Applied Machine Learning

Classification - Support Vector Machines

Support Vector Machine

- Training of a Support Vector Machine
  - Stochastic Gradient Descent
  - Regularization Constant
- Multi-class problems with Support Vector Machines

Training a Support Vector Machine

Training dataset: N pairs $(\mathbf{x}_i, y_i)$

Classification function: $f(\mathbf{x}) \quad = \quad sign(\mathbf{a}^\top \mathbf{x} + b) \quad \in \quad \{-1, +1\}$

$$\mathbf{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_{k-1} \end{bmatrix} \qquad \mathbf{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_{k-1} \end{bmatrix} \qquad b$$

Training: find $\mathbf{a}$, b, and λ that minimize the cost function:

$$S(\mathbf{a}, b; \lambda) = \frac{1}{N} \sum_{i=1}^{N} [\max(0, 1 - y_i * (\mathbf{a}^\top \mathbf{x} + b))] + \lambda \frac{1}{2} \mathbf{a}^\top \mathbf{a}$$

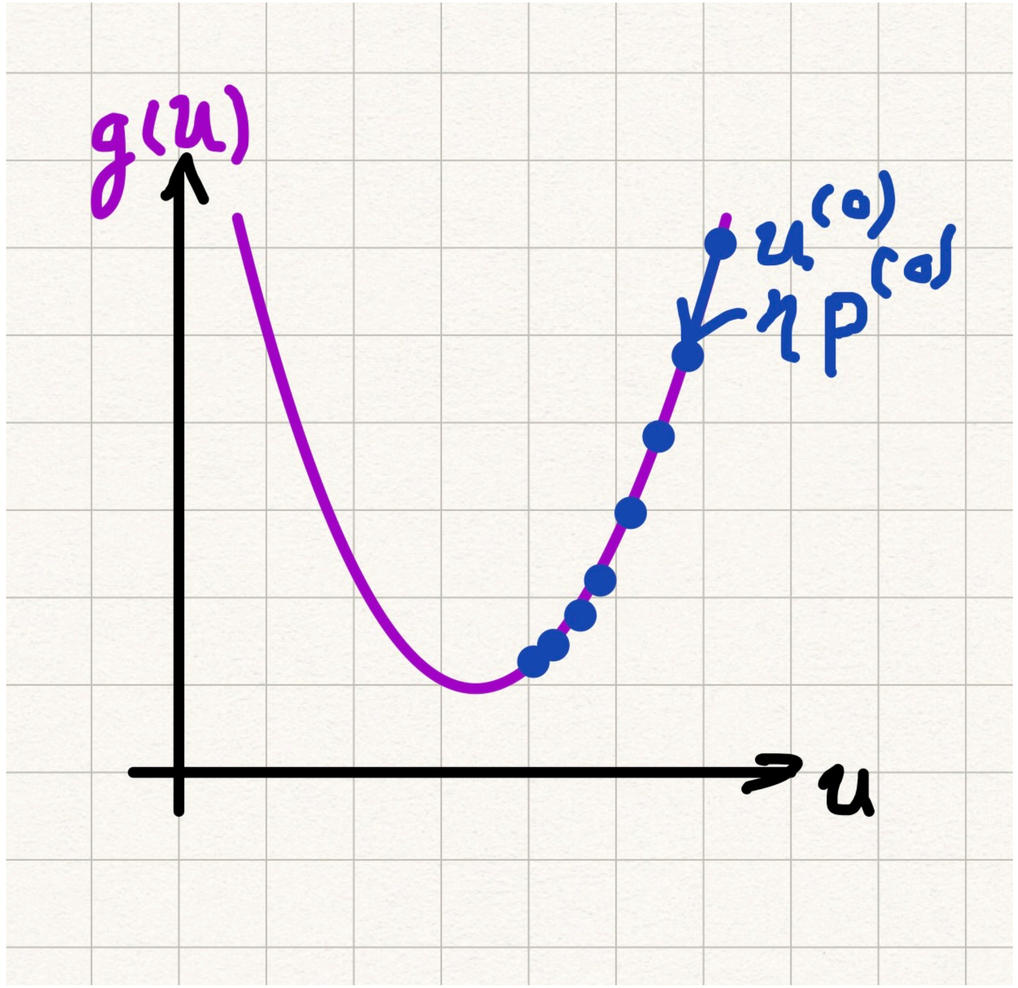
- Find $\mathbf{a}$ and b
- Find λ

$\mathbf{a}$ and b: Stochastic Gradient Descent

- Rescale data so each feature has unit variance
- $\mathbf{u} = \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix}$
- Minimize cost function
- $$\begin{aligned} g(\mathbf{u}) &= [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u}) \\ g_i(\mathbf{u}) &= \max(0, 1 - y_i * (\mathbf{a}^\top \mathbf{x} + \mathbf{b})) \\ g_0(\mathbf{u}) &= \lambda \frac{1}{2} \mathbf{a}^\top \mathbf{a} \end{aligned}$$

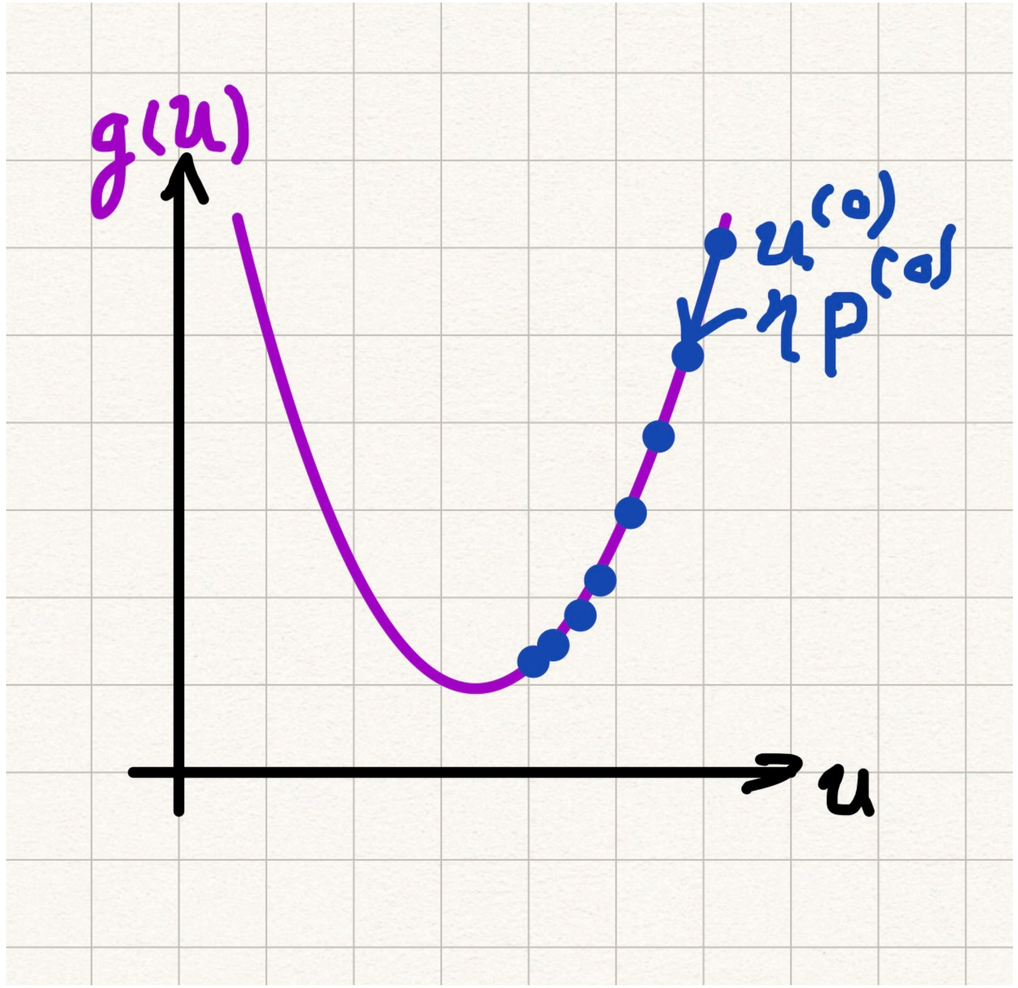
$\mathbf{a}$ and b: Stochastic Gradient Descent

