Machine Learning: Basic Principles

Classification

Salo, September 2018

Guiding Motto

similar features give similar labels

Material

this lecture is inspired by

- video lectures of Andrew Ng
  - https://www.youtube.com/watch?v=-la3q9d7AKQ
  - https://www.youtube.com/watch?v=7F-CuXdTQ5k
- lecture notes
  http://cs229.stanford.edu/notes/cs229-notes1.pdf
- Ch. 2.2 of the tutorial "Kernel Methods in Computer Vision" by Ch. Lampert https://pub.ist.ac.at/~chl/papers/lampert-fnt2009.pdf
- lecture notes http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

In a Nutshell

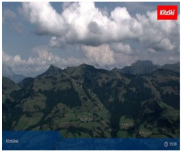
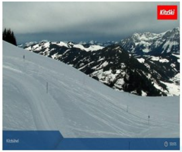
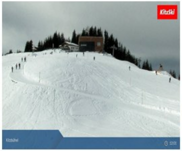
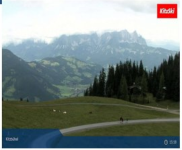
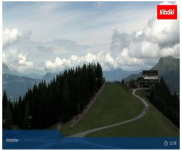
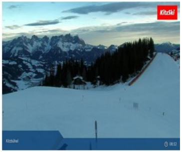
- consider some data set $\mathbb{X} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- data point with features $\mathbf{x} \in \mathbb{R}^d$ and label $y \in \{0, 1\}$
- a classifier $h(\mathbf{x})$ takes feature $\mathbf{x}$ as input and predicts label y
- classify data point $\hat{y} = 1$ if $h(\mathbf{x}) > 1/2$ and $\hat{y} = 0$ else
- learn/find optimal classifier using labeled data $\mathbb{X}$

Outline

Ski Resort Marketing

- you are working in the marketing agency of a ski resort
- hard disk full of webcam snapshots (gigabytes of data)
- want to group them into “winter” and ”summer” images
- you have only a few hours for this task ...

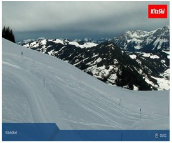
Webcam Snapshots

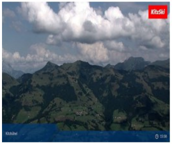
Labeled Webcam Snapshots

- create dataset $\mathbb{X}$ by randomly selecting $N = 6$ snapshots
- manually categorise/label them ($y^{(i)} = 1$ for summer)

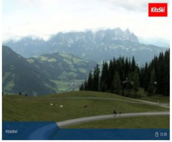
$y^{(2)} = 0$



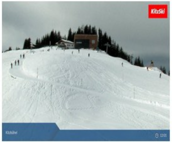
$y^{(1)} = 1$



$y^{(3)} = 1$

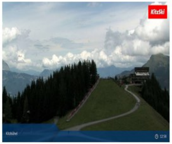


$y^{(4)} = 0$



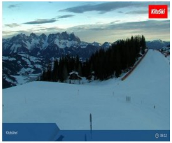
winter

$y^{(5)} = 1$



summer

$y^{(6)} = 0$



Towards an ML Problem

- we have few labeled snapshots in $\mathbb{X}$
- need an algorithm/method/software-app to automatically label all snapshots as either "winter" or "summer"
- each snapshot is several MByte large
- computational/time constraints force us to use more compact representation (features)
- what are good features for classifying summer vs. winter?

Redness, Greenness and Blueness

- summer images are expected to be more colourful
- winter images of Alps tend to contain much "white" (snow)
- lets use redness x_r, greenness x_g and blueness x_b

$$\text{redness } x_r := (1/K) \sum_{j \in \text{pixels}} \big(r[j] - (1/2)(g[j] + b[j])\big)$$

$$\text{greenness } x_g := (1/K) \sum_{j \in \text{pixels}} \big(g[j] - (1/2)(r[j] + b[j])\big)$$

$$\text{blueness } x_b := (1/K) \sum_{j \in \text{pixels}} \big(b[j] - (1/2)(r[j] + g[j])\big)$$

