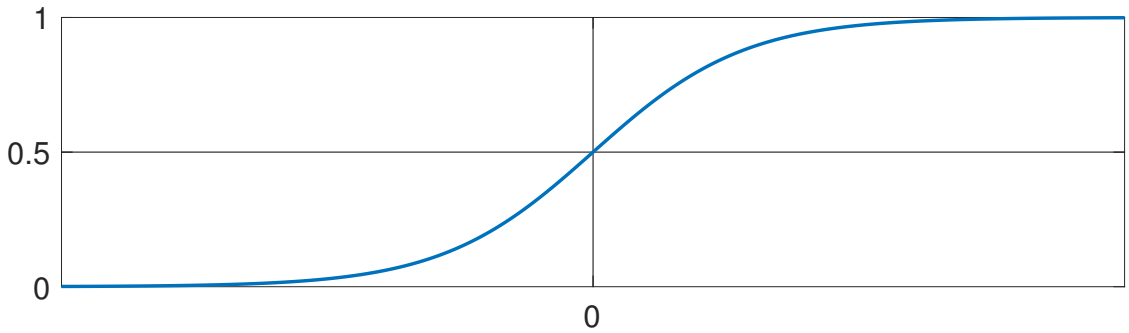
Linear Predictors Part 2

COMPSCI 371D — Machine Learning

Outline

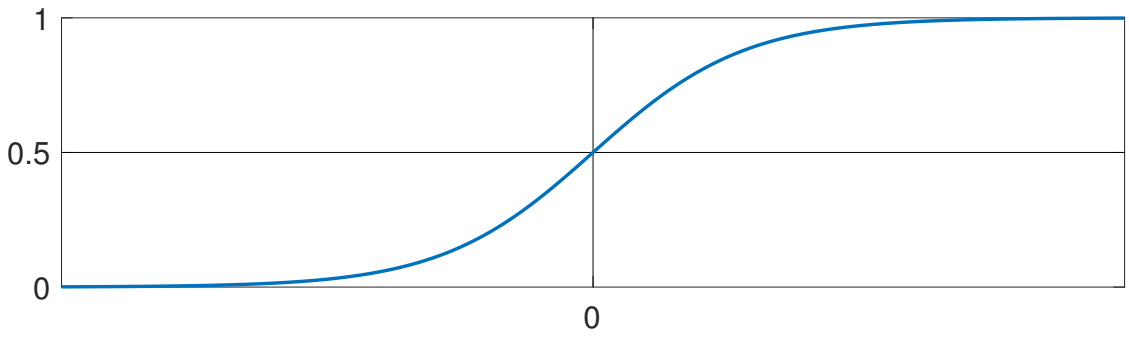
Ingredient 2: The Logistic Function

- Want to make the score of $\mathbf{x}$ be only a function of the signed distance$\Delta(\mathbf{x})$
- Given Δ_0, all points s.t. $\Delta(\mathbf{x}) = \Delta_0$ have the same score
- Score $s(\mathbf{x}) = f(\Delta(\mathbf{x}))$
- How to pick f?
- $\lim_{\Delta \to -\infty} f(\Delta) = 0 \qquad f(0) = 1/2 \qquad \lim_{\Delta \to \infty} f(\Delta) = 1$
- *Logistic function:* $f(\Delta) \stackrel{\text{def}}{=} \frac{1}{1+e^{-\Delta}}$



The Logistic Function

- *Logistic function:* $f(\Delta) \stackrel{\text{def}}{=} \frac{1}{1+e^{-\Delta}}$



- Scale-free: Why not $\frac{1}{1+e^{-\Delta/c}}$?
- Can use both c and $\Delta(\mathbf{x}) \stackrel{\text{def}}{=} \frac{b+\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|}$

  ... or more simply use no c but use $a(\mathbf{x}) \stackrel{\text{def}}{=} b + \mathbf{w}^T\mathbf{x}$
- The affine function takes care of scale implicitly
- Score: $s(\mathbf{x}) \stackrel{\text{def}}{=} f(a(\mathbf{x})) = \frac{1}{1+e^{-b-\mathbf{w}^T\mathbf{x}}}$
- Write $s(\mathbf{x}\,;\, b, \mathbf{w})$ to remind us of dependence

