

Probabilistic Classification

Sanjay Lall and Stephen Boyd

EE104
Stanford University

Point and list classifiers

Point classifiers

- ▶ a classifier predicts *one* value \hat{v} , given u
- ▶ sometimes called a *point classifier* or *point predictor*, since it makes just one guess

- ▶ in this lecture we'll study classifiers that produce more than just a single guess, given u
 - ▶ an ordered list of guesses, *e.g.*, \hat{v}^{top} , \hat{v}^{2nd} , \hat{v}^{3rd} (first, second, third guesses)
 - ▶ a probability distribution on \mathcal{V} , *e.g.*, 15% rain, 85% shine

List classifiers

- ▶ a *list classifier* produces an ordered list, such as $\hat{v}^{\text{top}}, \hat{v}^{\text{2nd}}, \hat{v}^{\text{3rd}}$
- ▶ we interpret as our top, second, and third guesses
- ▶ (we show three here, but any number, or a variable number, is possible)
- ▶ we're happiest when $v = \hat{v}^{\text{top}}$, *i.e.*, our top guess is correct, a bit less happy when $v = \hat{v}^{\text{2nd}}$, *etc.*

- ▶ common application: recommendation system
 - ▶ $u \in \mathcal{U}$ is a user query, $v \in \mathcal{V}$ is the item a user wants
 - ▶ list classifier gives our top 10 (ordered) guesses

Nearest-neighbor un-embedding for a list classifier

- ▶ we can generalize nearest-neighbor un-embedding to give a list classifier
- ▶ start with embedding $\psi_i = \psi(v_i) \in \mathbb{R}^m$
- ▶ a predictor guesses $\hat{y} \in \mathbb{R}^m$
- ▶ \hat{v}^{top} is the closest representative ψ_i to \hat{y}
- ▶ \hat{v}^{2nd} is the second closest representative ψ_i to \hat{y} , *etc.*

Probabilistic classifiers

Probability distribution on \mathcal{V}

- ▶ a *probability distribution* on \mathcal{V} is a function $p : \mathcal{V} \rightarrow \mathbb{R}$
- ▶ $p(v)$ is the probability of the value v
- ▶ we have $p(v) \geq 0$ for all $v \in \mathcal{V}$ and $\sum_{v \in \mathcal{V}} p(v) = 1$
- ▶ example: with $\mathcal{V} = \{\text{RAIN}, \text{SHINE}\}$, $p(\text{RAIN}) = 0.15$, $p(\text{SHINE}) = 0.85$

- ▶ can also represent distribution p as a K -vector, with $p_i = p(v_i)$, $i = 1, \dots, K$
- ▶ in vector notation, $p \geq 0$ (elementwise) and $\mathbf{1}^T p = 1$

Probabilistic classifiers

- ▶ a *probabilistic classifier* produces a probability distribution \hat{p} on \mathcal{V} , given u
- ▶ we write this as $\hat{p} = G(u)$
- ▶ this notation means
 - ▶ G is a function that takes $u \in \mathcal{U}$ and returns a distribution (which is itself a function)
 - ▶ if $\hat{p} = G(u)$ then \hat{p} is a function
 - ▶ we can call the function; $\hat{p}(v_i)$ is the probability that $v = v_i$, when the independent variable is u
 - ▶ we can also write this as $G(u)(v_i)$
- ▶ at any point $u \in \mathcal{U}$, calling the predictor G returns the *probability distribution* \hat{p}
- ▶ we can evaluate \hat{p} at any $v_i \in \mathcal{V}$

Point classifier as a probabilistic classifier

- ▶ a point classifier can be considered a special case of a probabilistic classifier
- ▶ if point classifier predicts $\hat{v} \in \mathcal{V}$, associated probabilistic classifier returns \hat{p} , with

$$\hat{p}(v) = \begin{cases} 1 & \text{if } v = \hat{v} \\ 0 & \text{otherwise} \end{cases}$$

- ▶ *i.e.*, we return a distribution that has 100% probability on our point guess \hat{v} , and 0% on others
- ▶ we'll see this is likely a poor probabilistic classifier

