Linear Predictors

COMPSCI 371D — Machine Learning
Outline

1. Definitions and Properties
2. The Least-Squares Linear Regressor
3. The Logistic-Regression Classifier
4. Probabilities and the Geometry of Logistic Regression
5. The Logistic Function
6. The Cross-Entropy Loss
7. Multi-Class Linear Predictors
Definitions

• A linear *regressor* fits an affine function to the data
  \[ y \approx h(x) = b + w^T x \quad \text{for} \quad x \in \mathbb{R}^d \]
• A linear, binary *classifier* separates the two classes with a hyperplane in \( \mathbb{R}^d \)
• The actual data can be separated only if it is linearly separable (!)
• Multi-class classifiers separate any two classes with a hyperplane
• The resulting decision regions are convex and simply connected
Properties of Linear Predictors

• Linear Predictors...
  • ...have a very small $\mathcal{H}$ with $d + 1$ parameters (resist overfitting)
  • ... are trained by a convex optimization problem (global optimum)
  • ... are fast at inference time (and training is not too slow)
  • ... work well if the data is close to linearly separable
The Least-Squares Linear Regressor

- Déjà vu: Polynomial regression with $k = 1$
  \[ y \approx h_v(x) = b + w^T x \text{ for } x \in \mathbb{R}^d \]
- Parameter vector $v = \begin{bmatrix} b \\ w \end{bmatrix} \in \mathbb{R}^{d+1}$
  \[ \mathcal{H} = \mathbb{R}^m \text{ with } m = d + 1 \]
- “Least Squares:” $\ell(y, \hat{y}) = (y - \hat{y})^2$
- $\hat{v} = \arg \min_{v \in \mathbb{R}^m} L_T(v)$
- Risk $L_T(v) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h_v(x_n))$
- We know how to solve this
Linear Regression Example

- Left: All of Ames. Residual $\sqrt{\text{Risk}}$: $55,800$
- Right: One Neighborhood. Residual $\sqrt{\text{Risk}}$: $23,600$
- Left, yellow: Ignore two largest homes
Binary Classification by Logistic Regression

\[ Y = \{ c_0, c_1 \} \]

- Multi-class case later
- The logistic-regression classifier is a classifier!
- A linear classifier implemented through regression
- The logistic is a particular function
Score-Based Classifiers

\[ Y = \{ c_0, c_1 \} \]

- Think of \( c_0, c_1 \) as numbers: \( Y = \{0, 1\} \)
- We saw the idea of level sets:
  Regress a score function \( s \) such that
  \( s \) is large where \( y = 1 \), small where \( y = 0 \)
- Threshold \( s \) to obtain a classifier:
  \[ h(x) = \begin{cases} 
  c_0 & \text{if } s(x) \leq \text{threshold} \\
  c_1 & \text{otherwise.} 
\end{cases} \]
- A linear classifier implemented through regression
Idea 1

- $s(x) = b + w^T x$

- Not so good!
- A line does not approximate a step well
- Why not fit a step function?
- NP-hard unless the data is separable
Idea 2

- How about a “soft step?”
- The *logistic function*

\[
f(x) \overset{\text{def}}{=} \frac{1}{1 + e^{-x}}
\]

- If a true step moves, the loss does not change until a data point flips label
- If the logistic function moves, the loss changes gradually
- We have a gradient!
- The optimization problem is no longer combinatorial
What is a Logistic Function in $d$ Dimensions?

- We want a *linear* classifier
- The level crossing must be a hyperplane
- Level crossing: Solution to $s(x) = 1/2$
- Shape of the crossing depends on $s$
- Compose an affine $a(x) = c + u^T x$  
  ...with a monotonic $f(a)$ that crosses $1/2$
  $s(x) = f(a(x)) = f(c + u^T x)$
- Then, if $f(\alpha) = 1/2$, the equation $s(x) = 1/2$
  is the same as $c + u^T x = \alpha$
- A hyperplane!
- Let $f$ be the logistic function
Example