Optimize the Regressor, not the Classifier

- We would like to minimize the average of
$$\ell_{0\text{-}1}(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$
- However, $\frac{\partial \ell_{0\text{-}1}}{\partial \hat{y}} = 0$ almost everywhere (and is undefined everywhere else)
- Use the score $p = s(\mathbf{x}\,;\ b, \mathbf{w})$ instead of $\hat{y}$:
- $\hat{y} \in \{0, 1\}$ while $p \in [0, 1]$
- Instead of measuring the loss on $\hat{y} = h(\mathbf{x})$, we measure it on $p = s(\mathbf{x}\,;\ b, \mathbf{w})$, a proxy for $\hat{y}$
- We still need a different $\ell(y, p)$ for differentiability and $\frac{\partial \ell}{\partial p} \neq 0$

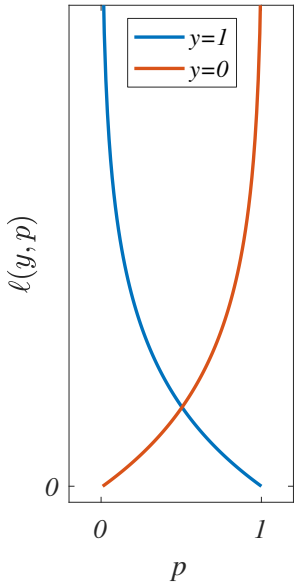
Differentiable Risk with Nonzero Gradient

- We want $\ell(y, p)$ to be differentiable in p and $\frac{\partial \ell}{\partial p} \neq 0$
- Since $p = s(\mathbf{x}\,;\ b, \mathbf{w})$ is differentiable in $(b, \mathbf{w})$, then ℓ will be, too, and the gradient has a chance to be nonzero
- Why do we insist on differentiability and nonzero gradient, again?
- Risk: $L_T(b, \mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, s(\mathbf{x}_n\,;\ b, \mathbf{w}))$
- Use a gradient method (steepest descent, Newton, ...)
- We have not yet chosen the specific form of ℓ
- We can make $L_T(b, \mathbf{w})$ a differentiable *and convex* function of $\mathbf{v} = (b, \mathbf{w})$ by a suitable choice of ℓ

The Cross-Entropy Loss

$$\ell(y, p) \stackrel{\text{def}}{=} \begin{cases} -\log p & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 0 \end{cases}$$

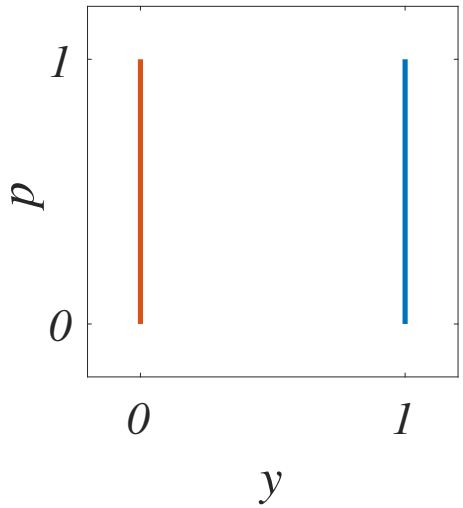
- Base of log is unimportant: unit of loss is conventional

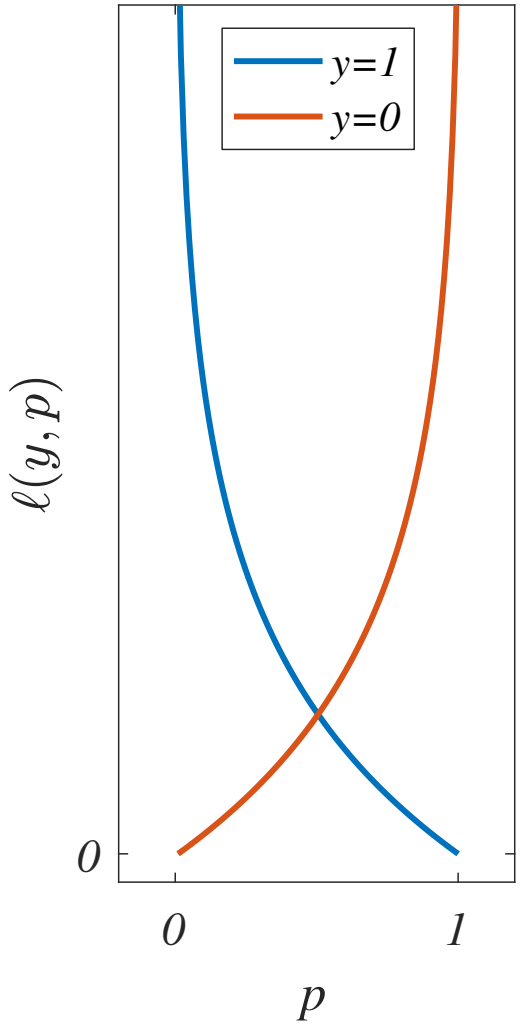


- Same as $\ell(y, p) = -y \log p - (1-y) \log(1-p)$
  (Second is more convenient for differentiation)

