Lecture 5: gradient descent; bias/variance tradeoff

TTIC 31020: Introduction to Machine Learning

Instructors: Greg Shakhnarovich
Suriya Gunasekar

TTI-Chicago

October 11, 2016

Review: regularization

- General form of a regularized objective:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \log p(\text{data}_i;\ \mathbf{w}) - \text{penalty}(\mathbf{w}) \right\}$$

- Ridge regression:

$$\mathbf{w}^*_{\text{ridge}} = \operatorname*{argmin}_{\mathbf{w}} \left\{ \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{m} w_j^2 \right\}$$

convex, closed form solution $\widehat{\mathbf{w}}^*_{\text{ridge}} = \left(\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$

- Lasso:

$$\mathbf{w}^*_{\text{lasso}} = \operatorname*{argmax}_{\mathbf{w}} \left\{ -\sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 - \lambda \sum_{j=1}^{m} |w_j| \right\}$$

convex, no closed form (need numerical optimization tools)

Review: geometry of regularization

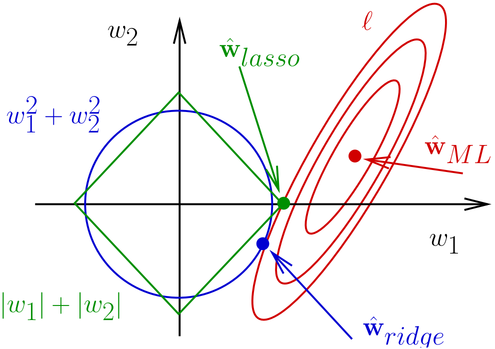
- Can write unconstrained optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{m} |w_j|^p$$

as an equivalent constrained problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

subject to $\sum_{j=1}^{m} |w_j|^p \leq t$

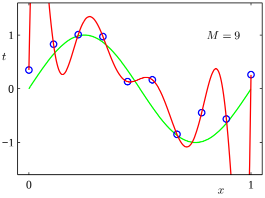


- $p = 1$ may lead to sparsity, $p = 2$ generally won't

Review: regularization and overfitting

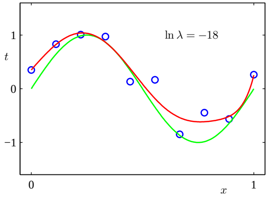
- Intuition: limiting norm of $\mathbf{w}$ $\Rightarrow$limiting description length $\Rightarrow$limiting complexity $\Rightarrow$controlling overfitting
- Departure from pure ERM principle
- Different from "normal" model selection: we can use the richest model class, but control overfitting via value of λ

$\lambda = 0$



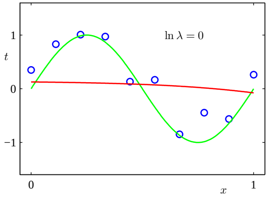
$\|\mathbf{w}^*\|^2 > 10^{12}$

$\lambda = e^{-18}$



$\|\mathbf{w}^*\|^2 \approx 21595$

$\lambda = 1$



$\|\mathbf{w}^*\|^2 \approx 0.027$

Roadmap

So far:

- General regression, with squared loss
- Tools to deal with overfitting:
  Model selection by heldout/cross validation
  Regularization using shrinkage: ridge (closed form) or lasso (no closed form)

Today:

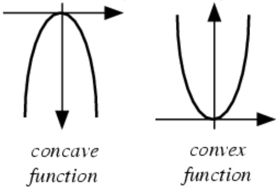
- Gradient descent
- Deeper understanding of overfitting: bias/variance tradeoff
- Intro to classification

Beyond closed form solution

- So far: solve (least squares) regression with a closed form solution

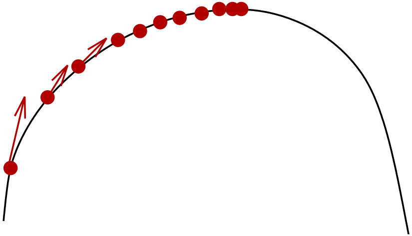
$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- Sometimes we can not do this. E.g., the data matrix is too large to compute the pseudoinverse for
- If we move away from simple squared loss (e.g., in PS1: asymmetric loss) also lose the closed form solution
- Alternative: numerical optimization – gradient descent
- Consider (for now) convex or concave functions



Gradient ascent/descent

- The idea behind gradient ascent: “hill climbing” on the function surface.