- Gold line: Regression problem $\mathbb{R} \rightarrow \mathbb{R}$
- Black line: Classification problem $\mathbb{R}^2 \rightarrow \mathbb{R}$
  (result of running a logistic-regression classifier)
- Labels: Good (red squares, $y = 1$) or poor quality (blue circles, $y = 0$) homes
- All that matters is how far a point is from the black line
A Probabilistic Interpretation

• All that matters is how far a point is from the black line
• \( s(x) = f(\Delta(x)) \) where \( \Delta \) is a signed distance
• We could interpret the score \( s(x) \) as “the probability that \( y = 1: \)  \( f(\Delta(x)) = P[y = 1] \)
• (…or as “1 – the probability that \( y = 0 \)”)
  \[
  \lim_{\Delta \to -\infty} P[y = 1] = 0 \quad \lim_{\Delta \to \infty} P[y = 1] = 1 \\
  \Delta = 0 \Rightarrow P[y = 1] = 1/2 \quad \text{(just like the logistic function)}
  \]
Ingredients for the Regression Part

- Determine the distance $\Delta$ of a point $\mathbf{x} \in X$ from a hyperplane $\chi$, and the side of $\chi$ on which the point is on (Geometry: *affine functions* as unscaled, signed distances)
- Specify a monotonically increasing function that turns $\Delta$ into a probability (Choice based on convenience: the *logistic function*)
- Define a loss function $\ell(y, \hat{y})$ such that the minimum risk yields the optimal classifier (Ditto, matches function in previous bullet to obtain a *convex risk*: the *cross-entropy loss*)
Normal to a Hyperplane

- Hyperplane $\chi$: $b + \mathbf{w}^T \mathbf{x} = 0$ (w.l.o.g. $b \leq 0$)
  $\mathbf{a}_1, \mathbf{a}_2 \in \chi \Rightarrow \mathbf{c} = \mathbf{a}_1 - \mathbf{a}_2$ parallel to $\chi$
- Subtract $b + \mathbf{w}^T \mathbf{a}_1 = 0$ from $b + \mathbf{w}^T \mathbf{a}_2 = 0$
- Obtain $\mathbf{w}^T \mathbf{c} = 0$ for any $\mathbf{a}_1, \mathbf{a}_2 \in \chi$
- $\mathbf{w}$ is perpendicular to $\chi$
Distance of a Hyperplane from the Origin

- Unit-norm version of \( \mathbf{w} \): \( \mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \)
- Rewrite \( \chi \): \( b + \mathbf{w}^T \mathbf{x} = 0 \) (w.l.o.g. \( b \leq 0 \)) as \( \mathbf{n}^T \mathbf{x} = \beta \) where \( \beta = -\frac{b}{\|\mathbf{w}\|} \geq 0 \)
- Line along \( \mathbf{n} \): \( \mathbf{x} = \alpha \mathbf{n} \) for \( \alpha \in \mathbb{R} \) (parametric form) \( \alpha \) is the distance from the origin
- Replace into eq. for \( \chi \): \( \alpha \mathbf{n}^T \mathbf{n} = \beta \) that is, \( \alpha = \beta \geq 0 \)
- In particular, \( \mathbf{x}_0 = \beta \mathbf{n} \)
- \( \beta \) is the distance of \( \chi \) from the origin
Signed Distance of a Point from a Hyperplane

\[ \mathbf{n}^T \mathbf{x} = \beta \] where \( \beta = -\frac{b}{\|\mathbf{w}\|} \geq 0 \) and \( \mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \)

\( \mathbf{x}_0 = \beta \mathbf{n} \)

- In one half-space, \( \mathbf{n}^T \mathbf{x} \geq \beta \)
- Distance of \( \mathbf{x} \) from \( \chi \) is \( \mathbf{n}^T \mathbf{x} - \beta \geq 0 \)
- In other half-space, \( \mathbf{n}^T \mathbf{x} \leq \beta \)
- Distance of \( \mathbf{x} \) from \( \chi \) is \( \beta - \mathbf{n}^T \mathbf{x} \geq 0 \)
- On decision boundary, \( \mathbf{n}^T \mathbf{x} = \beta \)
- \( \mathbf{n}^T \mathbf{x} - \beta \) is the signed distance of \( \mathbf{x}_0 \) from the hyperplane
Summary