The Cross-Entropy Loss

- Domain: $\{0,1\} \times [0,1]$
  $\ell(1,p) = \ell(0,1-p)$
  $\ell(1,1/2) = \ell(0,1/2) = -\log(1/2)$





Why Cross-Entropy?

- Literature (and Appendix in the class notes) gives an interpretation in terms of information theory
- A more cogent explanation: With cross-entropy and the logistic function,
  - *The risk becomes a convex function of the parameters* $\mathbf{v} = (b, \mathbf{w})$
  - The gradient and Hessian of the risk are easy to compute
- A crucial cancellation occurs when computing derivatives of the risk with respect to the parameters
- You *will* be asked to *use* gradient and Hessian, and be able to compute them
- You will *not* be asked to *remember* their formulas, or know how to derive them

The Magic

- Logistic function and loss were chosen to simplify the math
- Here is the magic:

  $L_T(\mathbf{v}) = L_T(\ell(f(a(\mathbf{v}))))$, so $\nabla L_T = \frac{dL_T}{d\ell}\frac{d\ell}{df}\frac{df}{da}\nabla a$

  $\ell = -y \log f - (1-y)\log(1-f)$ so that $\frac{d\ell}{df} = \frac{f-y}{f(1-f)}$

  $f(a) = \frac{1}{1+e^{-a}}$ so that $\frac{df}{da} = f(1-f)$
- Therefore, $\frac{d\ell}{df}\frac{df}{da} = f - y$
- This is the cancellation that simplifies everything

Turning the Crank

- Gradient of the risk (recall that $s(\mathbf{x}\,;\,\mathbf{v}) = f(a(\mathbf{x}\,;\,\mathbf{v}))$):

$$\nabla L_T(\mathbf{v}) \;=\; \frac{1}{N}\sum_{n=1}^{N}[s(\mathbf{x}_n\,;\,\mathbf{v}) - y_n]\begin{bmatrix}1\\ \mathbf{x}_n\end{bmatrix}$$

- Hessian of the risk:

$$H_{L_T}(\mathbf{v}) \;=\; \frac{1}{N}\sum_{n=1}^{N} s(\mathbf{x}_n\,;\,\mathbf{v})\,[1 - s(\mathbf{x}_n\,;\,\mathbf{v})]\begin{bmatrix}1\\ \mathbf{x}_n\end{bmatrix}\begin{bmatrix}1 & \mathbf{x}_n\end{bmatrix}$$

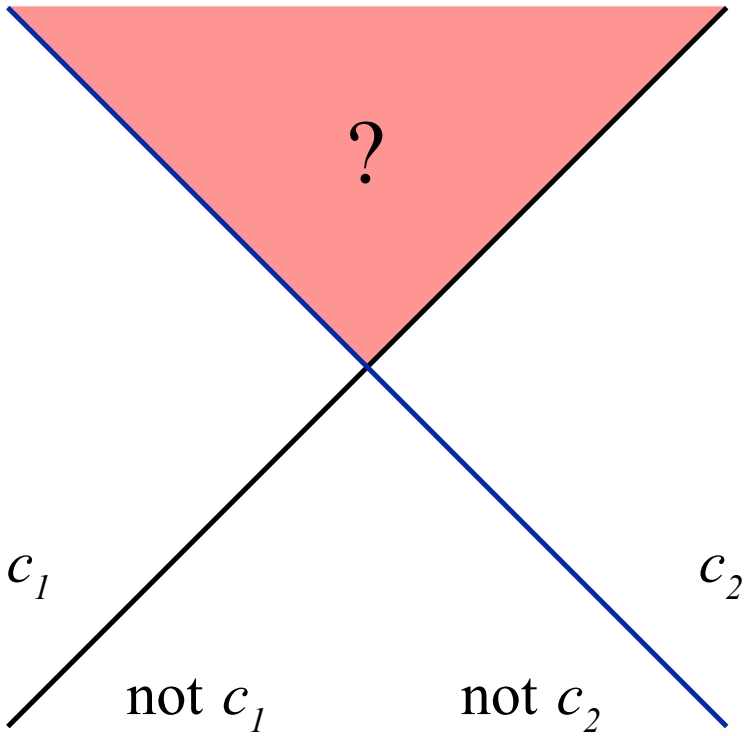
- Each term in the summation for H_{L_T} is an outer product
- This implies (easily) that H_{L_T} is positive semidefinite
- $L_T(\mathbf{v})$ **is a convex function**
- No need to check eigenvalues (See Appendix if you are curious)