- K denotes total number of pixels in snapshot
- $r[j]$, $g[j]$, $b[j]$ denote red/green/blue intensity of pixel j

A Classification Problem

- labeled dataset $\mathbb{X} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- feature vector $\mathbf{x}^{(i)} = (x_r^{(i)}, x_g^{(i)}, x_b^{(i)})^T \in \mathbb{R}^3$
- set $y^{(i)}\!=\!1$ if ith snapshot taken in summer and $y^{(i)}\!=\!0$ else
- find a classifier $h(\cdot) : \mathbb{R}^3 \to \{0, 1\}$ with $y \approx h(\mathbf{x})$
- which hypothesis space $\mathcal{H}$ and loss $L((\mathbf{x}, y), h(\cdot))$ should we use?

Linear Regression Classifier

- lets first try to recycle ideas from linear regression
- use $\mathcal{H} = \{h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}, \text{ for } \mathbf{w} \in \mathbb{R}^d\}$ and squared error loss
- classify $\hat{y} = 1$ if $h^{(\mathbf{w})}(\mathbf{x}) > 1/2$ and $\hat{y} = 0$ else
- two shortcomings of this approach:
  - predictor $h^{(\mathbf{w})}(\mathbf{x})$ can be any real number, while $y \in \{0, 1\}$
  - squared error loss would penalize correct decisions

Outline

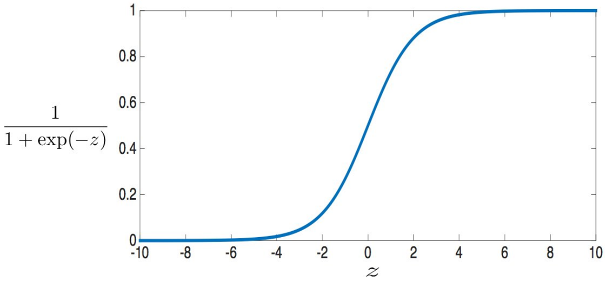
Taking Label Space Into Account

- lets exploit that labels y take only values 0 or 1
- use predictor $h(\cdot)$ with $h(\mathbf{x}) \in [0,1]$
- one particular choice is

$$h^{(\mathbf{w})}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \text{ with } \sigma(z) := 1/(1+\exp(-z))$$

- $\sigma(z)$ known as logistic or sigmoid function
- predictor $h^{(\mathbf{w})}(\mathbf{x})$ parametrized by weight $\mathbf{w} \in \mathbb{R}^d$

The Sigmoid Function



entire real line $\mathbb{R}$ "squashed" or "squeezed" into $[0, 1]$

A Probabilistic Interpretation

- LogReg predicts $y \in \{0,1\}$ by $h(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \in [0,1]$
- lets model the label y and features $\mathbf{x}$ as random variables
- features $\mathbf{x}$ are given/observed/measured
- conditional probabilities $P(y=1|\mathbf{x})$ and $P(y=0|\mathbf{x})$
- approximate $P(y=1|\mathbf{x})$ by $h^{(\mathbf{w})}(\mathbf{x})$
- for any label value $y \in \{0,1\}$, the following formula is valid

$$P(y|\mathbf{x}) = \left[h^{(\mathbf{w})}(\mathbf{x})\right]^{y} \left[1 - h^{(\mathbf{w})}(\mathbf{x})\right]^{(1-y)}$$

Logistic Regression

- max. likelihood $\max_{\mathbf{w}\in\mathbb{R}^d} \left[h^{(\mathbf{w})}(\mathbf{x})\right]^y \left[1-h^{(\mathbf{w})}(\mathbf{x})\right]^{(1-y)}$
- max. $P(y|\mathbf{x})$ equivalent to min. logistic loss

$$L((\mathbf{x},y),h^{(\mathbf{w})}(\cdot)) := -\log P(y|\mathbf{x})$$
$$= -y\log h^{(\mathbf{w})}(\mathbf{x}) - (1-y)\log(1-h^{(\mathbf{w})}(\mathbf{x}))$$

