CSC 2515: Lecture 02: Classification

Rich Zemel & Raquel Urtasun

University of Toronto

Sep 24, 2015
Today (first half)

- Linear Classification
- Key Concepts:
  - Classification as regression
  - Decision boundary
  - Loss functions
  - Metrics to evaluate classification
Classification vs Regression

- We are interested in mapping the input $x \in \mathcal{X}$ to a label $t \in \mathcal{Y}$
- In regression typically $\mathcal{Y} = \mathbb{R}$
- Now it's a category
- Examples?
Examples of Classification

What digit is this? How can I predict this? What are my input features?

Is this a dog? How can I predict this? What are my input features?

What about this one? Am I going to pass the exam? How can I predict this? What are my input features?

Do I have diabetes? How can I predict this? What are my input features?
Classification as Regression

- Can we do this task using what we have learned in previous lectures?
- Simple hack: Ignore that the input is categorical!
- Suppose we have a binary problem, $t \in \{-1, 1\}$
- Assuming the standard model used for regression
  \[ y = f(x, w) = w^T x \]

- How can we obtain $w$?
- Use least squares, $w = (X^T X)^{-1} X^T t$. How is $X$ computed? and $t$?
- Which loss are we minimizing? Does it make sense?
  \[ \ell_{\text{square}}(w, t) = \frac{1}{N} \sum_{n=1}^{N} \left( t^{(n)} - w^T x^{(n)} \right)^2 \]

- How do I compute a label for a new example? Let’s see an example
Classification as Regression

A 1D example:
\[ y \pm 1 \]
\[ w_0 + w^T x \]
\[ \hat{y} = -1 \quad \hat{y} = +1 \]
Our classifier has the form

\[ f(x, w) = w_0 + w^T x \]

A reasonable decision rule is

\[ y = \begin{cases} 
1 & \text{if } f(x, w) \geq 0 \\
-1 & \text{otherwise}
\end{cases} \]

How can I mathematically write this rule?

\[ y = \text{sign}(w_0 + w^T x) \]

How does this function look like?
How can I mathematically write this rule?

\[ y = \text{sign}(w_0 + w^T x) \]

This specifies a linear classifier: it has a linear boundary (hyperplane)

\[ w_0 + w^T x = 0 \]

which separates the space into two "half-spaces"
Example in 1D

- The linear classifier has a linear boundary (hyperplane)
  \[ w_0 + w^T x = 0 \]
  which separates the space into two "half-spaces"
- In 1D this is simply a threshold
The linear classifier has a linear boundary (hyperplane)

\[ w_0 + w^T x = 0 \]

which separates the space into two "half-spaces"

In 2D this is a line
Example in 3D

- The **linear classifier** has a linear boundary (hyperplane)

\[ w_0 + w^T x = 0 \]

which separates the space into two "half-spaces"

- In 3D this is a plane
- What about higher-dimensional spaces?
\( \mathbf{w}^T \mathbf{x} = 0 \) a line passing though the origin and orthogonal to \( \mathbf{w} \)

\( \mathbf{w}^T \mathbf{x} + w_0 = 0 \) shifts it by \( w_0 \)

Figure from G. Shakhnarovich
Learning consists in estimating a "good" decision boundary.

We need to find \( w \) (direction) and \( w_0 \) (location) of the boundary.

What does "good" mean?

Is this boundary good?

We need a criteria that tell us how to select the parameters.

Do you know any?
Classifying using a linear decision boundary reduces the data dimension to 1

\[ y(x) = \text{sign}(w_0 + w^T x) \]

What is the cost of being wrong?

**Loss function:** \( L(y, t) \) is the loss incurred for predicting \( y \) when correct answer is \( t \)

For medical diagnosis: For a diabetes screening test is it better to have false positives or false negatives?

