lesson 8: support vector machines

goal: maximize margin between decision boundary and classes w/ data

- **hard margins:** overfits data
- **soft margins:** allows misclassifications to generalize better with unseen data

$$\underbrace{\text{hinge loss}}_{\text{cost}} = \sum_{i=1}^{m} \max(0,\ 1 - y^{(i)}(w^T x^{(i)}))$$

$$J(w) = C \overbrace{\sum_{i=1}^{m}}^{\text{sample size}} \max(0,\ 1 - \overbrace{y^{(i)}}^{\text{true label}}(\overbrace{w^T x^{(i)}}^{\text{predicted label}})) + \frac{1}{2} \overbrace{\sum_{j=1}^{n}}^{\#\text{ of features}} w_j^2$$

$C = \frac{1}{\lambda}$ s.t. λ is the regularization parameter

$$\text{hypothesis: } h_w(x) = \begin{cases} 1 & \text{if } w^T x \geq 0 \\ -1 & \text{if } w^T x < 0 \end{cases}$$

-1: predicted label

goal:

$$\min_w J(w) = \min_w C \sum_{i=1}^{m} \max(0,\ 1 - y^{(i)}(w^T x^{(i)})) + \frac{1}{2} \sum_{j=1}^{n} w_j^2$$

Kernels

- during training, features of x are projected to a higher dimensional space

- for a datapoint $x^{(i)}$, the new features $f^{(i)} = \{f_1^{(i)}, \dots, f_m^{(i)}\}$
  - $x^{(i)}$: n features
  - $f^{(i)}$: each datapoint now has m features instead n

- given that each datapoint x is also a landmark l:

$$f = K(x, l^{(i)}) \quad \forall \text{ features} \rightarrow K(x, l^{(i)}) = 1 \text{ when } x = l^{(i)}$$

- K: kernel function | "similarity"

- updated $J(w) = C \sum_{i=1}^{m} \max(0, 1 - y^{(i)}(w^T f^{(i)})) + \frac{1}{2} \sum_{j=1}^{m} w_j^2$

Polynomial kernel

- $f = K(x, l^{(i)}) = (x \cdot l^{(i)} + r)^d$
  - d: degree
  - r: coefficient of the polynomial

radial basis function (RBF) kernel (aka gaussian kernel)

- $f = K(x, l^{(i)}) = e^{-\gamma(x - l^{(i)})^2}$

- technically, $f \in \mathbb{R}^{\infty}$, but the "kernel trick" is to calculate f without actually projecting the data point

when to use SVM vs logistic regression ?

1. features >> sample size $n >> m$

   ⇒ logistic regression or SVM w/out kernel (linear)

2. features << sample size $n << m$

   ⇒ SVM w/ kernel (if difference is moderate but still $n < m$, feature engineer to 1)

lesson 9: anomaly detection

- given a dataset $X \in \mathbb{R}^{m \times n}$ → m samples, n features
- a model $p(x)$ = probability of x being seen in dataset
- $$h(x) = y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \Rightarrow \text{anomaly} \\ 0 & \text{if } p(x) \geq \varepsilon \Rightarrow \text{normal} \end{cases}$$ (threshold: ε)
- $$p(x_j; \mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}\,\sigma_j} e^{-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}}$$
  - x_j: j^{th} feature for a sample $x \in X$

anomaly detection algorithm

1. given a datapoint $x \in X$ s.t. $x \in \mathbb{R}^n$:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

μ_j: mean of each feature j ; σ_j^2: variance of feature j

$$\mu = \{\mu_1, \mu_2, \cdots, \mu_j\} \qquad \sigma^2 = \{\sigma_1^2, \sigma_2^2, \cdots, \sigma_j^2\}$$

2. $$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$ (product of $p(x_j)$ $\forall$ features j)

   $p(x)$: for a single datapoint $x^{(i)}$

3. classify as anomaly if $p(x) < \varepsilon$

computing ε

includes both normal & anomalous data

1. using the validation dataset, compute $p(x)$ $\forall$ x in dataset

2. extract anomalies w val. data

3. $\varepsilon = \max(p(x))$ $\forall$ x that are anomalies

multivariate

same thing but w vector form lol

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \qquad \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^2$$

$$p(x_1; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} e^{\left(\frac{-1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$$

lesson 11: PCA & t-SNE

cramer's $V = \sqrt{\frac{\chi^2}{n(k-1)}}$ ⇒ magnitude of association b/w two categorical features

(n: sample size; k: # of classes)

$\eta^2 = \frac{SS_{between}}{SS_{total}}$ ⇒ magnitude of association b/w a categorical & numerical feature

principal component analysis

- a PC explains a % of variability in data

1. center and standardize data
2. calculate covariance $\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$
3. extract eigenvectors & eigenvalues of Σ
4. use the 2 largest eigenvalues & their corresponding eigenvectors
   - eigenvalues → % of variance captured
   - eigenvectors → the 2 PC's to use as new axes
5. project original data to PC's

t-SNE

- preserves local relationships between data
- perplexity:
  - determines balance b/w preserving local & global relationships in the data
  - higher P ⇒ more separation in low-dimensional space