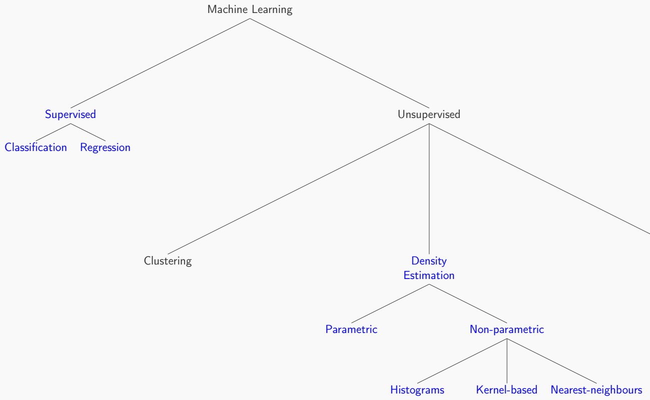
CS-567 Machine Learning

Nazar Khan

PUCIT

Lecture 15
Probabilistic Models for Linear Classification

Machine Learning So Far ...

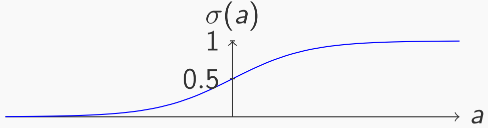


Linear Models for Classification

- Discriminant Functions
  - Least Squares ($\mathbf{w}^*$ via pseudoinverse)
  - Fisher's Linear Discriminant ($\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$)
  - Perceptron ($\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \eta t_n \phi_n$ for every misclassified sample x_n)
- Generative Models
  - $p(\mathcal{C}_k|\phi) = \frac{p(\phi|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\phi)} = \frac{p(\phi|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\phi|\mathcal{C}_j)p(\mathcal{C}_j)}$
  - Model class-conditional densities $p(\phi|\mathcal{C}_k)$ and the priors $p(\mathcal{C}_k)$ from data.
  - We will not cover such models because
    1. they require too many parameters for high dimensional inputs
    2. perform poorly when assumed density models do not represent the data properly.
- Discriminative Models
  - Since classification is based on posterior $p(\mathcal{C}_k|\phi)$, model it directly.

Background Math
Logistic Sigmoid Function

- For $a \in \mathbb{R}$, the *logistic sigmoid* function is given by $\sigma(a) = \frac{1}{1+e^{-a}}$
- *Sigmoid* means S-shaped.
- Maps $-\infty \leq a \leq \infty$ to the range $0 \leq \sigma \leq 1$. Also called *squashing* function.
- **Can be treated as a probability value.**
- Symmetry $\sigma(-a) = 1 - \sigma(a)$. **Prove it.**
- Easy derivative $\sigma' = \sigma(1 - \sigma)$. **Prove it.**



Background Math

Softmax Function

- For real numbers $a_1, \ldots, a_K$, the *softmax* function is given by $\frac{e^{a_k}}{\sum_j e^{a_j}}$.
- Softmax is ≈ 1 when $a_k >> a_j \; \forall j \neq k$ and ≈ 0 otherwise.
- Provides a smooth (differentiable) approximation to finding the index of the maximum element.
  - Compute softmax for $1, 10, 100$.
  - Does not work everytime.
    - Compute softmax for $1, 2, 3$. (Solution: scale-up/scale-down)
    - Compute softmax for $1, 10, 1000$. (Solution: subtract/add)
- Also called the *normalized exponential* function (for obvious reasons).
- **Can be treated as probability values**.
- Take-home Quiz 1: Show that $\frac{\partial y_k}{\partial a_j} = y_k(\delta_{jk} - y_j)$.
- **You must know this in order to understand later parts of the course.**

Background Math

Positive Definite Matrices

- A square matrix $\mathbf{M}$ is positive definite if *for every* non-zero vector $\mathbf{x}$, $\mathbf{x}^T\mathbf{M}\mathbf{x} > 0$.
- Positive semidefinite for the condition $\mathbf{x}^T\mathbf{M}\mathbf{x} \geq 0$.
- In 1D, a function f is convex if its second derivative f'' is always positive. This proves existence of *unique, global* minimum.
- In more than 1D, a function f is convex if its Hessian matrix (of second derivatives) $\mathbf{H}$ is positive definite. This proves existence of *unique, global* minimum.

