xi_i \geq L(y_i, x_i^T\beta+\beta_0)$$

$$\min_{\beta,\beta_0} \quad \sum_{i=1}^{N} L(y_i, x_i^T\beta+\beta_0) + \frac{1}{2C}\|\beta\|^2$$

ERM for hinge loss with L_2-penalty, $\lambda = \frac{1}{2C}$.

d) Dual Formulation

$$\max_{\alpha_1,\dots,\alpha_N} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k \underbrace{\langle x_i, x_k\rangle}_{= x_i^T x_k}$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$

Then solution to (*) is:

$$\hat{\beta} = \sum_{i=1}^{N}\hat{\alpha}_i y_i x_i$$ ← very different, but reminiscent of nearest neighbor, because also defined in terms of training data

$\hat{\alpha}_i = 0$ for x_i outside margin

$0 < \hat{\alpha}_i < C$ for x_i on margin

$\hat{\alpha}_i = C$ for x_i inside margin

solve $\hat{\beta}_0$ from $\hat{\alpha}_i[y_i(x_i^T\hat{\beta}+\hat{\beta}_0) - 1] = 0$

for any i s.t. $0 < \hat{\alpha}_i < C$

e) Kernel trick

Fig. 2.5: how to learn something like this with linear classifier?

map features x to a larger set of features $h(x)$!

e.g.

$$h\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1 \cdot x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$

Then $\langle x_i, x_k \rangle$ in dual formulation becomes $\langle h(x_i), h(x_k) \rangle$

kernel trick: don't need to specify h,
only need to know the kernel function

really nice if this were a simple function..., so turn things around and start by defining $K(x_i, x_k)$

$$K(x_i, x_k) = \langle h(x_i), h(x_k) \rangle$$ ← measure of similarity, larger if x_i, x_k more similar.

$h(x)$ may even be infinite-dimensional!

Examples: $K(x, x') = (1 + \langle x, x' \rangle)^d$: d-th degree polynomial

e.g. $d = 2$, $x \in \mathbb{R}^2$:

$$K(x, x') = (1 + x_1 x_1' + x_2 x_2')^2$$
$$= \langle h(x), h(x') \rangle$$

for

$$h(x) = \begin{pmatrix} 1 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix}$$

radial basis: $K(x, x') = e^{-\gamma \|x - x'\|^2}$

neural network: $K(x, x') = \tanh(a \langle x, x' \rangle + b)$

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

If K satisfies certain technical conditions (symmetric, positive definite) then there always exists some mapping h s.t. $K(x, x') = \langle h(x), h(x') \rangle$.

Classifying a new x:

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0 = h(x)^T \sum_{i=1}^{N} \alpha_i y_i h(x_i) + \hat{\beta}_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \hat{\beta}_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + \hat{\beta}_0$$

Again no need to specify h; only need to know kernel.

Fig. 12.3

f.) Derivation of Dual Formulation ← optional!

$$\min_{\beta, \beta_0, \xi_i} \quad \frac{1}{2}\|\beta\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad \xi_i \geq 0, \quad y_i(x_i^T \beta + \beta_0) \geq (1 - \xi_i) \geq 0 \quad i = 1, \ldots, N$$

$$\min_{\beta, \beta_0, \xi_i} \sup_{\alpha_i, \mu_i \geq 0} \underbrace{\frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i(x_i^T\beta + \beta) - (1 - \xi_i)) - \sum_{i=1}^{N} \mu_i \xi_i}_{A}$$

(If β, β_0, ξ_i violate constraints, then α_i or μ_i becomes ∞ and $A = \infty$, so β, β_0, ξ_i only achieve minimum while satisfying constraints. And then α_i, μ_i become 0, so the constraints drop away and we are minimizing the previous objective.)

~~$\nabla_\beta A = 0 \Rightarrow \beta = \sum_{i=1}^{N} \alpha_i y_i x_i$~~ ~~$\nabla_{\beta_0} A = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$~~

~~$\nabla_{\xi_i} A = 0 \Rightarrow \mu_i = C - \alpha_i$~~

~~plugging~~

$$\min_{\beta,\beta_0,\xi_i} \sup_{\alpha_i,\mu_i \geq 0} A = \sup_{\alpha_i,\mu_i \geq 0} \min_{\beta,\beta_0,\xi_i} A$$

by convex optimization theory (Slater's condition)

↑ can solve this

$$\nabla_\beta A = 0 \Rightarrow \beta = \sum_{i=1}^{N} \alpha_i y_i x_i \qquad \nabla_{\beta_0} A = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\nabla_{\xi_i} A = 0 \Rightarrow \mu_i = C - \alpha_i$$