- Start at a (random) location
- Make steps in the direction of maximal altitude increase.

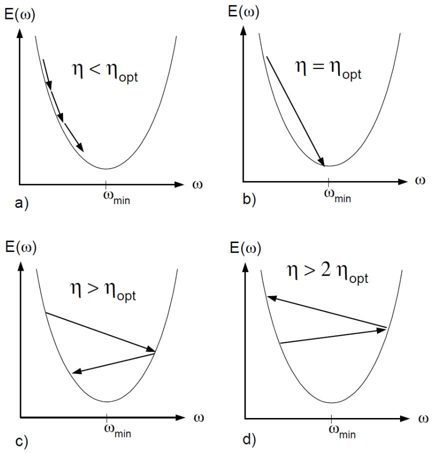
- An equivalent: gradient *descent* on the *convex* loss $-\log p\left(y \,|\, \mathbf{x}; \mathbf{w}\right)$

Gradient descent algorithm on $f(\mathbf{X}, \mathbf{y}; \mathbf{w})$

- Iteration counter $t = 0$
- Initialize $\mathbf{w}^{(t)}$ (to zero or a small random vector)
- for $t = 1, \ldots$:
  compute gradient
  $$\mathbf{g}^{(t)} = \nabla f\left(\mathbf{X}, \mathbf{y}; \mathbf{w}^{(t-1)}\right)$$
  update model
  $$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \mathbf{g}^{(t)}$$
  check for convergence
- The *learning rate* η controls the step size

Gradient descent convergence

- Generally, for convex functions, gradient descent will converge
- Setting the learning rate η may be very important to ensure rapid convergence



From Lecun et al, 1996

Gradient descent convergence

- A lot of theory on convergence of gradient descent
- Usually relies on various properties of the objective function: strong convexity, smoothness, etc.
- In practice, need to monitor the objective, tweak learning rate, and consider stopping ("convergence") criteria
- Common criteria (often use a combination):
  - Maximum number of iterations (time budget)
  - Minimum required change in objective value (loss)
  - Minimum required change in model parameters ($\mathbf{w}$)
- If stopped because of max iterations: may not have converged
- Problematic criteria: monitor absolute (not relative) value of something like objective or parameters. Often hard to know what the "right" value for these is.

A bit of estimation theory

- An *estimator* $\widehat{\theta}$ of a parameter θ is a function that for data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ produces estimate (estimated value) $\widehat{\theta}$.
- Examples:
  ML estimator for a Gaussian mean, given X, produces an estimate (vector) $\widehat{\boldsymbol{\mu}}$.
  ML estimator for linear regression parameters $\mathbf{w}$ under Gaussian noise model
- The estimate $\hat{\theta}$ is a random variable since it is based on a randomly drawn set X.
- We can talk about $E_X\left[\hat{\theta}\right]$ and $\mathrm{var}(\hat{\theta})$.
  (When θ is a vector, we have $\mathrm{Cov}(\hat{\theta})$.)
  - *Analysis done assuming that the data* **is** *distributed according to* $p(\mathbf{x}; \theta)$ *where* θ *is the true parameter value!*

Bias of an estimator

- The *bias* of an estimator $\hat{\theta}$ is defined as

$$\mathrm{bias}(\hat{\theta}) \triangleq E_X\left[\hat{\theta} - \theta\right].$$

  i.e. the expected deviation of the estimate from the correct parameter (taken over all possible sets of N examples).
- An *unbiased* estimator therefore satisfies $E_X\left[\hat{\theta}\right] = \theta$.
- Example: ML estimators of 1D Gaussian parameters
  $\hat{\mu}_{ML} = \frac{1}{N}\sum_i x_i$, $\qquad \widehat{\sigma^2}_{ML} = \frac{1}{N}\sum_i (x_i - \hat{\mu})^2$.