Discriminative Models for Classification

Logistic Regression

- For two classes, model via logistic sigmoid.
  - $p(\mathcal{C}_1|\phi) = \sigma(\mathbf{w}^T\phi + w_0)$.
  - Leads to *logistic regression* for learning $\mathbf{w}^*$ and w_0^*.
- For more than two classes, model via softmax.
  - $p(\mathcal{C}_k|\phi) = \frac{e^{\mathbf{w}_k^T\phi + w_{k\mathbf{0}}}}{\sum_j e^{\mathbf{w}_j^T\phi + w_{j\mathbf{0}}}}$.
  - Leads to *multiclass logistic regression* for learning $\mathbf{w}_k^*$ and w_{k0}^*.
- In the following, we will absorb the bias term w_0 into the parameter vector $\mathbf{w}$ and add a constant input $\phi_0(\mathbf{x}) = 1$ so that we can write activation simply as $a = \mathbf{w}^T\phi$.

Logistic Regression
Formulation

- Assume i.i.d. data $\{\phi_n, t_n\}_1^N$ with binary targets $t_n \in \{0, 1\}$.
- Model outputs via logistic sigmoid as $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(\mathbf{w}^T\phi_n)$.
- Likelihood can be written as

$$p(t_1, \ldots, t_N|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

- Negative log-likelihood becomes

$$E(\mathbf{w}) = -\ln p(t_1, \ldots, t_N|\mathbf{w}) = -\sum_{n=1}^{N} t_n \ln y_n + (1 - t_n)\ln(1 - y_n)$$

which is also called the *cross-entropy* error function.

Logistic Regression

Gradient

- Gradient can be written as **(Prove it)**

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n = \sum_{n=1}^{N} \text{error}_n \times \text{input}_n$$

- Now stochastic gradient descent (SGD) can be used to find $\mathbf{w}^*$.
- However, the error function $E(\mathbf{w})$ is *convex* and therefore has a unique global minimum.
- Instead of gradient descent, we can use the more efficient iterative scheme known as the *Newton-Raphson* method.

Logistic Regression
Newton-Raphson Updates

- Newton-Raphson update for minimising *any* function $E(\mathbf{w})$ is given as

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \mathbf{H}^{-1}\nabla_{\mathbf{w}}E(\mathbf{w})$$

  where $\mathbf{H}$ is the *Hessian matrix* composed of second derivatives $\frac{\partial^2 E}{\partial w_i \partial w_j}$.
- To apply Newton-Raphson updates to the cross-entropy error, we need the gradient $\nabla_{\mathbf{w}}E(\mathbf{w})$ as well as the Hessian

$$\mathbf{H} = \nabla_{\mathbf{w}}\nabla_{\mathbf{w}}E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T$$

- Notice that Hessian $\mathbf{H}$ depends on the current estimate $\mathbf{w}^{\tau}$ through its dependence on the y_n.

Logistic Regression
Newton-Raphson Updates

- Take-home Quiz 1: Using matrix-vector notation, show that
  1. The gradient can be written as $\mathbf{\Phi}^T(\mathbf{y}-\mathbf{t})$ where $\mathbf{\Phi}$ is the $N \times M$ design matrix, $\mathbf{y}$ is the vector of per-sample outputs and similarly for targets $\mathbf{t}$.
  2. The Hessian $\mathbf{H}$ can be written as $\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$ where $\mathbf{R}$ is a diagonal $N \times N$ matrix with elements $R_{nn} = y_n(1-y_n)$.
  3. $\mathbf{H}$ is positive definite.

