Key Concepts:

- Logistic Regression
- Regularization
- Cross validation

note: we are still talking about binary classification (with \{0, 1\} labels)
So far: Turned a real score $\mathbf{w}^T \mathbf{x} = w_0 \cdot 1 + \sum_{i=1}^{d} w_i \cdot x_i$ to binary decision by thresholding.

Alternative: Model the probability $P(y = 1|\mathbf{x})$.

Need to squash $\mathbf{w}^T \mathbf{x}$ into $[0, 1]$, $p(y = 1|\mathbf{x}) = f(\mathbf{w}^T \mathbf{x})$.

What about $P(y = -1|\mathbf{w})$? $P(y = -1|\mathbf{w}) = 1 - P(y = 1|\mathbf{w}) = 1 - f(\mathbf{w}^T \mathbf{x})$

How to chose label? Pick the most probable (when shouldn’t you do that?).

Benefits:
- Models uncertainty (in a limited manor)
- Can use probability for decision making.
- Can use probabilistic objective (ML/MAP).
Useful squashing function: sigmoid or logistic function

\[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]

- Smooth function.
- Monotonic increasing.
- \( \sigma(0) = 0.5 \)
- \( \sigma(z) \xrightarrow{z \to -\infty} 0 \), \( \sigma(z) \xrightarrow{z \to \infty} 1 \)
Let's look at how modifying $w$ changes the shape of the function

1D example:

$$y = \sigma(w_1 x + w_0)$$

$w_0 = 0, w_1 = 1$

$w_0 = 0, w_1 = 0.5$

$w_0 = -2, w_1 = 1$

The magnitude of $w_{[1:]}$ decides the slope.

It can be seen as a smooth alternative to the step function.
What is the decision boundary for logistic regression?

\[ p(y = 1|x, w) = \sigma(w^T x) \geq 0.5 \Rightarrow w^T x \geq 0 \]

Decision boundary: \[ w^T x = w_0 + \sum_{j=1}^{d} w_j x_j = 0. \]

Logistic regression has a linear decision boundary.

The decision boundary is invariant to scaling but the probability isn’t.
- When we have a d-dim input $\mathbf{x} \in \mathbb{R}^d$
- How should we learn the weights $\mathbf{w} = (w_0, w_1, \cdots, w_d)$?
- We have a probabilistic model
- Let’s use maximum likelihood
Assume $y \in \{0, 1\}$, we can write the probability distribution of each of our training points $p(y^{(1)}, \cdots, y^{(N)}|x^{(1)}, \cdots x^{(N)}; w)$.

Assuming that the training examples are sampled IID: independent and identically distributed, we can write the *likelihood function*:

$$L(w) = p(y^{(1)}, \cdots, y^{(N)}|x^{(1)}, \cdots x^{(N)}; w) = \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}; w)$$

We can write each probability as (will be useful later):

$$p(y^{(i)}|x^{(i)}; w) = p(y = 1|x^{(i)}; w)y^{(i)} p(y = 0|x^{(i)}; w)^{1-y^{(i)}}$$

$$= p(y = 1|x^{(i)}; w)y^{(i)} \left(1 - p(y = 1|x^{(i)}; w)\right)^{1-y^{(i)}}$$

We can learn the model by maximizing the likelihood

$$\max_w L(w) = \max_w \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}; w)$$

Easier to maximize the log likelihood $\log L(w)$. 

---

CSC411-Lec4
Logistic regression

Regularization

Validation

Optimization

\[ L(w) = \prod_{i=1}^{N} p(y^{(i)} | x^{(i)}) \quad \text{(likelihood)} \]

\[ = \prod_{i=1}^{N} \left( 1 - p(y = 1 | x^{(i)}) \right)^{1-y^{(i)}} p(y = 1 | x^{(i)})^{y^{(i)}} \]