For movie ratings: The "truth" is that Alice thinks E.T. is worthy of a 4. How bad is it to predict a 5? How about a 2?
A possible loss to minimize is the zero/one loss

\[ L(y(x), t) = \begin{cases} 0 & \text{if } y(x) = t \\ 1 & \text{if } y(x) \neq t \end{cases} \]

Is this minimization easy to do? why?
Other Loss functions

- **Zero/one loss** for a classifier

  \[
  L_{0-1}(y(x), t) = \begin{cases} 
  0 & \text{if } y(x) = t \\
  1 & \text{if } y(x) \neq t
  \end{cases}
  \]

- **Asymmetric Binary Loss**

  \[
  L_{ABL}(y(x), t) = \begin{cases} 
  \alpha & \text{if } y(x) = 1 \land t = 0 \\
  \beta & \text{if } y(x) = 0 \land t = 1 \\
  0 & \text{if } y(x) = t
  \end{cases}
  \]

- **Squared (quadratic) loss**

  \[
  L_{squared}(y(x), t) = (t - y(x))^2
  \]

- **Absolute Error**

  \[
  L_{quadratic}(y(x), t) = |t - y(x)|
  \]
What if the movie predictions are used for rankings? Now the predicted ratings don’t matter, just the order that they imply.

In what order does Alice prefer E.T., Amelie and Titanic?

Possibilities:

- 0-1 loss on the winner
- Permutation distance
- Accuracy of top K movies.
Can we always separate the classes?

- If we can separate the classes, the problem is linearly separable.
Causes of non perfect separation:

- Model is too simple
- Noise in the inputs (i.e., data attributes)
- Simple features that do not account for all variations
- Errors in data targets (miss labelings)

Should we make the model complex enough to have perfect separation in the training data?
How to evaluate how good my classifier is?

- **Precision**: is the fraction of retrieved instances that are relevant
  \[ P = \frac{TP}{TP + FP} \]

- **Recall**: is the fraction of relevant instances that are retrieved
  \[ R = \frac{TP}{TP + FN} \]

- **F1 score**: harmonic mean of precision and recall
  \[ F1 = 2 \frac{P \cdot R}{P + R} \]
More on Metrics

How to evaluate how good my classifier is?

- **Precision**: is the fraction of retrieved instances that are relevant
- **Recall**: is the fraction of relevant instances that are retrieved
- **Precision Recall Curve**

![Precision Recall Curve](image)

- **Average Precision (AP)**: mean under the curve
Key Concepts:

- Logistic Regression
- Regularization
- Cross validation
Logistic Regression

- An alternative: replace the \( \text{sign}(\cdot) \) with the sigmoid or logistic function
- We assumed a particular functional form: sigmoid applied to a linear function of the data

\[
y(x) = \sigma (w^T x + w_0)
\]

where the sigmoid is defined as

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

- The output is a smooth function of the inputs and the weights
Logistic Regression

- We assumed a particular functional form: sigmoid applied to a linear function of the data
  \[ y(x) = \sigma(w^T x + w_0) \]
  
  where the sigmoid is defined as
  \[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]

- One parameter per data dimension (feature) and the bias
- Features can be discrete or continuous
- Output of the model: value \( y \in [0, 1] \)
- This allows for gradient-based learning of the parameters: smoothed version of the \( \text{sign}(\cdot) \)
Let’s look at how modifying $\mathbf{w}$ changes the function shape

1D example:

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

$w_0 = -2, w_1 = 1$  
$w_0 = 0, w_1 = 1$  
$w_0 = 0, w_1 = 0.5$

Demo
Probabilistic Interpretation

- If we have a value between 0 and 1, let’s use it to model the posterior
  \[ p(C = 0 | x) = \sigma(w^T x + w_0) \quad \text{with} \quad \sigma(z) = \frac{1}{1 + \exp(-z)} \]

- Substituting we have
  \[ p(C = 0 | x) = \frac{1}{1 + \exp(-w^T x - w_0)} \]

- Suppose we have two classes, how can I compute \( p(C = 1 | x) \)?

- Use the marginalization property of probability
  \[ p(C = 1 | x) + p(C = 0 | x) = 1 \]

- Thus (show matlab)
  \[ p(C = 1 | x) = 1 - \frac{1}{1 + \exp(-w^T x - w_0)} = \frac{\exp(-w^T x - w_0)}{1 + \exp(-w^T x - w_0)} \]
Conditional likelihood

- Assume $t \in \{0, 1\}$, we can write the probability distribution of each of our training points $p(t^{(1)}, \ldots, t^{(N)}|x^{(1)}, \ldots x^{(N)})$

- Assuming that the training examples are sampled IID: independent and identically distributed

$$p(t^{(1)}, \ldots, t^{(N)}|x^{(1)}, \ldots x^{(N)}) = \prod_{i=1}^{N} p(t^{(i)}|x^{(i)})$$