Point classifier from a probabilistic classifier

- ▶ conversely, we can construct a point classifier from a probabilistic classifier
- ▶ if probabilistic classifier gives \hat{p} , our point classifier guesses

$$\hat{v} = \operatorname{argmax}_{v \in \mathcal{V}} \hat{p}(v)$$

i.e., the value in \mathcal{V} that has highest probability

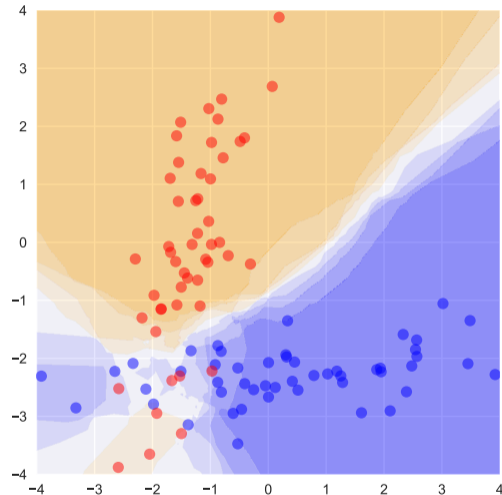
- ▶ called a *maximum likelihood classifier*
- ▶ extends to a list classifier, by giving values sorted by probability, largest to smallest

Types of probabilistic classifiers

- ▶ tree-based probabilistic classifiers
 - ▶ decision tree with nodes labeled as feature and threshold
 - ▶ leaves contain distributions \hat{p}
- ▶ nearest-neighbor probabilistic classifiers
 - ▶ find k nearest neighbors of x to x^i
 - ▶ use empirical distribution of v^i among these as \hat{p}
- ▶ later we'll see probabilistic classifiers based on linear or neural network predictors

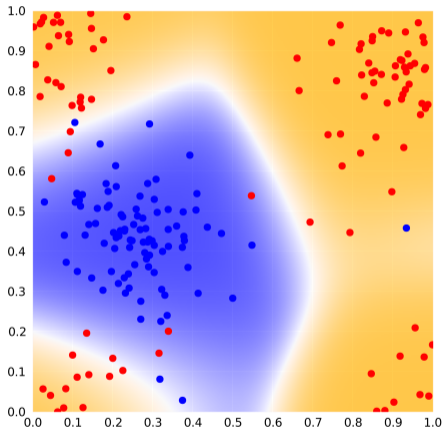
k -nearest neighbors based probabilistic classifier

- ▶ embed u^i as $x^i = \phi(u^i)$
- ▶ given u , find k nearest neighbors of $x = \phi(u)$
- ▶ guess \hat{p} as the empirical distribution of v for these neighbors
- ▶ here $k = 8$



soft k -nearest neighbors based probabilistic classifier

- ▶ embed u^i as $x^i = \phi(u^i)$
- ▶ let w be the softargmax of the squared distances between x and the x^i
- ▶ w^i is the probability of record i
- ▶ let \hat{p} be the distribution of y



Performance metrics

Judging list classifiers

- ▶ error rates versus list rank, *e.g.*, on a test data set
 - ▶ $v = \hat{v}^{\text{top}}$ (*i.e.*, our top guess is correct) for 68% of samples
 - ▶ $v \in \{\hat{v}^{\text{top}}, \hat{v}^{\text{2nd}}\}$ (*i.e.*, true value is among our top two guesses) for 79% of samples
 - ▶ $v \in \{\hat{v}^{\text{top}}, \hat{v}^{\text{2nd}}, \hat{v}^{\text{3rd}}\}$ (*i.e.*, the true value is one of our top three guesses) for 85% of samples

- ▶ average score on a test data set
 - ▶ three points for $v = \hat{v}^{\text{top}}$ (top guess correct)
 - ▶ two points if $v = \hat{v}^{\text{2nd}}$ (second guess correct)
 - ▶ one point if $v = \hat{v}^{\text{3rd}}$ (third guess correct)
 - ▶ zero points if v is not in your list (no guesses correct)

Judging a probabilistic classifier

- ▶ consider a data pair u, v with prediction $\hat{p} = G(u)$
- ▶ we'd like to have $\hat{p}(v) = G(u)(v)$ large, *i.e.*, *we assign high probability to the actual value*
- ▶ for rain / shine prediction example:
 - ▶ we want $\hat{p}(\text{RAIN})$ large when $v = \text{RAIN}$
 - ▶ we want $\hat{p}(\text{RAIN})$ small when $v = \text{SHINE}$
- ▶ there are several ways to formalize this
- ▶ we'll focus on most common formalization, based on *log-likelihood*