- Minimize cost function:
  - $g(\mathbf{u}) \quad = \quad [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Start at random point for $\mathbf{u}^{(0)}$
- Iteratively, at step (n)
  - compute descent direction $\mathbf{p}^{(n)}$ and step size η
  - so that $g(\mathbf{u}^{(n)} + \eta\mathbf{p}^{(n)}) < g(\mathbf{u}^{(n)})$
  - update $\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \eta\mathbf{p}^{(n)}$



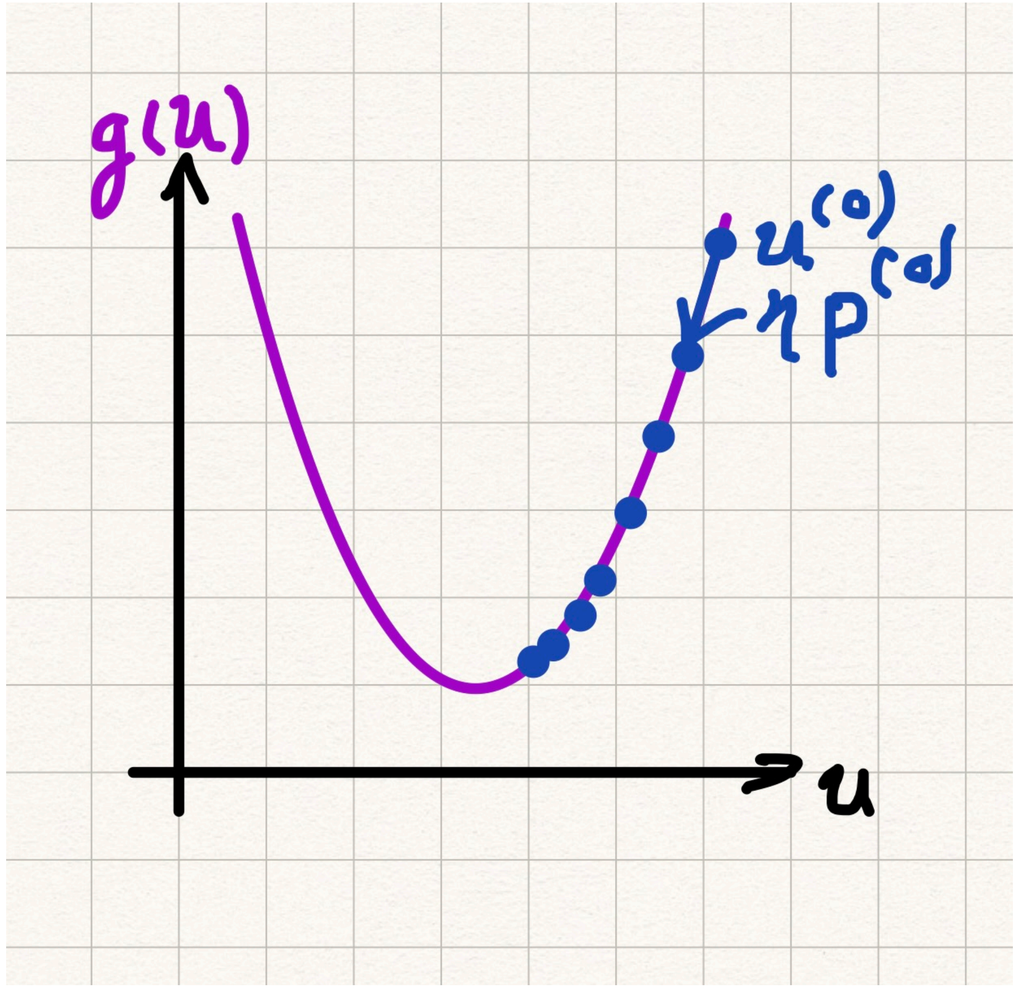
SGD: Descent Direction

- Minimize cost function:
  - $g(\mathbf{u}) = [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Descent direction:
  - $\mathbf{p}^{(n)} = -\nabla g((u)^{(n)})$
  - $= -([\frac{1}{N}\sum_{i=1}^{N} \nabla g_i(\mathbf{u})] + \nabla g_0(\mathbf{u}))$
- Estimation through mean of batch:
  - $\mathbf{p}^{(\mathbf{n})_{\mathbf{N_b}}} = -([\frac{1}{N_b}\sum_{j\in batch} \nabla g_j(\mathbf{u})] + \nabla g_0(\mathbf{u}))$



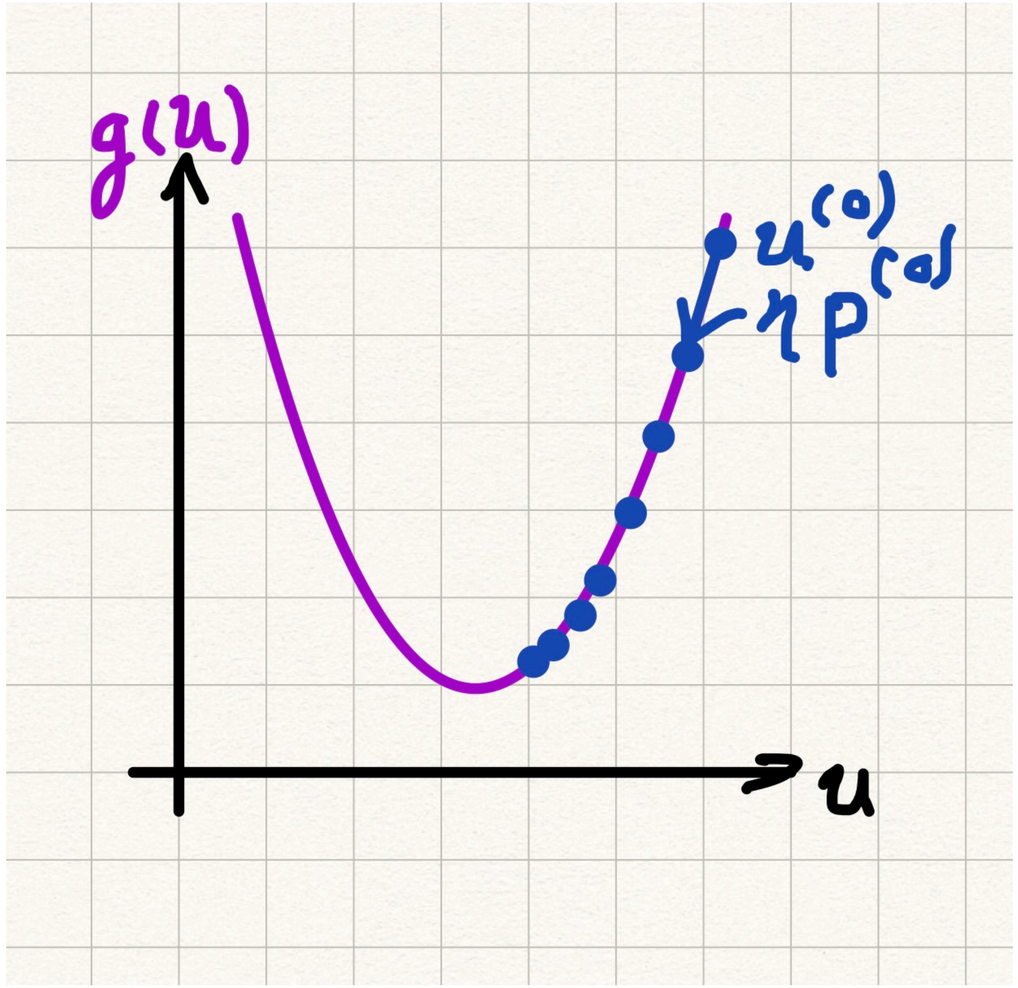
SGD: Step Length

- Minimize cost function:
  - $g(\mathbf{u}) = [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Step length: η
  - if too small: many steps
  - if too large: overshoots and oscillation
  - refine η as minimum gets closer