Logistic Regression
Newton-Raphson Updates

- We can now write the Newton-Raphson updates for minimising the cross-entropy error

$$\begin{aligned}
\mathbf{w}^{\tau+1} &= \mathbf{w}^{\tau} - \mathbf{H}^{-1}\nabla_{\mathbf{w}}E(\mathbf{w}) \\
&= \mathbf{w}^{\tau} - (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T(\mathbf{y}-\mathbf{t}) \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}(\Phi^T\mathbf{R}\Phi)\mathbf{w}^{\tau} - (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T(\mathbf{y}-\mathbf{t}) \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}\left\{(\Phi^T\mathbf{R}\Phi)\mathbf{w}^{\tau} - \Phi^T(\mathbf{y}-\mathbf{t})\right\} \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T\left\{\mathbf{R}\Phi\mathbf{w}^{\tau} - (\mathbf{y}-\mathbf{t})\right\} \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T\left\{\mathbf{R}\Phi\mathbf{w}^{\tau} - \mathbf{R}\mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right\} \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T\mathbf{R}\underbrace{\left\{\Phi\mathbf{w}^{\tau} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right\}}_{\mathbf{z}} \\
&= (\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T\mathbf{R}\mathbf{z}
\end{aligned}$$

Logistic Regression
Iterative Reweighted Least Squares

- This is the same as the solution to $\arg\min_{\mathbf{w}} ||\mathbf{R}(\Phi\mathbf{w} - \mathbf{z})||^2$ which is a weighted least squares problem.
  - **How is it weighted least squares?**.
  - **Show that the solution is $(\Phi^T\mathbf{R}\Phi)^{-1}\Phi^T\mathbf{R}\mathbf{z}$?**.
- So the iterative Newton-Raphson updates correspond to weighted least squares with weight matrix $\mathbf{R}$.
- But weights depend on current $\mathbf{w}^\tau$ and therefore weights are recomputed for every iteration.
- Therefore, these Newton-Raphson iterations are known as the *iterative reweighted least squares (IRLS)* algorithm.

Multiclass Logistic Regression
Formulation

- For $K > 2$ classes, model posterior via softmax.

$$p(\mathcal{C}_k|\phi_n) = y_{nk} = \frac{e^{a_{nk}}}{\sum_j e^{a_{nj}}} = \frac{e^{\mathbf{w}_k^T\phi_n}}{\sum_j e^{\mathbf{w}_j^T\phi_n}}$$

- Trick to avoid $\frac{\infty}{\infty}$: use $y_{nk} = \frac{e^{a_{nk}-m}}{\sum_j e^{a_{nj}-m}}$ where $m = \max(a_{n1}, \ldots, a_{nK})$. **(How will y_{nk} be correct now?)**
- Assume i.i.d. data $\{\phi_n, \mathbf{t}_n\}_1^N$ using 1-of-K coding for $\mathbf{t}_n$.

Multiclass Logistic Regression
Formulation

- Likelihood can be written as

$$p(\mathbf{t}_1, \ldots, \mathbf{t}_N | \mathbf{W}) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathcal{C}_k | \phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

- Negative log-likelihood becomes

$$E(\mathbf{w}) = -\ln p(\mathbf{t}_1, \ldots, \mathbf{t}_N | \mathbf{W}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

which is also called the *cross-entropy* error function for multiclass classification.

Multiclass Logistic Regression
Gradient

- Gradient is given by

$$\nabla_{\mathbf{w}_j} E(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{t_{nk}}{y_{nk}}\frac{\partial y_{nk}}{\partial a_{nj}}\frac{da_{nj}}{d\mathbf{w}_j}$$
$$= -\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{t_{nk}}{y_{nk}} y_{nk}(\delta_{jk} - y_{nj})\phi_n$$
$$= \sum_{n=1}^{N}(y_{nj} - t_{nj})\phi_n = \underbrace{\sum_{n=1}^{N}\text{error}_n \times \text{input}_n}_{\text{as for log. reg.}}$$

- This allows us to use SGD.

Multiclass Logistic Regression

Hessian

- As before, batch alternative is IRLS where the Hessian matrix can be computed in blocks of size $M \times M$ via

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{W}) = \sum_{n=1}^{N} y_{nk}(\delta_{jk} - y_{nj})\phi_n \phi_n^T$$

- The Hessian is positive definite and therefore multiclass logistic regression too is a convex optimisation problem and has a unique, global minimiser $\mathbf{W}^*$.
- Newton-Raphson updates are

$$\mathbf{W}^{\tau+1} = \mathbf{W}^{\tau} - \mathbf{H}^{-1} \nabla_{\mathbf{W}} E(\mathbf{W})$$

- **Note, however, that for high-dimensional spaces, SGD might be a better option memory-wise.**