Likelihood

- ▶ we have a probabilistic classifier $\hat{p} = G(u)$, and data set $u^1, \dots, u^n, v^1, \dots, v^n$
- ▶ at the i th data point, the predicted probability distribution is $\hat{p}^i = G(u^i)$
- ▶ assuming outcomes v^i are independent with distributions \hat{p}^i , probability of observing these outcomes is

$$\text{prob}(v^1, v^2, \dots, v^n) = \prod_{i=1}^n \hat{p}^i(v^i)$$

- ▶ this probability is called the *likelihood* of $\hat{p}^1, \dots, \hat{p}^n$; we'd like it to be high
- ▶ a fundamental measure of how well the predicted distribution matches the data
- ▶ we can compare two probabilistic classifiers G and \tilde{G} by their associated likelihood on test data

Negative log likelihood

- ▶ it's more convenient to work with log probabilities, since the likelihood is a product
- ▶ the *negative log likelihood* of a probabilistic classifier on a data set is

$$-\log \text{prob}(v^1, v^2, \dots, v^n) = -\log \prod_{i=1}^n \hat{p}^i(v^i) = -\sum_{i=1}^n \log \hat{p}^i(v^i)$$

- ▶ the negative log likelihood is nonnegative; we'd like it to be small
- ▶ to compare likelihood on different size data sets (*e.g.*, train and test) we use the *average negative log likelihood*

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^n \log \hat{p}^i(v^i)$$

Constant probabilistic classifier

Constant probabilistic classifier

- ▶ consider a constant probabilistic classifier
- ▶ *i.e.*, distribution \hat{p} does not depend on u (which need not even exist)
- ▶ given data set v^1, \dots, v^n , guess distribution \hat{p} on \mathcal{V}
- ▶ suppose we choose \hat{p} to minimize average negative log likelihood

$$-\frac{1}{n} \sum_{i=1}^n \log \hat{p}(v^i)$$

(subject to $\hat{p}(v) \geq 0$ for all $v \in \mathcal{V}$ and $\sum_{v \in \mathcal{V}} \hat{p}(v) = 1$)

- ▶ the *empirical distribution* of the data is

$$q(v) = \text{fraction of } v^j \text{ that have value } v$$

- ▶ we'll see: *the optimal constant probabilistic classifier is $\hat{p} = q$*
- ▶ ... a very sensible prediction of \hat{p}

Cross entropy

- ▶ we can express average negative log likelihood as

$$-\frac{1}{n} \sum_{i=1}^n \log \hat{p}(v^i) = -\sum_{j=1}^K q(v_j) \log \hat{p}(v_j)$$

- ▶ the quantity $H(q, \hat{p}) = -\sum_{j=1}^K q(v_j) \log \hat{p}(v_j)$ is called the *cross entropy* of \hat{p} relative to q

- ▶ compare with the entropy $H(p) = -\sum_{k=1}^K p(v_k) \log p(v_k)$

Kullback-Leibler divergence

For p, q two probability distributions, the *Kullback-Leibler divergence* is

$$d_{kl}(q, p) = H(q, p) - H(q)$$

► $d_{kl}(q, p) \geq 0$ for all distributions p, q , because

$$\begin{aligned} d_{kl}(q, p) &= - \sum_{j=1}^K q_j \log(p_j/q_j) \\ &\geq - \sum_{j=1}^K q_j (p_j/q_j - 1) = 0 \quad \text{because } \log x \leq x - 1 \text{ for all } x > 0 \end{aligned}$$

► can be shown even if some $q_j = 0$

Constant predictor

- ▶ the optimal constant probabilistic classifier is the \hat{p} that minimizes $H(q, \hat{p})$, which is

$$H(q, \hat{p}) = d_{kl}(q, \hat{p}) + H(q)$$

- ▶ optimal choice is $\hat{p} = q$, then $H(q, \hat{p}) = H(q)$

Summary

Summary

- ▶ a point classifier makes a single guess of v , given u
- ▶ a probabilistic classifier guesses a probability distribution on \mathcal{V} , given u
- ▶ we judge a probabilistic classifier by its average log likelihood on test data