Training

- $L_T(\mathbf{v})$ is convex in $\mathbf{v} \in \mathbb{R}^m$ with $m = d + 1$
- Use any gradient-based method to minimize
- When d is not too large, use Newton's method
- More efficient, problem-specific algorithms exist
- They capitalize on $L_T(\mathbf{v})$ being a sum of squares
- Typically, train with cross-entropy loss, test with 0-1 loss

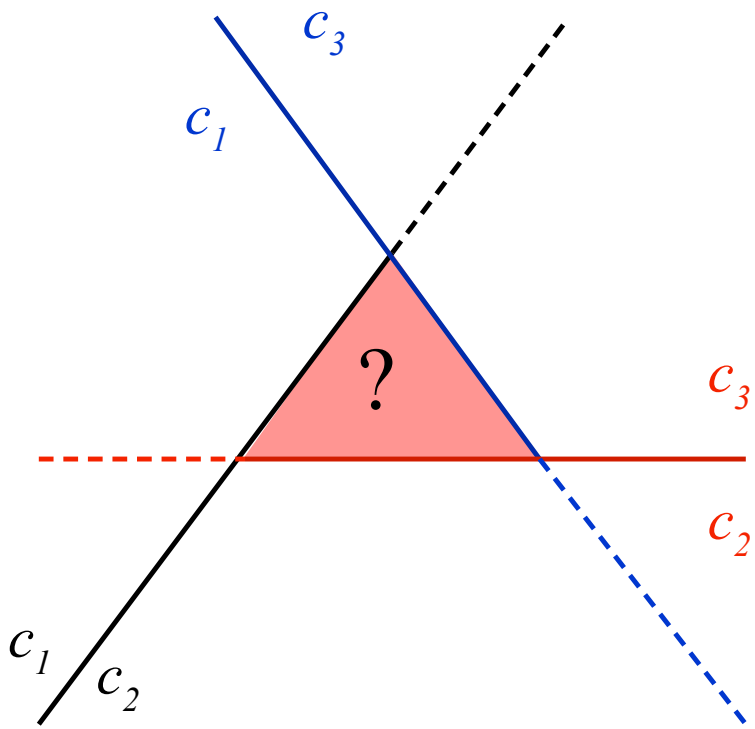
Multi-Class Linear Predictors

- Obvious approach 1: One-versus-rest
- Build $K - 1$ classifiers c_k versus not c_k
- Works for $K = 2$ but not for $K = 3$



Multi-Class Linear Predictors

- Obvious approach 2: One-versus-one
- Build $\binom{K}{2}$ classifiers c_i versus c_j
- Works for $K = 2$ but not for $K = 3$



A Symmetric View of the Binary **Score**

- Rename classes: 0 becomes 1 and 1 becomes 2
- Activation: $a = b + \mathbf{w}^T \mathbf{x}$
- Score for class 1: $s_1(a) = \frac{1}{1+e^{-a}}$
- Score for class 2: $s_2(a) = 1 - s_1(a) = s_1(-a)$
- More symmetrically, two activations:
  $a_1 = b + \mathbf{w}^T \mathbf{x}$, $a_2 = -b - \mathbf{w}^T \mathbf{x}$
- Note: $\frac{1}{1+e^{-a}} = \frac{e^{\frac{a}{2}}}{e^{\frac{a}{2}}} \frac{1}{1+e^{-a}} = \frac{e^{\frac{a}{2}}}{e^{\frac{a}{2}}+e^{-\frac{a}{2}}}$
- Score for class 1: $s_1 = s(\mathbf{a}_1) = \frac{e^{\frac{a_1}{2}}}{e^{\frac{a_1}{2}}+e^{-\frac{a_1}{2}}} = \frac{e^{\frac{a_1}{2}}}{e^{\frac{a_1}{2}}+e^{\frac{a_2}{2}}}$
- Score for class 2 (switch a_1 with a_2): $s_2 = s(\mathbf{a}_2) = \frac{e^{\frac{a_2}{2}}}{e^{\frac{a_1}{2}}+e^{\frac{a_2}{2}}}$
- Class with highest score wins