Bias of an estimator

- The *bias* of an estimator $\hat{\theta}$ is defined as

$$\text{bias}(\hat{\theta}) \triangleq E_X\left[\hat{\theta} - \theta\right].$$

  i.e. the expected deviation of the estimate from the correct parameter (taken over all possible sets of N examples).
- An *unbiased* estimator therefore satisfies $E_X\left[\hat{\theta}\right] = \theta$.
- Example: ML estimators of 1D Gaussian parameters
  $\hat{\mu}_{ML} = \frac{1}{N}\sum_i x_i, \qquad \widehat{\sigma^2}_{ML} = \frac{1}{N}\sum_i (x_i - \hat{\mu})^2.$
- Turns out $\hat{\mu}$ is unbiased;

Bias of an estimator

- The *bias* of an estimator $\hat{\theta}$ is defined as

$$\text{bias}(\hat{\theta}) \triangleq E_X\left[\hat{\theta} - \theta\right].$$

i.e. the expected deviation of the estimate from the correct parameter (taken over all possible sets of N examples).

- An *unbiased* estimator therefore satisfies $E_X\left[\hat{\theta}\right] = \theta$.
- Example: ML estimators of 1D Gaussian parameters
$\hat{\mu}_{ML} = \frac{1}{N}\sum_i x_i, \qquad \widehat{\sigma^2}_{ML} = \frac{1}{N}\sum_i (x_i - \hat{\mu})^2.$
- Turns out $\hat{\mu}$ is unbiased; however, $\widehat{\sigma}_{ML}$ *underestimates* the variance in the data!

$$E\left[\widehat{\sigma^2}_{ML}\right] = \frac{N-1}{N}\sigma^2.$$

Consistency of an estimator

- With enough data, bias *may* not be so much of a problem.
- Consider an infinite sequence $\mathbf{x}_1, \ldots$ and define $\hat{\theta}_N$ an estimate obtained on $\mathbf{x}_1, \ldots, \mathbf{x}_N$.
- An estimator $\hat{\theta}$ is *consistent* if

$$\lim_{N \to \infty} \hat{\theta}_N = \theta.$$

Note: this limit is *in probability*.

- So, $\widehat{\sigma^2}_{ML} = \frac{1}{N} \sum_i (x_i - \mu_{ML})^2$, even though biased, is a consistent estimator of σ^2.

Estimation and regression

- The true model: $y = F(\mathbf{x}) + \nu$, zero-mean additive noise ν.
- We approximate F by $f(\mathbf{x}; \hat{\mathbf{w}}) \in \mathcal{F}$, with $\hat{\mathbf{w}}$ estimated from data X.
- Here we will focus on point-wise estimate of F on any $\mathbf{x}$
- Consider:
  $\hat{f}(\mathbf{x}) \;=\; f(\mathbf{x}; \hat{\mathbf{w}})$ estimate based on particular X,
  $\bar{f}(\mathbf{x}) \;=\; E_X\left[f(\mathbf{x}; \hat{\mathbf{w}})\right]$ average estimate over training sets X,
  $f^*(\mathbf{x}) \;=\; f(\mathbf{x}; \operatorname{argmin}_{\mathbf{w}} E_{p(\mathbf{x},y)}\left[(y - f(\mathbf{x}; \mathbf{w}))^2\right])$ the best estimate by a function $\in \mathcal{F}$.

Bias-variance decomposition

- Consider squared loss $\left(\hat{\theta}-\theta\right)^2$.
- Denote $\bar{\theta}=E\left[\hat{\theta}\right]$. Then, the expected error:

$$E\left[(\hat{\theta}-\theta)^2\right]=E\left[(\hat{\theta}-\bar{\theta}+\bar{\theta}-\theta)^2\right]$$

Bias-variance decomposition

- Consider squared loss $\left(\hat{\theta} - \theta\right)^2$.
- Denote $\bar{\theta} = E\left[\hat{\theta}\right]$. Then, the expected error:

$$\begin{aligned} E\left[(\hat{\theta} - \theta)^2\right] &= E\left[(\hat{\theta} - \bar{\theta} + \bar{\theta} - \theta)^2\right] \\ &= E\left[(\hat{\theta} - \bar{\theta})^2\right] + 2(\bar{\theta} - \theta)\underbrace{E\left[\hat{\theta} - \bar{\theta}\right]}_{=0} + E\left[(\bar{\theta} - \theta)^2\right] \end{aligned}$$