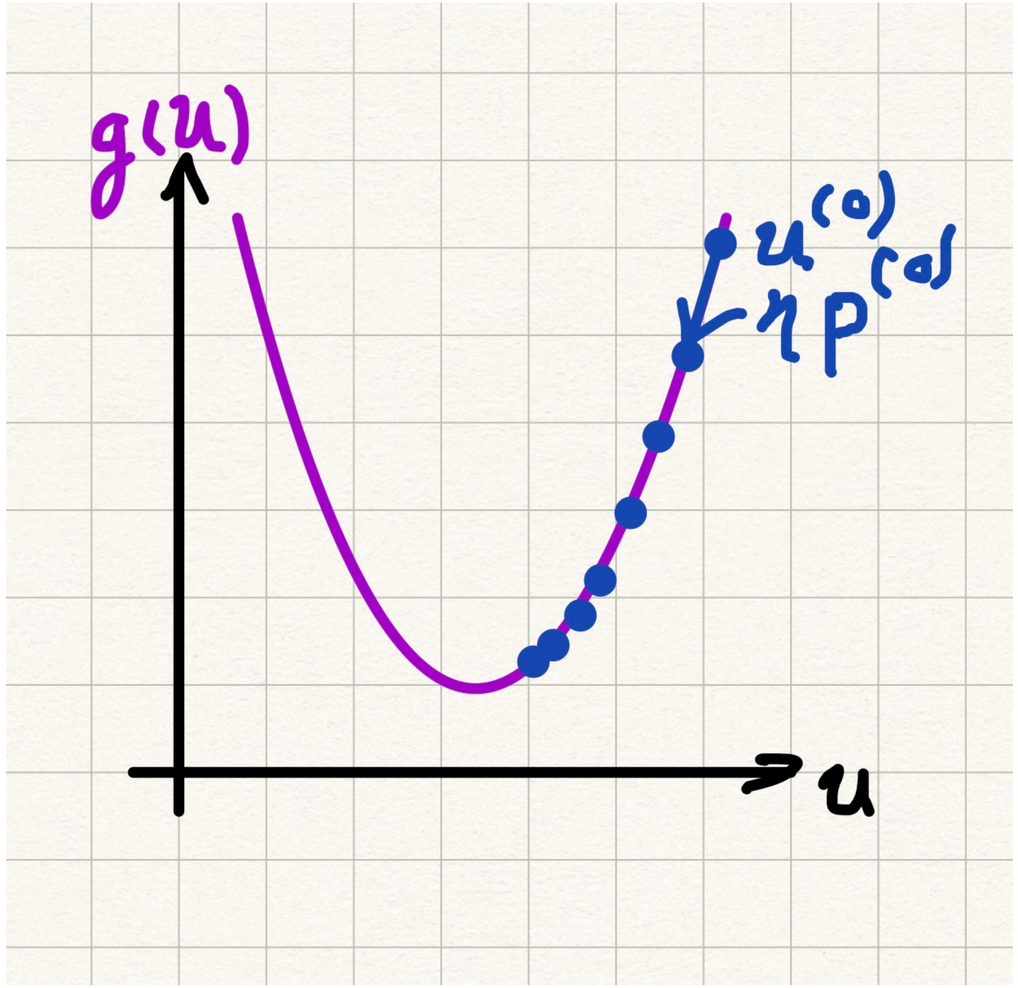
SGD: Step Length

- Minimize cost function:
  - $g(\mathbf{u}) \quad = \quad [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Epoch
  - one pass on training set of size N
  - each step sees a batch of N_b items
  - the dataset is covered in $\frac{N}{N_b}$ steps
  - step size in epoch e: $\eta^{(e)} = \frac{m}{e+l}$
  - constants m and l: tune on small subsets