- We can convert the maximization problem into minimization the **negative log-likelihood (NLL)**:

\[ L_{\log}(w) = - \log L(w) = - \sum_{i=1}^{N} \log p(y^{(i)} | x^{(i)}; w) \]

\[ L_{\log}(w) = - \log L(w) \]

\[ = - \sum_{i=1}^{N} y^{(i)} \log(p(y = 1 | x^{(i)}, w)) - \sum_{i=1}^{N} (1 - y^{(i)}) \log p(y = 0 | x^{(i)}; w) \]

- Is there a closed form solution?
Optimization

\[
\min L(w) = \min_w \left\{ -\sum_{i=1}^{N} y^{(i)} \log p(y = 1 | x^{(i)}, w) - \sum_{i=1}^{N} (1 - y^{(i)}) \log(1 - p(y = 1 | x^{(i)}, w)) \right\}
\]

- Gradient descent: iterate and at each iteration compute steepest direction towards optimum, move in that direction, step-size \( \lambda \)

\[
w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \frac{\partial L(w)}{\partial w_j}
\]

- You can write this in vector form

\[
\nabla L(w) = \left[ \frac{\partial L(w)}{\partial w_0}, \ldots, \frac{\partial L(w)}{\partial w_k} \right]^T \quad w^{(t+1)} \leftarrow w^{(t)} - \lambda \nabla_w L(w^{(t)})
\]

- But where is \( w \)?

\[
p(y = 1 | x) = \frac{1}{1 + \exp(-w^T x)}, \quad p(y = 0 | x) = \frac{\exp(-w^T x)}{1 + \exp(-w^T x)}
\]
The loss is

\[ L_{\text{log-loss}}(w) = -\sum_{i=1}^{N} y^{(i)} \log p(y = 1|x^{(i)}, w) - \sum_{i=1}^{N} (1 - y^{(i)}) \log p(y = 0|x^{(i)}, w) \]

where the probabilities are

\[ p(y = 1|x, w) = \frac{1}{1 + \exp(-z)} \quad p(y = 0|x, w) = \frac{\exp(-z)}{1 + \exp(-z)} = \frac{1}{1 + \exp(z)} \]

and \( z = w^T x \)

We can simplify

\[
L(w)_{\text{log-loss}} = \sum_{i} y^{(i)} \log(1 + \exp(-z^{(i)})) + \sum_{i} (1 - y^{(i)})z^{(i)} + \sum_{i} (1 - y^{(i)}) \log(1 + \exp(-z^{(i)}))
\]

\[
= \sum_{i} \log(1 + \exp(-z^{(i)})) + \sum_{i} (1 - y^{(i)})z^{(i)}
\]

Now it’s easy to take derivatives
\[ L(w) = \sum_i (1 - y^{(i)}) z^{(i)} + \sum_i \log(1 + \exp(-z^{(i)})) \]

- Now it’s easy to take derivatives
- Remember \( z = w^T x \Rightarrow \frac{\partial z}{\partial w_j} = x_j \)
  \[
  \frac{\partial \ell}{\partial w_j} = \frac{\partial \ell}{\partial z} \cdot \frac{\partial z}{\partial w_j} = \sum_i x_j^{(i)} \left( 1 - y^{(i)} - \frac{\exp(-z^{(i)})}{1 + \exp(-z^{(i)})} \right) = \sum_i x_j^{(i)} \left( \frac{1}{1 + \exp(-z^{(i)})} - y^{(i)} \right)
  \]
- What’s \( x_j^{(i)} \)? The \( j \)-th dimension of the \( i \)-th training example \( x^{(i)} \)
- And simplifying
  \[
  \frac{\partial \ell}{\partial w_j} = \sum_i x_j^{(i)} \left( p(y = 1|x^{(i)}; w) - y^{(i)} \right)
  \]
- Don’t get confused with indices: \( j \) for the weight that we are updating and \( i \) for the training example
Putting it all together (plugging the update into gradient descent): Gradient descent for logistic regression:

\[ w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \sum_i x_j^{(i)} \left( p(y = 1|x^{(i)}; w) - y^{(i)} \right) \]

where:

\[ p(y = 1|x^{(i)}; w) = \frac{1}{1 + \exp(-w^T x)} \]

This is all there is to learning in logistic regression. Simple, huh?
We are optimizing $\sum_i (1 - y^{(i)})z^{(i)} + \sum_i \log(1 + \exp(-z^{(i)}))$.