Bias-variance decomposition

- Consider squared loss $\left(\hat{\theta}-\theta\right)^2$.
- Denote $\bar{\theta}=E\left[\hat{\theta}\right]$. Then, the expected error:

$$
\begin{aligned}
E\left[(\hat{\theta}-\theta)^2\right] &= E\left[(\hat{\theta}-\bar{\theta}+\bar{\theta}-\theta)^2\right] \\
&= E\left[(\hat{\theta}-\bar{\theta})^2\right]+2(\bar{\theta}-\theta)\underbrace{E\left[\hat{\theta}-\bar{\theta}\right]}_{=0}+E\left[(\bar{\theta}-\theta)^2\right] \\
&= E\left[(\hat{\theta}-\bar{\theta})^2\right]
\end{aligned}
$$

Bias-variance decomposition

- Consider squared loss $\left(\hat{\theta} - \theta\right)^2$.
- Denote $\bar{\theta} = E\left[\hat{\theta}\right]$. Then, the expected error:

$$
\begin{aligned}
E\left[(\hat{\theta} - \theta)^2\right] &= E\left[(\hat{\theta} - \bar{\theta} + \bar{\theta} - \theta)^2\right] \\
&= E\left[(\hat{\theta} - \bar{\theta})^2\right] + 2(\bar{\theta} - \theta)\underbrace{E\left[\hat{\theta} - \bar{\theta}\right]}_{=0} + E\left[(\bar{\theta} - \theta)^2\right] \\
&= E\left[(\hat{\theta} - \bar{\theta})^2\right] + (\bar{\theta} - \theta)^2
\end{aligned}
$$

Bias-variance decomposition

- Consider squared loss $\left(\hat{\theta} - \theta\right)^2$.
- Denote $\bar{\theta} = E\left[\hat{\theta}\right]$. Then, the expected error:

$$
\begin{aligned}
E\left[(\hat{\theta} - \theta)^2\right] &= E\left[(\hat{\theta} - \bar{\theta} + \bar{\theta} - \theta)^2\right] \\
&= E\left[(\hat{\theta} - \bar{\theta})^2\right] + 2(\bar{\theta} - \theta)\underbrace{E\left[\hat{\theta} - \bar{\theta}\right]}_{=0} + E\left[(\bar{\theta} - \theta)^2\right] \\
&= E\left[(\hat{\theta} - \bar{\theta})^2\right] + (\bar{\theta} - \theta)^2 \\
&= \mathrm{var}(\hat{\theta}) + \mathrm{bias}^2(\hat{\theta}).
\end{aligned}
$$

- Recall expected squared loss decomposition:
  - bias^2 term $\Leftrightarrow$ approximation error,
  - variance $\Leftrightarrow$ estimation error due to finite data.

Bias-variance in regression

- For a single $\mathbf{x}_0$:

$$E_X\left[\left(y_0-\hat{f}(\mathbf{x}_0)\right)^2\right] \;=\; \left(y_0-\bar{f}(\mathbf{x}_0)\right)^2 + \underbrace{E_X\left[\left(\hat{f}(\mathbf{x}_0)-\bar{f}(\mathbf{x}_0)\right)^2\right]}_{variance}.$$

- The first term can be further decomposed:

$$\left(y_0-\bar{f}(\mathbf{x}_0)\right)^2 = \underbrace{\left(y_0-F(\mathbf{x}_0)\right)^2}_{\text{noise}} + \underbrace{\left(F(\mathbf{x}_0)-\bar{f}(\mathbf{x}_0)\right)^2}_{\mathrm{bias}^2}$$

  - See derivation notes
  - The noise term is *irreducible* (independent of data)
  - The bias^2 term is due to difference between f and F.

Need to integrate all of this over $\mathbf{x}_0, y_0$ to get the *expected* bias and variance. (see notes)

Bias-variance tradeoff

- So,

$$E\left[\text{squared loss}\right] = \mathrm{bias}^2 + \mathrm{var} + \text{noise}.$$

- Can do nothing about noise
- Ideally, want to minimize bias and variance; can we drive both to zero?