SGD: Step Length

- Minimize cost function:
  - $g(\mathbf{u}) = [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Season
  - constant number of iterations, much smaller than epochs
  - each step sees a batch of N_b items
  - step size in season s: $\eta^{(s)} = \frac{m}{s+l}$
  - constants m and l: tune on small subsets



SGD: Gradient

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix} \qquad \nabla g = \begin{bmatrix} \frac{\partial g}{u_1} \\ \vdots \\ \frac{\partial g}{u_d} \end{bmatrix}$$

- Batches of 1 sample at each training step: $N_b = 1$

- $\nabla g(\mathbf{u}) = \nabla(\max(0, 1 - y_i * (\mathbf{a}^\top \mathbf{x}_i + b)) + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a})$

$$\begin{bmatrix} \mathbf{a}^{(n+1)} \\ b^{(n+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(n)} - \eta \nabla_{\mathbf{a}} \\ b^{(n)} - \eta \nabla_b \end{bmatrix}$$

$y_i * (\mathbf{a}^\top \mathbf{x}_i + b) \geq 1$. Correct, away from boundary

- $\nabla_a(0 + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a}) = \lambda \mathbf{a}$

- $\nabla_b(0 + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a}) = \mathbf{0}$

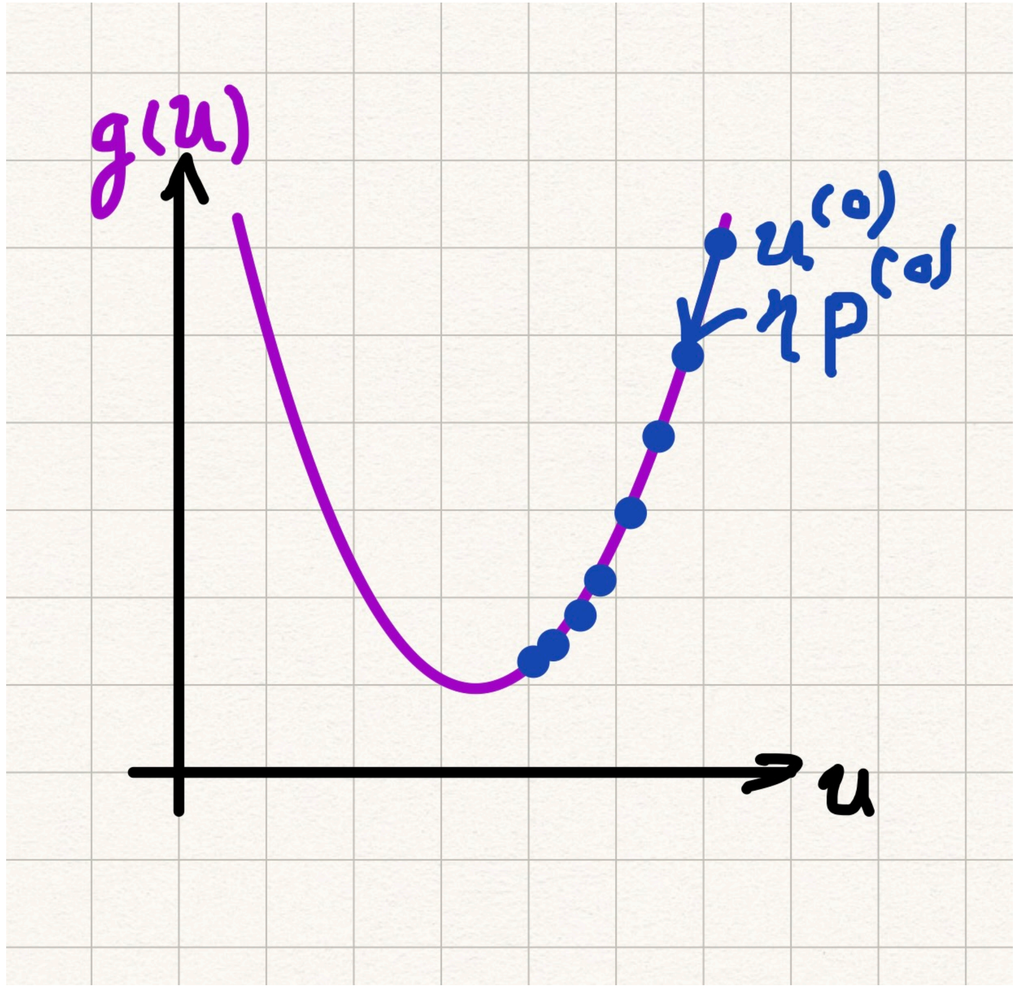
$y_i * (\mathbf{a}^\top \mathbf{x}_i + b) < 1$. Correct, close to boundary, or incorrect

