Alex Ivensky - ECE 1395 Midterm Sheet God help me

General Concepts:

Supervised Learning: given training data

↳ regression, classification (continuous, discrete)

Unsupervised Learning: finds hidden patterns

↳ clustering

Linear Regression

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$

Cost: $J(\theta_0, \theta_1, \dots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} [\underset{\text{prediction}}{h_\theta(x^{(i)})} - \underset{\text{right answer}}{y^{(i)}}]^2$

Gradient Descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$$

} simultaneously update for every j

- Feature Scaling:

$$x_{j,new} = \frac{x_j - \mu_j}{\sigma_j}$$

← mean j

← std. dev. j

- Normal Equation: $\theta = (X^T X)^{-1} X^T y$

Logistic Regression. (classification)

- $g(z) = \frac{1}{1+e^{-z}}$,
- $h_\theta(x) = g(\theta^T x)$.

→ $h(x) = P(y = 1 \mid x)$

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \cdot \frac{1}{m}$$

(simultaneously update all θ_j)

}

Regularization. now also minimizing magnitudes of θ to reduce overfitting.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \overset{\downarrow}{\sum_{j=1}^{n}} \underline{\theta_j^2}\right]$$

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$(j = \cancel{0}, 1, 2, 3, \dots, n)$

}

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}\right)^{-1} X^T y$$

KNN

- Classification:

$$y = \text{argmax}_c \#(y_i = c)$$

$$y = \text{argmax}_c \sum_{i \in N_K(x)} \text{I}(y_i = c)$$

← set of K-NN's

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

- Regression:

$$y = \frac{1}{K} \sum_{i \in N_K(x)} y_i$$

Weighted KNN

- Neighbors weighted differently:
  - Use all samples, i.e. K = N
  - Weight on i-th sample: $w_i = e^{\frac{-||\mathbf{x}-\mathbf{x}_i||^2}{\sigma^2}}$
  - σ = the bandwidth parameter, expresses how quickly our weight function "drops off" as points get further and further from the query x
- Classification: $y = \text{argmax}_c \sum_{i=1}^{N} w_i \text{I}(y_i = c)$
- Regression: $y = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}$

$$d(\mathbf{x}, \mathbf{z}) = [\sum_{i=1}^{D} (x_i - z_i)^2]^{\frac{1}{2}}$$

Perceptron linear classifier

Assume $y \in \{-1, 1\}$

$h_\theta(x) = f(\theta_T x)$ where $f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$

So $\theta^T x > 0$ → classify 1
$\theta^T x < 0$ → classify -1.

$J(\theta) = -\sum_{i \in M} \theta^T x^{(i)} y^{(i)}$ M is set of wrongly classified vectors.

Perceptron Gradient Descent:

$$\theta^{(\tau+1)} = \theta^{(\tau)} + \alpha \sum_{i \in M} x^{(i)} y^{(i)}$$

PCA method of dimensionality reduction.

$C = (x - \mu)(x - \mu)^T$, covariance matrix

$$= Q \Lambda Q^{-1}$$

$$= \begin{bmatrix} | & & | \\ ev_1 & \dots & ev_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} — ev_1^T — \\ \vdots \\ — ev_n^T — \end{bmatrix}$$

eigenvectors eigenvalues

% variance captured by the top k evs

$$= \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j} \quad \text{want} \geq 0.9.$$

- Given: M data points $x_1,\ldots,x_M$ in R^d where d is big ($d \gg M$)
- We want some directions, $u_1, u_2, \ldots, u_k, \ldots, u_d$, in R^d that capture most of the variation of the x_i. The coefficients would be: $\alpha_k = u_k^T(x_i - \mu)$
  (μ: mean of data points)

Let Φ_i be the (very big vector of length d) that is face image I with the mean image subtracted.

Define $C = \frac{1}{M}\sum \Phi_i \Phi_i^T = AA^T$

where $A = [\Phi_1 \Phi_2 \ldots \Phi_M]$ is the matrix of faces, and is $d \times M$.

Note: C is a huge $d \times d$ matrix (remember d is the length of the vector of the image).

Recognition with eigenfaces

- Given novel image **x**:
  - Project onto subspace:

$$w_1,\ldots,w_k = [u_1^T(\mathbf{x}-\mu), \ldots, u_k^T(\mathbf{x}-\mu)]$$

  - Optional: check reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$ to determine whether image is really a face
  - Classify as closest training face in k-dimensional subspace

SVM.

x_i positive $(y_i = 1)$: $\theta^T x_i \geq \textcircled{1}$

x_i negative $(y_i = -1)$: $\theta^T x_i \leq \textcircled{-1}$

For support vector, $\theta^T x_i = \pm 1$

- Want line that maximizes the margin.

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

- Distance between point and line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors:

$$\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left|\frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|}\right| = \frac{2}{\|\mathbf{w}\|}$$

Margin

$$\boxed{\begin{array}{l} \text{Minimize } \dfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{Subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \end{array}} \Rightarrow$$

The "kernel trick"

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x}_i \rightarrow \varphi(\mathbf{x}_i)$, the dot product becomes: $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$
- A *kernel function* is similarity function that corresponds to an inner product in some expanded feature space
- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that: $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$

Using SVMs

$x_1, x_1^2 x_2,$ 3rd feature → 150K 5000 x_2 x_3 $x_1^2, x_2^2, x_3^2, \ldots$ $x.$

1. Select a kernel function.

   This implicitly assumes a mapping to a higher dimensional (yet, not known) space.

g optim 2. Compute pairwise kernel values between labeled examples.

3. Use this "kernel matrix" to solve for SVM support vectors & alpha weights.

4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output. $f(x_{test}) = \sum \alpha \cdot y \cdot K$ x_{te}

Soft-margin SVMs (allow misclassification)

data samples

Misclassification cost

Slack variable

$$\underset{\boldsymbol{w}}{\min} \; \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{Maximize margin}} + \underbrace{C\sum_{i=1}^{N}\xi_i}_{\text{Minimize misclassification}}$$

The w that minimizes…

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq \boxed{1 - \xi_i,}$$

$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

- Solution: $\theta = \sum_i \alpha_i y_i x_i$

$\forall$ S.V

Learned weight

Support vector

- Classification function:

$$f(x) = \text{sign}(\theta \cdot x)$$

$$= \text{sign}\left(\sum_i \alpha_i y_i x_i \cdot x\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

<0

$$D = \frac{\theta^T x}{\|\theta\|}$$

↑ l2-norm