- We can write each probability as

$$p(t^{(i)}|x^{(i)}) = p(C = 1|x^{(i)}) t^{(i)} p(C = 0|x^{(i)})^{1-t^{(i)}}$$

$$= \left(1 - p(C = 0|x^{(i)})\right)^{t^{(i)}} p(C = 0|x^{(i)})^{1-t^{(i)}}$$

- We might want to learn the model, by maximizing the conditional likelihood

$$\max_w \prod_{i=1}^{N} p(t^{(i)}|x^{(i)})$$

- Convert this into a minimization so that we can write the loss function
**Loss Function**

\[ p(t^{(1)}, \ldots, t^{(N)} | x^{(1)}, \ldots x^{(N)}) = \prod_{i=1}^{N} p(t^{(i)} | x^{(i)}) \]

\[ = \prod_{i=1}^{N} \left(1 - p(C = 0 | x^{(i)})\right)^{t^{(i)}} p(C = 0 | x^{(i)})^{1-t^{(i)}} \]

- It's convenient to take the logarithm and convert the maximization into minimization by changing the sign

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

- Why is this equivalent to maximizing the conditional likelihood?
- Is there a closed form solution?
- It's a convex function of \( w \). Can we get the global optimum?
Gradient Descent

\[
\min_w \ell(w) = \min_w \left\{ -\sum_{i=1}^{N} t^{(i)} \log(1 - p(C = 0|x^{(i)}, w)) - \sum_{i=1}^{N} (1 - t^{(i)}) \log p(C = 0|x^{(i)}, w) \right\}
\]

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

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

- But where is \( w \)?

\[
p(C = 0|x) = \frac{1}{1 + \exp(-w^T x - w_0)} \quad p(C = 1|x) = \frac{\exp(-w^T x - w_0)}{1 + \exp(-w^T x - w_0)}
\]

- You can write this in vector form

\[
\nabla \ell(w) = \left[ \frac{\partial \ell(w)}{\partial w_0}, \ldots, \frac{\partial \ell(w)}{\partial w_k} \right]^T, \quad \text{and} \quad \triangle (w) = -\lambda \nabla \ell(w)
\]
Let’s look at the updates

- The log likelihood is

\[ \ell_{\text{log-loss}}(\mathbf{w}) = -\sum_{i=1}^{N} t^{(i)} \log p(C = 1|\mathbf{x}^{(i)}, \mathbf{w}) - \sum_{i=1}^{N} (1-t^{(i)}) \log p(C = 0|\mathbf{x}^{(i)}, \mathbf{w}) \]

where the probabilities are

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

and \( z = \mathbf{w}^T \mathbf{x} + w_0 \)

- We can simplify

\[ \ell(\mathbf{w}) = \sum_{i} t^{(i)} \log(1 + \exp(-z^{(i)})) + \sum_{i} t^{(i)} z^{(i)} + \sum_{i} (1 - t^{(i)}) \log(1 + \exp(-z^{(i)})) \]

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

- Now it’s easy to take derivatives
Now it’s easy to take derivatives

Remember \( z = w^T x + w_0 \)

\[
\frac{\partial \ell}{\partial w_j} = \sum_i t^{(i)} x_j^{(i)} - x_j^{(i)} \cdot \frac{\exp(-z^{(i)})}{1 + \exp(-z^{(i)})}
\]

What’s \( x_j^{(i)} \)?

And simplifying

\[
\frac{\partial \ell}{\partial w_j} = \sum_i x_j^{(i)} \left( t^{(i)} - p(C = 1|x^{(i)}) \right)
\]

Don’t get confused with indices: \( j \) for the weight that we are updating and \( i \) for the training example

Logistic regression has linear decision boundary
Logistic regression vs least squares

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
  \[ p(w|\{t\}, \{x\}) \propto p(\{t\}|\{x\}, w) p(w) \]
  with \( \{t\} = (t^{(1)}, \ldots, t^{(N)}) \), 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

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

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

- This prior pushes parameters towards zero

Including this prior the new gradient is

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

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

- How do we decide the best value of \( \alpha \)?
Use of Validation Set

- 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.
- The parameter $\alpha$ is the importance of the regularization, and it’s a hyper-parameter.
Cross-Validation

- **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 \( C_p^n \) 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