If \( \mathbf{w} \) is nonzero (which it has to be), the distance of \( \chi \) from the origin is

\[
\beta \overset{\text{def}}{=} \frac{|b|}{\|\mathbf{w}\|}
\]

(a nonnegative number) and the quantity

\[
\Delta(\mathbf{x}) \overset{\text{def}}{=} \frac{b + \mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|}
\]

is the \textit{signed distance} of point \( \mathbf{x} \in \mathcal{X} \) from hyperplane \( \chi \). Specifically, the distance of \( \mathbf{x} \) from \( \chi \) is \( |\Delta(\mathbf{x})| \), and \( \Delta(\mathbf{x}) \) is nonnegative if and only if \( \mathbf{x} \) is on the side of \( \chi \) pointed to by \( \mathbf{w} \). Let us call that side the \textit{positive half-space} of \( \chi \).
Ingredient 2: The Logistic Function

- Want to make the score of $x$ be only a function of $\Delta(x)$
- Given $\Delta_0$, all points such that $\Delta(x) = \Delta_0$ have the same score
- Score $s(x) = f(\Delta(x))$
- How to pick $f$?
- $\lim_{\Delta \to -\infty} f(\Delta) = 0$ \hspace{1cm} $f(0) = 1/2$ \hspace{1cm} $\lim_{\Delta \to \infty} f(\Delta) = 1$
- Logistic function: $f(\Delta) \overset{\text{def}}{=} \frac{1}{1+e^{-\Delta}}$
The Logistic Function

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

![Graph of the logistic function]

• Scale-free: Why not \( \frac{1}{1+e^{-\Delta/c}} \) ?

• Can use both \( c \) and \( \Delta(x) \stackrel{\text{def}}{=} \frac{b+w^T x}{\|w\|} \)

... or more simply use no \( c \) but use \( a(x) \stackrel{\text{def}}{=} b + w^T x \)

• The affine function takes care of scale implicitly

• Score: \( s(x) \stackrel{\text{def}}{=} f(a(x)) = \frac{1}{1+e^{-b-w^T x}} \)

• Write \( s(x; b, w) \) to remind us of dependence
Optimize the Regressor, not the Classifier

- We would like something similar to
  \[ \ell_{0-1}(y, \hat{y}) = \begin{cases} 
  0 & \text{if } y = \hat{y} \\
  1 & \text{otherwise} 
\end{cases} \]
- However, \( \ell_{0-1} \) is not differentiable
- Use the score \( p = s(x; b, w) \) instead of \( \hat{y} \):
  - \( \hat{y} \in \{0, 1\} \) while \( p \in [0, 1] \)
  - Instead of measuring the loss on \( \hat{y} = h(x) \), we measure it on \( p = s(x; b, w) \approx \hat{y} \)
- We still need a different \( \ell(y, p) \) for differentiability
The Cross-Entropy Loss

Differentiability, Again

• We want $\ell(y, p)$ to be differentiable in $p$
• Since $p$ is differentiable in $v = (b, w)$, so then will be $\ell$
• Why do we insist on differentiability, again?
• Risk: $L_T(b, w) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, s(x_n ; b, w))$
• Use a gradient method (steepest descent, Newton, ...)
• We have not yet chosen the specific form of $\ell$
• We can make $L_T(b, w)$ a differentiable and convex function of $v = (b, w)$ by a suitable choice of $\ell$
The Cross-Entropy Loss

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

- 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:

\[ L_T(v) = L_T(\ell(s(a(v))), \text{ so } \nabla L_T = \frac{dL_T}{d\ell} \frac{d\ell}{ds} \frac{ds}{da} \nabla a \]

\[ \ell = -y \log s - (1 - y) \log(1 - s) \text{ so that } \frac{d\ell}{ds} = \frac{s - y}{s(1 - s)} \]

\[ s(a) = \frac{1}{1 + e^{-a}} \text{ so that } \frac{ds}{da} = s(1 - s) \]

• Therefore, \[ \frac{d\ell}{ds} \frac{ds}{da} = s - y \]
• This is the cancellation that simplifies everything
Turning the Crank