Exploiting Scalable Activations

- Score for class $k \in \{1, 2\}$: $s_k = \frac{e^{\frac{a_k}{2}}}{e^{\frac{a_1}{2}} + e^{\frac{a_2}{2}}}$
- Activations are freely scalable, so write $s_k = \frac{e^{a_k}}{e^{a_1} + e^{a_2}}$ instead
- Different function, same separating hyperplane
- This generalizes. Replace 2 classes with K

  $s_k(\mathbf{x}) = \frac{e^{a_k(\mathbf{x})}}{\sum_{j=1}^{K} e^{a_j(\mathbf{x})}}$ where $a_k(\mathbf{x}) = b_k + \mathbf{w}_k^T \mathbf{x}$
- Satisfies $\sum_{k=1}^{K} s_k(\mathbf{x}) = 1$
- Class with highest score wins: $\hat{y} = h(\mathbf{x}) \in \arg\max_k s_k(\mathbf{x})$
- This is the *Linear-Regression Multi-Class Classifier:* Compute k scores, each with parameters $\mathbf{v}_k = (b_k, \mathbf{w}_k)$, and pick the class with the highest score

The Soft-Max Function

$$s_k(\mathbf{x}) = \frac{e^{a_k(\mathbf{x})}}{\sum_{j=1}^{K} e^{a_j(\mathbf{x})}}$$

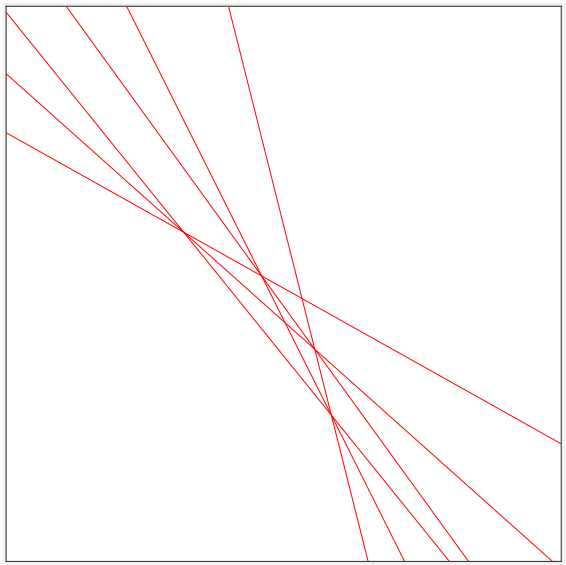
- $s_k(\mathbf{x}) > 0$ and $\sum_{k=1}^{K} s_k(\mathbf{x}) = 1$ for all $\mathbf{x}$
- If $a_i \gg a_j$ for $j \neq i$ then $\sum_{j=1}^{K} e^{a_j(\mathbf{x})} \approx e^{a_i(\mathbf{x})}$
- Therefore, $s_i \approx 1$ and $s_j \approx 0$ for $j \neq i$
- "Brings out the biggest:" *soft-max*
- More formally:

$$\lim_{\alpha \to \infty} \mathbf{a}^T \mathbf{s}(\alpha \mathbf{a}) = \max(\mathbf{a})$$