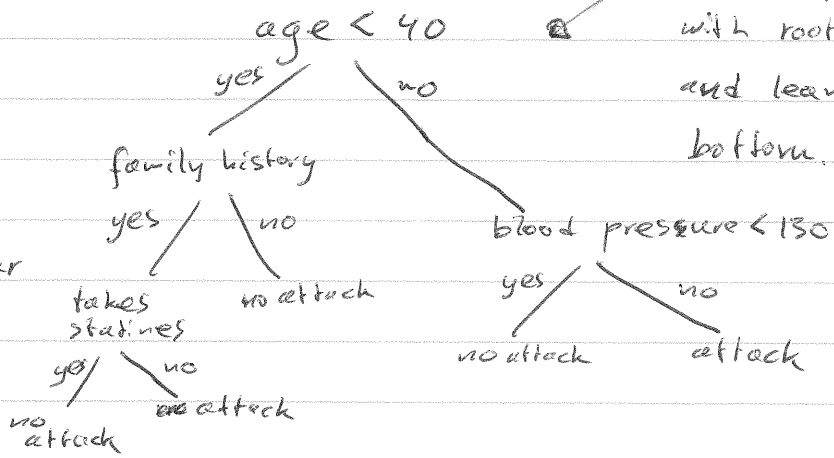
Plugging these in gives dual formulation.

2. Decision Trees

a) General Set-up

mathematicians see trees upside down with root at the top and leaves at the bottom.

decision tree for classification of heart attack next year



Advantages: * easy to interpret
* nice algorithms

Disadvantages: * not competitive with logistic regression/kernel methods in terms of prediction accuracy
(will be fixed next lecture)
* instability / learning algorithms have high variance
⇒ small changes in data lead to very different tree
⇒ undermines interpretation

Fig. 9.2

Estimates class or regression value independently per region R_m with ERM:

$$\hat{c}_m = \underset{c}{\operatorname{argmin}} \sum_{x_i \in R_m} L(y_i, c)$$

regression: $L(y,c) = (y-c)^2$ "squared error" $\Rightarrow$ c_m is avg y_i in R_m

classification: 0/1-loss $\Rightarrow$ c_m is most common class in R_m

or multinomial model with log loss:

(for two classes for simplicity)

* c_m is now probability $P(Y=1 \mid x \in R_m)$

$$\hat{c}_m = \underset{c}{\operatorname{argmin}} \sum_{x_i \in R_m} -\log P(y=y_i \mid x_i \in R_m)$$

$$= \underset{c}{\operatorname{argmin}} -n_1 \log c - n_0 \log(1-c) \qquad (n_k = \#k \text{ in } R_m)$$

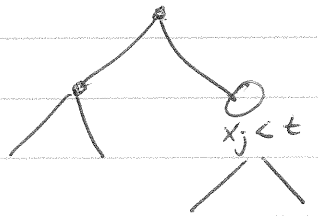
$$= \underset{c}{\operatorname{argmin}} -\hat{p}_{m,1} \log c - \hat{p}_{m,0} \log(1-c)$$

where $\hat{p}_{m,k} = \frac{n_k}{n_0+n_1}$

$\hat{c}_m = \hat{p}_{m,1}$ is ML estimate for Bernoulli model

$-\hat{p}_{m,1} \log \hat{p}_{m,1} - \hat{p}_{m,0} \log \hat{p}_{m,0}$ is "entropy"

b) Growing



How to choose feature x_j + threshold t?

Greedy approach: split region R_m into R_{m1} and R_{m2} that minimize

$$\text{Loss}(R_{m1}) + \text{Loss}(R_{m2})$$

$$\min_c \sum_{x_i \in R_{m1}} L(y_i, c)$$

Regression: always use squared error loss

Classification:

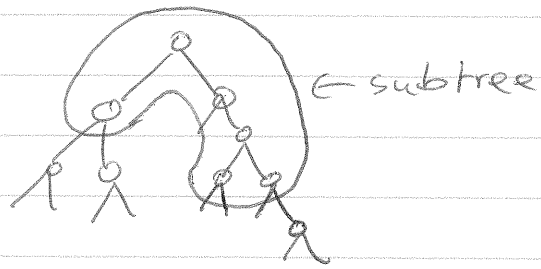
- use entropy or "Gini index" while growing
- 0/1-loss to estimate class in each region

Fig. 9.3: entropy and Gini more sensitive to changes around 0 and 1.

Pruning:

* Bigger tree → more regions → better fit training data
* Too large → overfitting because too little data per region
  too small → underfitting.

* Difficult to know when to stop growing tree
* Solution: grow very big tree, then prune to smaller tree.



choose subtree $T_\alpha \subseteq T$ that minimizes:

$$\sum_{i=1}^{N} L(y_i, T_\alpha(x_i)) + \alpha \cdot |T_\alpha|$$

$T_\alpha(x_i) = \hat{c}_m$ for m s.t. $x_i \in R_m$.

$|T_\alpha|$: nr of leaf nodes in tree

α: penalty