- choose **w** via empirical risk minimisation

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathcal{E}\{h^{(\mathbf{w})}(\cdot)|\mathbb{X}\} = \frac{1}{N}\sum_{i=1}^{N} L((\mathbf{x}^{(i)},y^{(i)}),h^{(\mathbf{w})}(\cdot))$$
$$= \frac{1}{N}\sum_{i=1}^{N} -y^{(i)}\log h^{(\mathbf{w})}(\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-h^{(\mathbf{w})}(\mathbf{x}^{(i)}))$$
$$= \frac{1}{N}\sum_{i=1}^{N} -y^{(i)}\log \sigma(\mathbf{w}^T\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))$$

ID Card of Logistic Regression

- input/feature space $\mathcal{X} = \mathbb{R}^d$
- label space $\mathcal{Y} = [0, 1]$
- hypothesis space $\mathcal{H} = \{h^{(\mathbf{w})}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$, with $\mathbf{w} \in \mathbb{R}^d\}$
- loss function $L((\mathbf{x}, y), h(\cdot)) = -y \log h(\mathbf{x}) - (1-y) \log(1-h(\mathbf{x}))$
- classify $\hat{y} = 1$ if $h^{(\mathbf{w})}(\mathbf{x}) \geq 1/2$ and $\hat{y} = 0$ otherwise

Classifying with Logistic Regression

- logistic regression problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} -y^{(i)}\log\sigma(\mathbf{w}^T\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))$$

- solved by optimal weight vector $\mathbf{w}_0$
- $h^{(\mathbf{w}_0)}(\mathbf{x})$ is an estimate for $P(y = 1|\mathbf{x})$
- classify $\hat{y} = 1$ if $h^{(\mathbf{w}_0)}(\mathbf{x}) \geq 1/2$ and $\hat{y} = 0$ else
- partitions $\mathcal{X}$ in $\mathcal{R}_1 = \{\mathbf{x} : h(\mathbf{x}) \geq 1/2\}$ and $\mathcal{R}_0 = \{\mathbf{x} : h(\mathbf{x}) < 1/2\}$

The Decision Boundary of Logistic Regression

Learning a Logistic Regression Model

- logistic regression problem

$$\mathbf{w}_0 = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} -y^{(i)}\log\sigma(\mathbf{w}^T\mathbf{x}^{(i)}) - (1-y^{(i)})\log(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))}_{f(\mathbf{w})}$$

- in contrast to LinReg, no closed-form for $\mathbf{w}_0$
- however, we can use GD:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha\nabla f(\mathbf{w}^{(k)})$$

- does $\mathbf{w}^{(k)}$ always converge to $\mathbf{w}_0$?
- choice for step-size $\alpha > 0$ crucial for convergence

A Learning Algorithm for Classification

- input: labeled data set $\mathbb{X}$, step-size or learning rate $\alpha > 0$
- output: weight vector $\mathbf{w}$ for classifier $h^{(\mathbf{w})}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$
- initalize: $k := 0$ and $\mathbf{w}_0 := 0$
- until stopping criterion satisfied do
  - $\mathbf{w}^{(k+1)} := \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} \mathcal{E}\{h^{(\mathbf{w}^{(k)})} | \mathbb{X}\}$
  - $k := k + 1$
- set $\mathbf{w} := \mathbf{w}^{(k)}$

Outline

Binary Linear Classifiers

- logistic regression is linear classifier (DB is hyperplane)
- linear classifier $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ specified by normal vector $\mathbf{w}$
- let us from now on code the binary labels as $+1$ and -1
- classify $\hat{y} = 1$ if $h^{(\mathbf{w})}(\mathbf{x}) > 0$ and $\hat{y} = -1$ else
- learn suitable weight $\mathbf{w}$ by empirical risk minimization
- seemingly, squared error loss is not good for binary labels

Minimizing Error Probability

- eventually, we aim at low error probability $P(\hat{y} \neq y)$
- using 0/1-loss $L((\mathbf{x}, y), h(\cdot)) = \mathcal{I}(\hat{y} \neq y)$ we can approximate