- $\nabla_a(1 - y_i * (\mathbf{a}^\top \mathbf{x}_i + b) + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a}) = -\mathbf{y_i} * \mathbf{x_i} + \lambda \mathbf{a}$

- $\nabla_b(1 - y_i * (\mathbf{a}^\top \mathbf{x}_i + b) + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a}) = -\mathbf{y_i}$

SGD: Stop

- Minimize cost function:
  - $g(\mathbf{u}) = [\frac{1}{N}\sum_{i=1}^{N} g_i(\mathbf{u})] + g_0(\mathbf{u})$
- Season or Epochs
- Stop when
  - predefined number of seasons or epochs
  - error on held-out data items is smaller than some threshold
  - other criteria



Regularization constant λ

- Regularization constant λ in $g_o(\mathbf{u}) = \lambda\frac{\mathbf{1}}{\mathbf{2}}\mathbf{a}^\top\mathbf{a}$: try at different scales (e.g., $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$)
- split dataset into Test Set and Train Set for cross-validation
  - for each λ_i in set to try, iteratively
    - generate a new Fold from Train Set with a Cross-Validation Train Set and Validation Set
    - using testing λ_i apply Stochastic Gradient Descent on Cross-Validation Train Set to find $\mathbf{a}, b$
    - evaluate $\mathbf{a}, b, \lambda_i$ on Validation Set and record error for current Fold
  - cross-validation error for chosen λ_i is average error over all the Folds
- using λ with the lowest cross-validation error, apply SGD on whole training set to get final $\mathbf{a}, b$

Summary: Finding $\mathbf{a}$, b, and λ

- Dataset with N pairs $(\mathbf{x}_i, y_i)$
- Rescale each $\mathbf{x}_i$ to have unit variance
- Use cross-validation to train an SVM on Training Set for each option in λ at different scales
  - choose the λ that achieve highest accuracy
- Train an SVM with the chosen λ to find the final $\mathbf{a}$ and b
- Compute accuracy and error on Test Set

Multi-class classification

- One SVM to predict each class
- Encode in binary word
  - 001 = Class 1
  - 010 = Class 2
  - 100 = Class 3
- Issue: More than one class can be predicted

Multi-class classification

- one-vs-all
  - One SVM to predict each class
  - To classify an item, compute score for each SVM
  - Class reported is the one with largest SVM score

Multi-class classification

- all-vs-all
  - One SVM for each pair of classes
    - SVM 1: class 1 vs class 2
    - SVM 2: class 1 vs class 3
    - SVM 3: class 2 vs class 3
  - To classify an item, each SVM votes on its classes
  - Report class with more votes
  - Issue: need $O(N^2)$ for N classes

Support Vector Machine

- Training of a Support Vector Machine
  - Stochastic Gradient Descent
  - Regularization Constant
- Multi-class problems with Support Vector Machines

Applied Machine Learning

Classification - Support Vector Machines