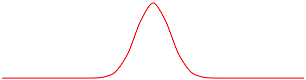
Bias-variance tradeoff: theory

- *Cramer-Rao inequality*: for an unbiased estimator $\hat{\theta}_N$,

$$\mathrm{var}(\hat{\theta}_N) \geq \frac{1}{E\left[\left(\frac{\partial}{\partial\theta}\log p(\mathbf{X};\theta)\right)^2\right]}.$$

- The *Fisher information* $\mathcal{I}(\theta) = E\left[\left(\frac{\partial}{\partial\theta}\log p(\mathbf{X};\theta)\right)^2\right]$ is related to the shape of $p(\mathbf{x};\theta)$. Intuitively, it measures the amount of information data X provides about parameter θ.

$\mathcal{N}\left(x;\,\mu,\sigma^2\right)$
Known σ^2

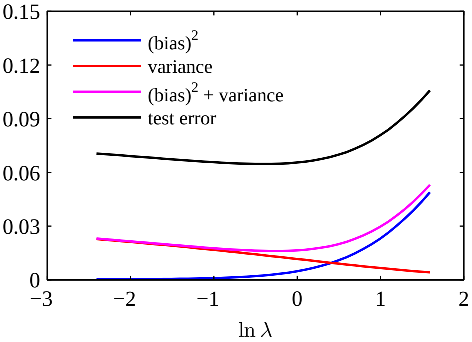


large $\mathcal{I}(\mu)$

small $\mathcal{I}(\mu)$

Regularization and bias/variance tradeoff

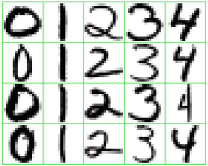
- Recall: $E\left[\text{squared loss}\right] = \text{bias}^2 + \text{var} + \text{noise}$.
- In hindsight:



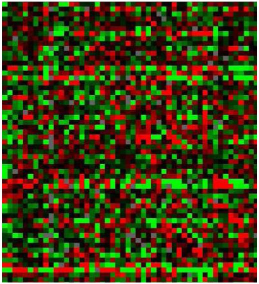
- In reality: often need to rely on procedure like (cross) validation

Classification

- Shifting gears: classification. Many successful applications of ML: vision, speech, medicine, etc.
- Setup: need to map $\mathbf{x} \in \mathcal{X}$ to a *label* $y \in \mathcal{Y}$.
- Examples:



digits recognition;
$\mathcal{Y} = \{0, \ldots, 9\}$



prediction from microarray data;
$\mathcal{Y} = \{$desease present/absent$\}$

Classification as regression

- Suppose we have a binary problem, $y \in \{-1, 1\}$
- Idea: treat it as regression, with squared loss
- Assuming the standard model $y = f(\mathbf{x}; \mathbf{w}) + \nu$, and solving with least squares, we get $\hat{\mathbf{w}}$.
- This corresponds to squared loss as a measure of classification performance! Does this make sense?

Classification as regression

- Suppose we have a binary problem, $y \in \{-1, 1\}$
- Idea: treat it as regression, with squared loss
- Assuming the standard model $y = f(\mathbf{x}; \mathbf{w}) + \nu$, and solving with least squares, we get $\hat{\mathbf{w}}$.
- This corresponds to squared loss as a measure of classification performance! Does this make sense?
- How do we decide on the label based on $f(\mathbf{x}; \hat{\mathbf{w}})$?

Classification as regression

$$f(\mathbf{x}; \hat{\mathbf{w}}) \; = \; w_0 + \hat{\mathbf{w}} \cdot \mathbf{x}$$

- Can't just take $\hat{y} = f(\mathbf{x}; \hat{\mathbf{w}})$ since it won't be a valid label.
- A reasonable *decision rule*:

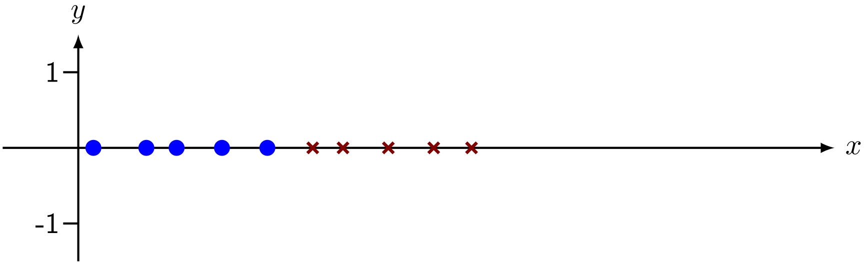
  decide on $\hat{y} = 1$ if $f(\mathbf{x}; \hat{\mathbf{w}}) \geq 0$, otherwise $\hat{y} = -1$.

  $$\hat{y} \; = \; \mathrm{sign}\left(w_0 + \hat{\mathbf{w}} \cdot \mathbf{x}\right)$$

- This specifies a *linear classifier*:
  - The linear *decision boundary* (hyperplane) given by the equation $w_0 + \hat{\mathbf{w}} \cdot \mathbf{x} = 0$ separates the space into two "half-spaces".

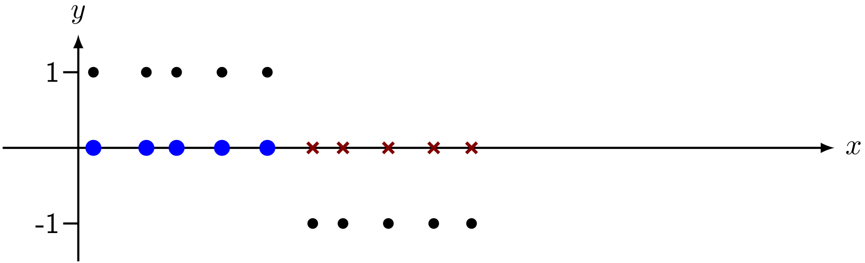
Classification as regression: example

- A 1D example:



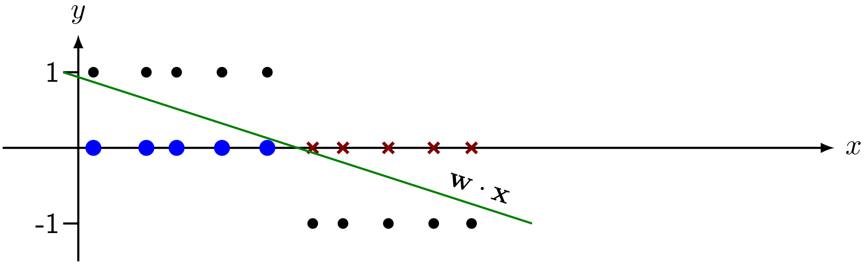
Classification as regression: example

- A 1D example:



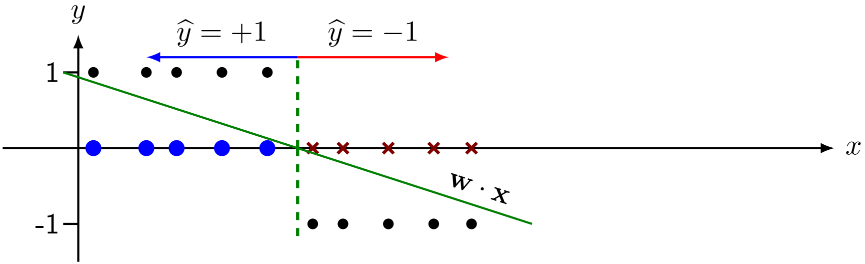
Classification as regression: example

- A 1D example:



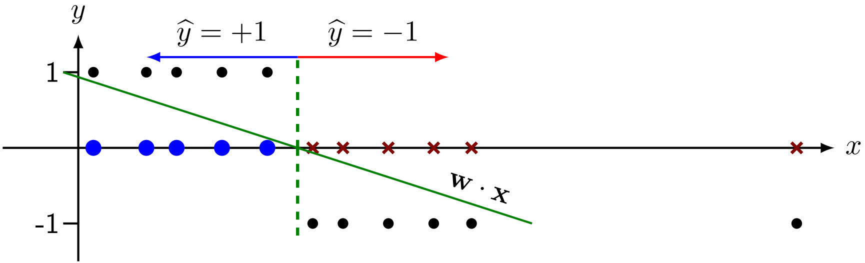
Classification as regression: example

- A 1D example:



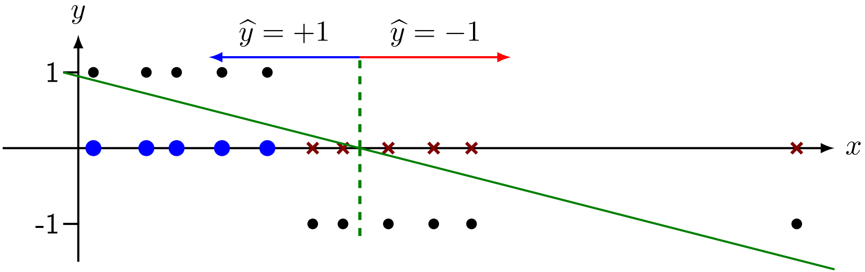
Classification as regression: example

- A 1D example:



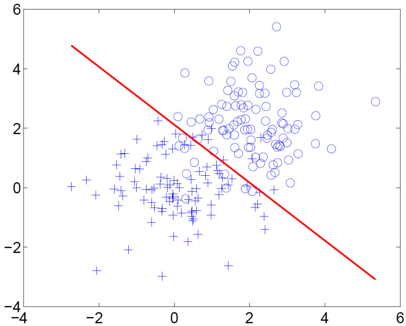
Classification as regression: example

- A 1D example:

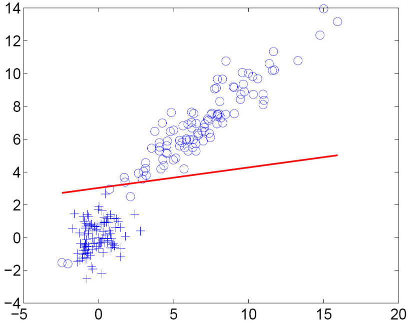


Classification as regression

- Same effect in 2D:

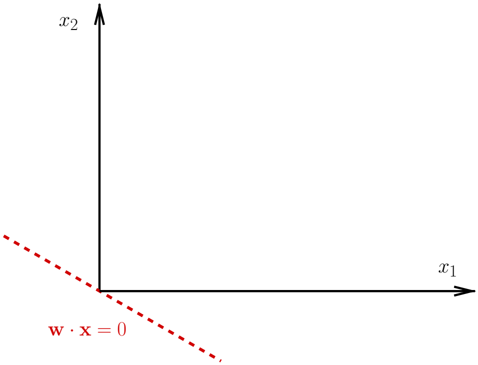


Seems to work well here

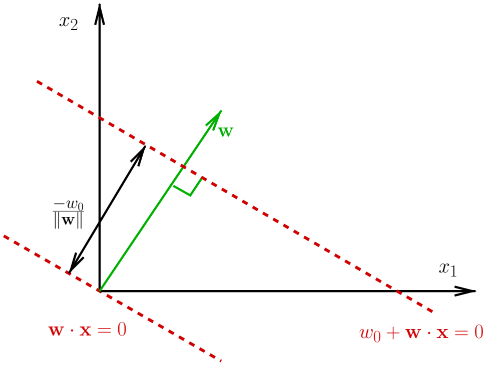


but not so well here

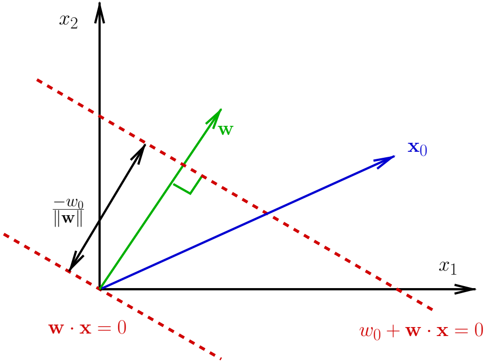
Geometry of projections



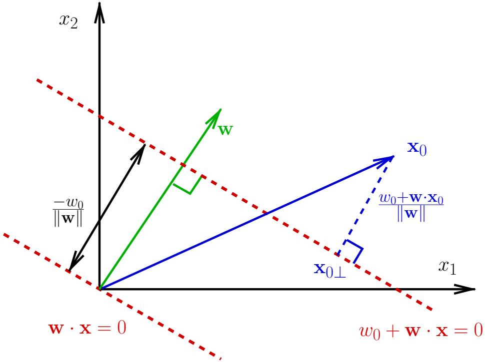
Geometry of projections



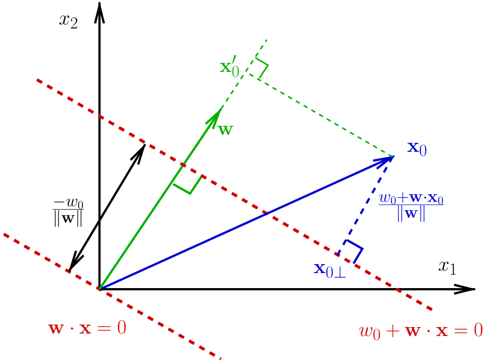
Geometry of projections



Geometry of projections

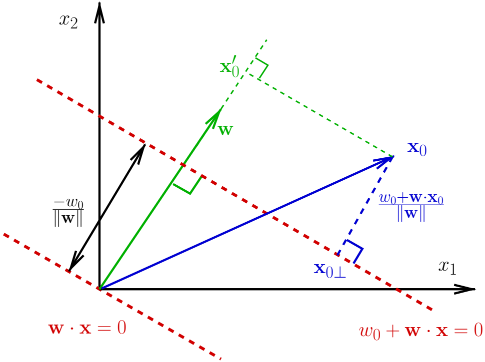


Geometry of projections



- $\mathbf{w}\cdot\mathbf{x}=0$: a line passing through the origin and *orthogonal* to $\mathbf{w}$
- $\mathbf{w}\cdot\mathbf{x}+w_0=0$ shifts the line along $\mathbf{w}$.

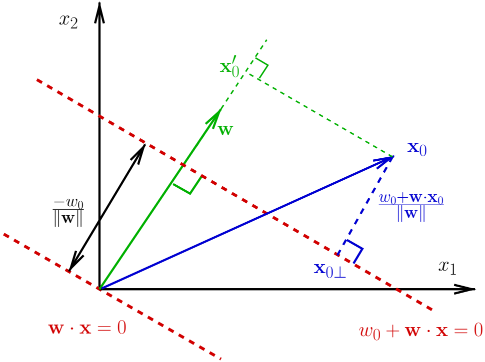
Geometry of projections



- $\mathbf{w} \cdot \mathbf{x} = 0$: a line passing through the origin and *orthogonal* to $\mathbf{w}$
- $\mathbf{w} \cdot \mathbf{x} + w_0 = 0$ shifts the line along $\mathbf{w}$.

- $\mathbf{x}'$ is the projection of $\mathbf{x}$ on $\mathbf{w}$.

Geometry of projections



- $\mathbf{w}\cdot\mathbf{x}=0$: a line passing through the origin and *orthogonal* to $\mathbf{w}$
- $\mathbf{w}\cdot\mathbf{x}+w_0=0$ shifts the line along $\mathbf{w}$.

- $\mathbf{x}'$ is the projection of $\mathbf{x}$ on $\mathbf{w}$.
- Set up a new 1D coordinate system: $\mathbf{x} \rightarrow (w_0+\mathbf{w}\cdot\mathbf{x})/\|\mathbf{w}\|$.

Linear classifiers

$$\hat{y} = h(\mathbf{x}) \; = \; \text{sign}\left(w_0 + \mathbf{w} \cdot \mathbf{x}\right)$$

- Classifying using a linear decision boundary effectively reduces the data dimension to 1.
- Need to find $\mathbf{w}$ (direction) and w_0 (location) of the boundary
- Want to minimize the expected zero/one loss for classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$, which for $(\mathbf{x}, y)$ is

$$L(h(\mathbf{x}), y) \; = \; \begin{cases} 0 & \text{if } h(\mathbf{x}) = y, \\ 1 & \text{if } h(\mathbf{x}) \neq y. \end{cases}$$