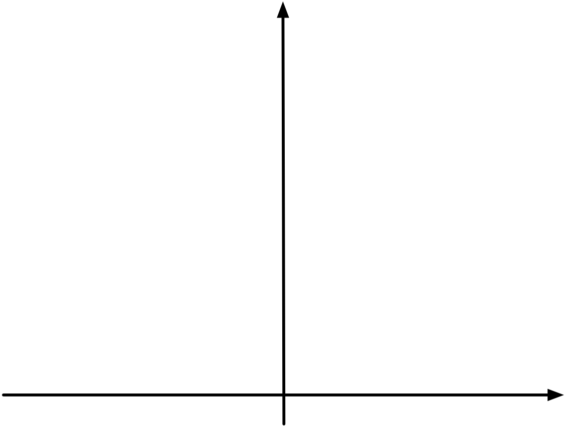
$$P(\hat{y} \neq y) \approx (1/N) \sum_{i=1}^{N} L((\mathbf{x}^{(i)}, y^{(i)}), h(\cdot))$$

- the optimal classifier is then obtained by

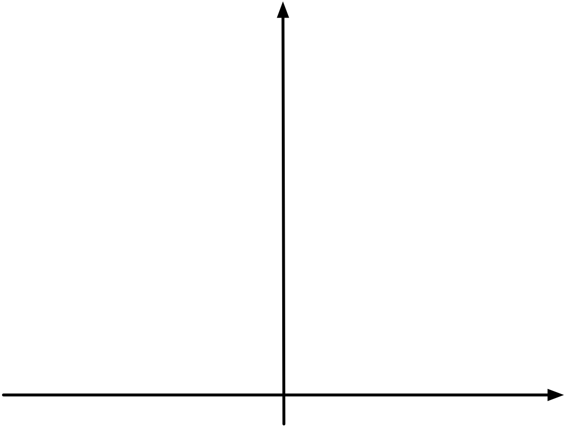
$$\min_{h(\cdot) \in \mathcal{H}} \sum_{i=1}^{N} L((\mathbf{x}^{(i)}, y^{(i)}), h(\cdot))$$

- non-convex and non-smooth optimization problem !

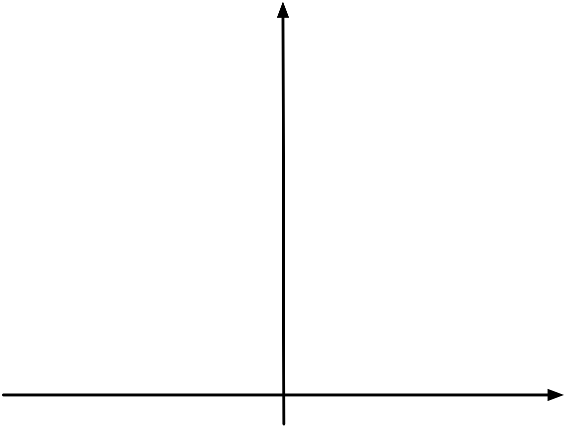
The 0/1 Loss



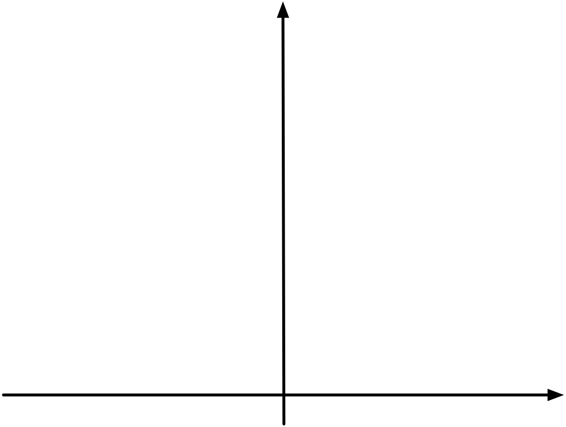
The Hinge Loss



The Hinge Loss ($y = 1$)



The Hinge Loss ($y = -1$)



Learning Linear Classifier using Hinge Loss

- linear classifier $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with some weight $\mathbf{w}$