We can forget the probabilistic interpretation and just think about a surrogate loss function

$$\ell(y, \hat{y}) = (1 - y)\hat{y} + \log(1 + \exp(-\hat{y})) = \begin{cases} 
\log(1 + \exp(-\hat{y})), & y = 1, \\
\log(1 + \exp(\hat{y})), & y = 0,
\end{cases}$$

It is convex, so gradient descent converges to global minimum.
Logistic Regression vs Least Squares Regression:

If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being “too correct” (tilts away from outliers)
Regularization:

- We can also look at

\[ p(w|\{y\}, \{x\}) \propto p(\{y\}|\{x\}, w) p(w) \]

with \( \{y\} = (y^{(1)}, \ldots, y^{(N)}) \), and \( \{x\} = (x^{(1)}, \ldots, x^{(N)}) \)

- We can define priors on parameters \( w \)

- This is a form of regularization

- Helps avoid large weights and overfitting

\[
\max_w \log \left[ p(w) \prod_i p(y^{(i)}|x^{(i)}, w) \right]
\]

- This is called maximum-a-posteriori estimation (MAP)?

- What’s \( p(w) \)?
For example, define prior: normal distribution, zero mean and identity covariance 
\[ p(w) \propto \mathcal{N}(0, \alpha^{-1}I) \] (best to exclude \( w_0 \))

This prior pushes parameters towards zero (why is this a good idea?)

Equivalent to \( L_2 \) regularization

Including this prior the new gradient is

\[
\begin{align*}
  w_j^{(t+1)} &\leftarrow w_j^{(t)} - \lambda \frac{\partial L(w)}{\partial w_j} - \lambda \alpha w_j^{(t)}
\end{align*}
\]

where \( t \) here refers to iteration of the gradient descent

The parameter \( \alpha \) is the importance of the regularization, and it’s a hyper-parameter

How do we decide the best value of \( \alpha \) (or a hyper-parameter in general)?
Logistic regression

Example

MNIST digit data-set: 60,000 training $28 \times 28$ digit images, 10,000 test images. Need to classify as 0-9.

Only take zero and ones - binary classification.
Train logistic regression with various regularization parameters-

![Plot showing test and train accuracy against regularization parameter](image)
How do the classifiers look?

(doesn’t overfit that much, still great on test)
Tuning hyper-parameters:

- Never use test data for tuning the hyper-parameters
- We can divide the set of training examples into two disjoint sets: training and validation.
- Use the first set (i.e., training) to estimate the weights $w$ for different values of $\alpha$.
- Use the second set (i.e., validation) to estimate the best $\alpha$, by evaluating how well the classifier does on this second set.
- This tests how well it generalizes to unseen data.
- Trade-off: Large validation set → less training data to use.
- Trade-off: Small validation set → less accurate estimation.
- Can overfit on the validation set!
- **Leave-p-out cross-validation:**
  - We use $p$ observations as the validation set and the remaining observations as the training set.
  - This is repeated on all ways to cut the original training set.
  - It requires $\binom{n}{p}$ for a set of $n$ examples

- **Leave-1-out cross-validation:** When $p = 1$, does not have this problem

- **k-fold cross-validation:**
  - The training set is randomly partitioned into $k$ equal size subsamples.
  - Of the $k$ subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data.
  - The cross-validation process is then repeated $k$ times (the folds).
  - The $k$ results from the folds can then be averaged (or otherwise combined) to produce a single estimate
Train your model:

- Leave-one-out cross-validation:
- k-fold cross-validation:
Logistic Regression wrap-up

Pros:

- Probabilistic view of class predictions
- Quick to train, convex loss
- Fast at classification
- Good accuracy for many simple data sets
- Resistant to overfitting (Rule of thumb: $\#data \geq 10 \cdot \#features$)
- Can interpret model coefficients as indicators of feature importance

Cons:

- Linear decision boundary (too simple for more complex problems?)
- Very simple model of the conditional probabilities