• Gradient of the risk:

\[ \nabla L_T(v) = \frac{1}{N} \sum_{n=1}^{N} [s(x_n ; v) - y_n] \left[ \begin{array}{c} 1 \\ x_n \end{array} \right] \]

• Hessian of the risk:

\[ H_{L_T}(v) = \frac{1}{N} \sum_{n=1}^{N} s(x_n ; v) [1 - s(x_n ; v)] \left[ \begin{array}{cc} 1 \\ x_n \end{array} \right] \left[ \begin{array}{c} 1 \\ x_n \end{array} \right] \]

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

• $L_T(v)$ is convex in $v \in \mathbb{R}^m$ with $m = d + 1$
• Use any gradient-based method to minimize
• When $d$ is not too large, use Newton’s method (homework!)
• More efficient, problem-specific algorithms exist
• They capitalize on $L_T(v)$ 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 \( K = 3 \)

\[
\begin{align*}
    c_1 & \quad \text{not } c_1 \\
    \text{not } c_1 & \quad c_2 \\
    \text{not } c_2 & \quad \text{not } c_2
\end{align*}
\]
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 1, 2 rather than 0, 1
- Activation: \( a = b + w^T x \)
- 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:
  \( a_1 = b + w^T x, \ a_2 = -b - w^T x \)
  \( \text{Note: } \frac{1}{1 + e^{-a}} = \frac{e^a}{e^a + 1} = \frac{e^a}{e^a + e^{-a/2}} \)
  \( s_1 = s(a_1) = \frac{e^{a_1/2}}{e^{a_1/2} + e^{-a_1/2}} = \frac{e^{a_1/2}}{e^{a_2/2} + e^{a_1/2}} \)
  \( s_2 = s(a_2) = \frac{e^{a_2/2}}{e^{a_1/2} + e^{a_2/2}} \)
  Class with highest score wins
Exploiting Scalable Activations

• Score for class \( k \in \{1, 2\} \): \( s_k = \frac{e^{a_k}}{e^{a_1} + e^{a_2}} \)

• 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 \( K \)

\[
s_k(x) = \frac{e^{a_k(x)}}{\sum_{j=1}^{K} e^{a_j(x)}} \text{ where } a_k(x) = b_k + w_k^T x
\]

• Satisfies \( \sum_{j=1}^{K} s_k(x) = 1 \)

• Class with highest score wins: \( \hat{y} = h(x) \in \arg \max_k s_k(x) \)

• This is the Linear-Regression Multi-Class Classifier
The Soft-Max Function

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

- \( s_k(x) > 0 \) and \( \sum_{k=1}^{K} s_k(x) = 1 \) for all \( x \)
- If \( a_i \gg a_j \) for \( j \neq i \) then \( \sum_{j=1}^{K} e^{a_j(x)} \approx e^{a_i(x)} \)
- Therefore, \( s_i \approx 1 \) and \( s_j \approx 0 \) for \( j \neq i \)
- “Brings out the biggest:” soft-max
- Collect into vectors: \( \mathbf{a} = (a_1, \ldots, a_K) \), \( \mathbf{s} = (s_1, \ldots, s_K) \)

\[ \mathbf{x} \in \mathbb{R}^d \rightarrow \mathbf{a} \in \mathbb{R}^K \rightarrow \mathbf{s} \in \mathbb{R}^K \]

\[ \mathbf{s}(\mathbf{a}(\mathbf{x})) = \frac{e^{\mathbf{a}(\mathbf{x})}}{\mathbf{1}^T e^{\mathbf{a}(\mathbf{x})}} \]

\[ \lim_{\alpha \to \infty} \mathbf{a}^T \mathbf{s}(\alpha \mathbf{a}) = \max(\mathbf{a}) \]
Geometry of Multi-Class Decision 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. Example:
  \[ s_3 > s_2 > s_4 > s_1 \rightarrow s_3 > s_4 > s_2 > s_1 \]
- When the *top two* scores switch, we cross a decision boundary. Example:
  \[ s_3 > s_2 > s_4 > s_1 \rightarrow s_2 > s_3 > s_4 > s_1 \]
- 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 $Y = \{1, 2\}$)

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

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