- measure goodness of $h^{(\mathbf{w})}(\cdot)$ using **hinge loss**

$$\begin{aligned} L((\mathbf{x}, y), h^{(\mathbf{w})}) &= \max\{0, 1 - y \cdot h^{(\mathbf{w})}(\mathbf{x})\} \\ &= \max\{0, 1 - y \cdot (\mathbf{w}^T\mathbf{x})\} \end{aligned}$$

- learn optimal weight $\mathbf{w}_0$ via empirical risk minimization

$$\begin{aligned} \mathbf{w}_0 = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \mathcal{E}(h^{(\mathbf{w})}|\mathbb{X}) &:= \frac{1}{N}\sum_{i=1}^{N} L((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}(\cdot)) \\ &= \frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)})\} \end{aligned}$$

SVC Maximizes Margin

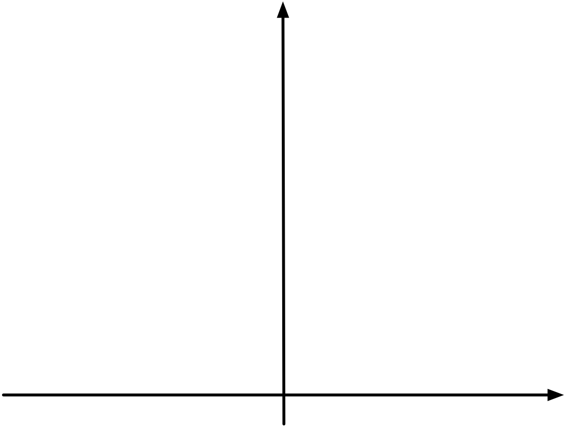
- we can rewrite hinge loss as

$$L((\mathbf{x}, y), h^{(\mathbf{w},b)}) = \max\{0, 1 - y \cdot (\mathbf{w}^T\mathbf{x})\}$$
$$= \min_{\xi \geq 0} \xi \text{ s.t. } \xi \geq 1 - \underbrace{y \cdot (\mathbf{w}^T\mathbf{x})}_{\text{“}margin\text{''}}$$

- minimizing hinge loss equivalent to maximizing margin

$$\min_{\mathbf{w} \in \mathbb{R}^d} \mathcal{E}(h^{(\mathbf{w})} | \mathbb{X}) = \frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)})\}$$
$$= \frac{1}{N} \min_{\xi^{(i)} \geq 0} \sum_{i=1}^{N} \xi^{(i)} \text{ s.t. } y^{(i)} \cdot (\mathbf{w}^T\mathbf{x}^{(i)}) \geq 1 - \xi^{(i)}$$

SVC Maximizes Margin



ID Card of Support Vector Classifier

- input/feature space $\mathcal{X} = \mathbb{R}^d$
- label space $\mathcal{Y} = \{-1, 1\}$
- hypothesis space $\mathcal{H} = \{h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, with $\mathbf{w} \in \mathbb{R}^d\}$
- loss function $L((\mathbf{x}, y), h(\cdot)) = \max\{0, 1 - y \cdot h^{(\mathbf{w})}(\mathbf{x})\}$
- classify $\hat{y} = 1$ if $h^{(\mathbf{w})}(\mathbf{x}) \geq 0$ and $\hat{y} = -1$ else

Outline

So What?

- LogReg and SVC both linear classifiers (DB is hyperplane)
- LogReg uses logistic loss; based on maximum likelihood
- SVC uses hinge-loss and amounts to max. margin
- model complexity is d and independent of sample size N
- once we learnt optimal parameter $\mathbf{w}_{\text{opt}}$, we can discard data !

Logistic Regression at a Glance

- is a linear classifier using hypothesis $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- allows for probabilistic interpretation of $h^{(\mathbf{w})}(\mathbf{x})$
- closely related to Bayes' classifier (see next lecture!)
- ERM amounts to SMOOTH convex problem

Support Vector Classifier (Machine) at a Glance

- is a linear classifier using hypothesis $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- based on geometry of $\mathbb{X}$ in feature space (max. margin)
- can be extended via feature maps (kernel methods)
- ERM amounts to NON-SMOOTH cvx opt problem