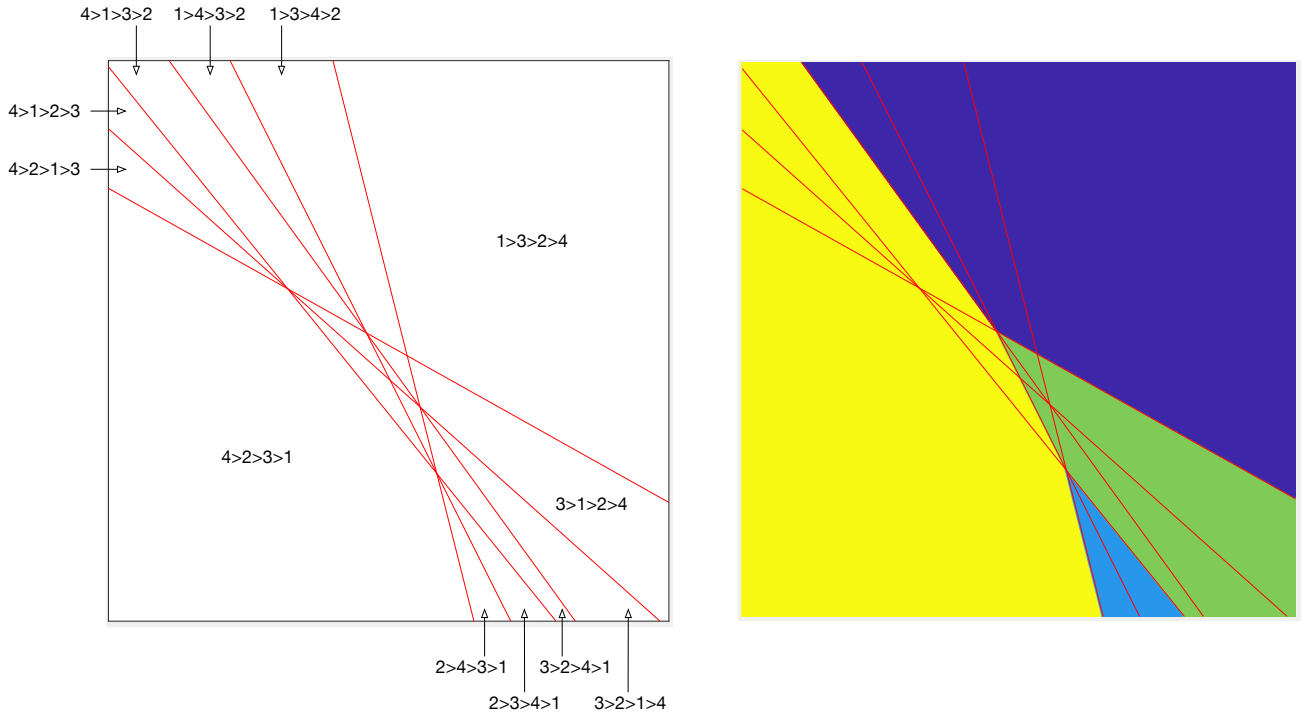
Geometry of Multi-ClassDecision Regions

- Separating hyperplane for classes $i, j \in \{1, \ldots, K\}$: $b_i + \mathbf{w}_i^T \mathbf{x} = b_j + \mathbf{w}_j^T \mathbf{x}$ (equal activations $\Rightarrow$ equal scores)
- Total of $M = \binom{K}{2}$ hyperplanes, just as in one-vs-one
- Example: $d = 2$, $K = 4 \Rightarrow$ 6 lines on the plane
- There are degeneracies ($M \times (d + 1)$ matrix of rank $K - 1$)
- Crossing a line switches two scores. Example:

$s_3 > s_2 > s_4 > s_1 \quad \rightarrow \quad s_3 > s_4 > s_2 > s_1$



Geometry of Decision Regions



- Crossing a line switches two scores:

  $s_4 > s_2 > s_3 > s_1 \quad \rightarrow \quad s_4 > s_3 > s_2 > s_1$

- When the *top two* scores switch, cross a boundary:

  $s_4 > s_1 > s_3 > s_2 \quad \rightarrow \quad s_1 > s_4 > s_3 > s_2$

- Decision regions are intersections of half-spaces $\Rightarrow$ convex

Multi-Class Cross-Entropy Loss

- Cross-entropy loss for $K = 2$:
  (remember that we renamed $Y = \{0, 1\}$ to $Y = \{1, 2\}$)

$$\ell(y, p) \stackrel{\text{def}}{=} \begin{cases} -\log p & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 2 \end{cases} = \begin{cases} -\log p_1 & \text{if } y = 1 \\ -\log p_2 & \text{if } y = 2 \end{cases}$$

- Same as $\ell(y, \mathbf{p}) = -\log p_y$
- But this is general!
- Can also write as follows: $\ell(y, \mathbf{p}) = -\sum_{k=1}^{K} q_k(y) \log p_k$
- $\mathbf{q}$ is the *one-hot encoding* of y
- Example: $K = 5$, then $y = 4$ is represented by $\mathbf{q} = [0, 0, 0, 1, 0]$

Convex Risk, Again

- Even with $K > 2$, the risk is a convex function of $\mathbf{v} = (b_1, \mathbf{w}_1, \ldots, b_K, \mathbf{w}_K) \in \mathbb{R}^m$ with $m = (d+1)K$
- Proof analogous to $K = 2$ case, just technically more involved
- Can still use gradient descent methods, including Newton